Quelques modèles et méthodes pour l'étude de la cognition

Pierre Courrieu

Université de Provence

Aix-Marseille I

Quelques Modèles et Méthodes pour l'Etude de la Cognition

dossier présenté pour l'obtention d'une

Habilitation à Diriger des Recherches

par

Pierre Courrieu

Chargé de Recherche au CNRS

Laboratoire de Psychologie Cognitive – UMR 6146

Centre National de la Recherche Scientifique

Université de Provence

Aix-Marseille I

Marseille - 2011

Quelques Modèles et Méthodes pour l'Etude de la Cognition

dossier présenté pour l'obtention d'une

Habilitation à Diriger des Recherches

par

Pierre Courrieu

Chargé de Recherche au CNRS

Laboratoire de Psychologie Cognitive – UMR 6146

Centre National de la Recherche Scientifique

SOMMAIRE

AVANT-PROPOS

Le dossier qui suit résume 26 années d'une carrière qui fut souvent solitaire, lorsque la matière traitée ne se prêtait guère à une réflexion collective, mais où j'ai aussi eu l'occasion de partager de bonnes questions (et de bonnes réponses) avec quelques-uns de mes pairs. J'ai même eu le privilège, trop rare sans doute, de guider les premiers pas dans la recherche de quelques étudiants très motivés. Initialement expérimentateur en psychologie cognitive, je suis entré au CNRS fin 1984 avec ce profil, après avoir soutenu une thèse de troisième cycle sur la reconnaissance visuelle des mots. Puis, la vague connexionniste de la décennie 80 et le développement des sciences cognitives aidant, il a été recommandé aux chercheurs de devenir interdisciplinaires, et je me suis aussitôt lancé avec enthousiasme dans la modélisation neurocomputationnelle. De fil en aiguille, je me suis rapproché de plus en plus des mathématiques appliquées, où se trouve la solution de bien des problèmes liés au développement des modèles, au calcul de leurs paramètres, et même à leur validation, ce qui nous ramène finalement à l'expérimentation humaine, mais avec une vision un peu différente de celle de l'expérimentaliste, encore que les faits s'avèrent toujours aussi "têtus". On peut considérer qu'il s'agit, en fait, d'une approche de psychologie mathématique, mais avec une dimension interdisciplinaire que la psychologie mathématique ne revendique pas habituellement. De fait, je pense que mes publications ont plus souvent intéressé des collègues travaillant dans des domaines de sciences pour l'ingénieur plutôt qu'en psychologie. Ceci n'est pas forcément définitif, et je tente, depuis un certain temps, un retour à des supports de publication plus familiers aux psychologues.

Ma démarche de modélisation s'inspire du constat, de prime abord un peu pessimiste, de P. Valéry: "tout ce qui est simple est faux, tout ce qui est compliqué est inutilisable". Ceci est vrai pour la psychologie cognitive, comme pour toute science, sans doute. Le rôle de la modélisation mathématique est précisément de rendre simple une complexité naturelle qui, sans cela, serait inabordable. Il s'agit, suivant le mot de J. Perrin, de "remplacer du visible compliqué par de l'invisible simple". En ce qui concerne la cognition humaine, c'est là un pari qui est encore bien loin d'être gagné, mais je ne doute pas que ce soit le seul objectif raisonnable à terme.

Ce document se subdivise en deux parties principales de natures différentes. La première partie "Activités Scientifiques" récapitule les informations que l'impétrant est réglementairement tenu de fournir: curriculum vitae, études suivies, liste des publications, activités scientifiques et de formation à la recherche, attestation de thèse de 3ème cycle. La seconde partie "Dossier de Travaux" présente une sélection de douze textes publiés dans des revues scientifiques internationales à comité de lecture. La présentation est organisée en six sections thématiques, chacune incluant de un à trois textes représentatifs, après une brève introduction. Toutes mes publications n'apparaissent pas dans cette présentation, et j'ai privilégié ceux de mes thèmes de recherche qui sont toujours en développement. Cette seconde partie se termine par un résumé de travaux en cours et quelques perspectives destinées à convaincre le lecteur que cette Habilitation à Diriger des Recherches n'est nullement un aboutissement, mais devrait au contraire se prolonger dans des investigations dont les précédentes ne sont que les prémices.

PREMIERE PARTIE

Activités Scientifiques

CURRICULUM VITAE

COURRIEU, Pierre

Chargé de Recherche de Première Classe (CR1)

Centre National de la Recherche Scientifique (CNRS)

Institut des Sciences Biologiques (INSB)

Section 27 du Comité National: Comportement, Cognition, Cerveau

Affectation actuelle:

Laboratoire de Psychologie Cognitive (LPC), UMR CNRS-UP N° 6146, Pôle 3C, Université de Provence, Centre Saint-Charles, 3, place Victor Hugo, 13331 Marseille Cedex 3

Directeur de l'Unité: J. Grainger

Equipe: Perception & Attention (responsable F. Vitu-Thibault)

Contact:

Tel.: (33 / 0) 4 42 22 87 77, (33 / 0) 4 13 55 09 89, Fax: (33 / 0) 4 13 55 09 98

E-mail: pierre.courrieu@univ-provence.fr

Etat civil: Né le 17 Mai 1954 à Sokodé (Togo), marié, 1 enfant, nationalité française.

Etudes Universitaires

1972: Diplôme de Bachelier de l'Enseignement du Second Degré

Etudes supérieures de psychologie à l'Université de Provence (1974-1983):

1974-1976: Diplôme d'Etudes Universitaires Générales

1977: License de Psychologie

1978: Maîtrise de Psychologie, directeur G. Noizet.

1980: DEA de Psychologie, directeur G. Noizet.

1983: Doctorat de troisième cycle en Psychologie, directeur C. Bastien. Thèse: "L'identification des mots au cours de la lecture", soutenue le 22/12/1983, mention "Très Bien" à l'unanimité du jury (C. Bastien, C. Bolusset, A. Lévy-Schoen, G. Noizet, & J. Pynte).

Activités professionnelles

1980-1983: Allocataire de Recherche DGRST, Laboratoire de Psychologie Cognitive (L.A. CNRS N° 182), Université de Provence.

1980-1981: Chargé de Cours à l'Université de Provence, UER de Psychologie (enseignement de la méthodologie et de l'informatique).

1981(novembre)-1983(mars): Chargé d'Etudes et Conseiller Scientifique, Centre National d'Etudes des Télécommunications (CNET-LAA-TSS-SEF), Rte de Tregastel, 22301 Lannion.

1984: Attaché de Recherche au CNRS, Centre de Recherche en Psychologie Cognitive (URA CNRS 182), Université de Provence.

1986: Chargé de Recherche de 2ème classe au CNRS, CREPCO (URA 182), Université de Provence.

1989: Chargé de Recherche de 1ère classe au CNRS, CREPCO (URA 182) devenu LPC (UMR 6146), Université de Provence.

Thèmes de recherche abordés

Perception visuelle des formes et des lettres, apprentissage perceptif, code orthographique, modèles de codage de données, modèles de codage d'images, modèles d'approximation de fonctions, méthodes de calcul des paramètres de modèles, méthodes de validation de modèles.

LISTE DES PUBLICATIONS

Articles dans des revues et ouvrages à comité de lecture

Courrieu, P. (1985). Des lettres sans position dans la perception des mots. *L'Année Psychologique, 85*, 9-25. doi : 10.3406/psy.1985.29064
(URL: http://www.persee.fr/web/revues/home/prescript/article/psy_0003-5033_1985_num_85_1_29064)

Courrieu, P. (1986). Analyse-t-on le mot de gauche à droite ? *Bulletin de Psychologie (N° spécial "Hommage à Georges Noizet"), T.XXXIX, N° 375*, 425-427.

Courrieu, P. (1986). Serial analysis in the perceptual discrimination of words. *Cahiers de Psychologie Cognitive, 6(3)*, 329-336.

Courrieu, P. (1988). Paradigmes temps réel interactifs. *In* J.P. Caverni, C. Bastien, P. Mendelsohn, & G. Tiberghien, *Psychologie Cognitive: Modèles et Méthodes*. Grenoble: P.U.G., pp.365-373.

Courrieu, P. (1988). Stratégies d'abbréviation de mots français. *L'Année Psychologique, 88*, 47-63. doi : 10.3406/psy.1988.29250
(URL: http://www.persee.fr/web/revues/home/prescript/article/psy_0003-5033_1988_num_88_1_29250)

Courrieu, P. (1993a). A convergent generator of neural networks. *Neural Networks, 6*, 835-844.

Courrieu, P. (1993b). A distributed search algorithm for global optimization on numerical spaces. *RAIRO: Recherche Opérationnelle / Operations Research, 27*, 281-292.

Courrieu, P. (1994a). Three algorithms for estimating the domain of validity of feedforward neural networks. *Neural Networks, 7*, 169-174.

Courrieu, P. (1994b). Connexionnisme et fonctions symboliques. *In* J.P. Caverni, C. George, & G. Politzer, *Raisonnements: Conjoncture et Prospective*. *Psychologie Française (numéro spécial), 39-2*, 231-236.

Courrieu, P. (1997). The Hyperbell algorithm for global optimization: a random walk using Cauchy densities. *Journal of Global Optimization, 10*, 37-55.

Courrieu, P. (2001). Two methods for encoding clusters. *Neural Networks, 14*, 175-183.

Courrieu, P. (2002). Straight monotonic embedding of data sets in Euclidean spaces. *Neural Networks, 15*, 1185-1196.

Courrieu, P. (2004). Solving time of least square systems in Sigma-Pi unit networks. *Neural Information Processing - Letters and Reviews, 4(3)*, 39-45.
(PDF: http://bsrc.kaist.ac.kr/nip-lr/V04N03/V04N03P2-39-45.pdf)

Courrieu, P. (2005a). Function approximation on non-Euclidean spaces. *Neural Networks, 18*, 91-102.

Courrieu, P. (2005b). Fast computation of Moore-Penrose inverse matrices. *Neural Information Processing - Letters and Reviews, 8(2)*, 25-29.
(PDF: http://bsrc.kaist.ac.kr/nip-lr/V08N02/V08N02P2-25-29.pdf)

Courrieu, P. (2006). Density codes and shape spaces. *Neural Networks, 19*, 429-445.

Courrieu, P. (2007). Fast density codes for image data. *Neural Information Processing - Letters and Reviews, 11(12)*, 247-255.
(PDF: http://bsrc.kaist.ac.kr/nip-lr/V11N12/V11N12P1-247-255.pdf)

Courrieu, P. (2009). Fast solving of Weighted Pairing Least-Squares systems. *Journal of Computational and Applied Mathematics, 231*, 39-48.

Courrieu, P., Brand-D'Abrescia, M., Peereman, R., Spieler, D., Rey, A. (2011). Validated intraclass correlation statistics to test item performance models. *Behavior Research Methods*. doi: 10.1007/s13428-010-0002-7

Courrieu, P., De Falco, S. (1989). Segmental vs. dynamic analysis of letter shape by preschool children. *CPC: European Bulletin of Cognitive Psychology, 9*, 189-198.

Courrieu, P., Dô, P. (1987). Perceptual analysis of words in Arabic. *In* J.K. O'Regan and A. Lévy-Schoen (Eds), *Eye Movements: From Physiology to Cognition*. Amsterdam: North Holland, pp. 451-458.

Courrieu, P., Farioli, F., Grainger, J (2004). Inverse discrimination time as a perceptual distance for alphabetic characters. *Visual Cognition, 11(7)*, 901-919.

Pynte, J., Courrieu, P., Frenck, C. (1989). Retrieval from verbal memory and motor programming during writing by hand. *In* P. Boscolo (Ed.), *Writing: Trends in European Research*. Padova: Upsel Editore, 205-212.

Pynte, J., Courrieu, P., Frenck, C. (1991). Evidence of repeated access to immediate verbal memory during handwriting. *European Journal of Psychology of Education, 6(2)*, 121-125.

Pynte, J., Courrieu, P., Kennedy, A., Murray, W.S. (1987). On the role of spatialisation in reading ambiguous sentences. *In* G. Lüer & U. Lass, *Fourth European Conference on Eye Movements. Volume 1: Proceedings*. Göttingen, C.J. Hogrefe, pp. 31-33.

Pynte, J., Kennedy, A., Murray, W.S., Courrieu, P. (1988). The effects of spatialisation on the processing of ambiguous pronominal reference. *In* G. Lüer, U. Lass, & J. Shallo-Hoffmann (Eds.), *Eye Movement Research: Physiological and Psychological Aspects*. Göttingen, C.J. Hogrefe, pp. 214-225.

Rey, A., & Courrieu, P. (2010). Accounting for item variance in large-scale databases. *Frontiers in Psychology 1:200*. doi:10.3389/fpsyg.2010.00200
(URL: http://www.frontiersin.org/language_sciences/10.3389/fpsyg.2010.00200/full)

Rey, A., Courrieu, P., Schmidt-Weigand, F., Jacobs, A.M. (2009). Item performance in visual word recognition. *Psychonomic Bulletin & Review, 16(3)*, 600-608

Ripoll, H., Baratgin, J., Laurent, E., Courrieu, P., & Ripoll, T. (2001). Mechanisms underlying the activation of knowledge basis in identification of basketball play configurations by expert and non-expert players. *In* A. Papaioannou, M. Goudas, & Y. Theodorakis (Eds.), *In the Dawn of the New Millennium: Proceedings of the 10th World Congress of Sport Psychology, Skiathos, Greece, 28th May - 2nd June, Vol. 2*, pp. 283-285. Thessaloniki: Christodoulidi Publications.

Archives et publications sans comité de lecture

Baratgin, J., Courrieu, P., Ripoll, T., Laurent, L., & Ripoll, H. (2002). *Similarity Judgements of Basketball Game Configurations by Experts and Novices: A Model and Some Experimental Tests*. Unpublished manuscript (2002): http://www.ergos-perf.com/Docs/Model_PSE.pdf, 54 p.

Courrieu, P., & Lequeux, M. (2009). *Anagram Effects in Visual Word Recognition*. Unpublished manuscript (1988): http://hal.archives-ouvertes.fr/hal-00429184/fr/, 40 p.

Courrieu, P., Ripoll, T., & Sabancioglu, F. (2009). *Affinely Invariant Features in Visual Perception of Letters and Words*. Unpublished manuscript (2002): http://hal.archives-ouvertes.fr/hal-00429562/fr/, 14 p.

Manuscrits actuellement soumis pour publication (Novembre 2010)

Courrieu, P. (submitted-1). *Quick Approximation of Bivariate Functions*.

Courrieu, P., & Rey, A. (submitted-2). *Missing Data Imputation and Corrected Statistics for Large-Scale Behavioral Databases*.

Communications

Bolusset, C., Devauchelle, P., Courrieu, P. (1982). Communication homme-terminal : aspects psychophysiologiques de la lecture sur écran. *18ème Congrès de la Société d'Ergonomie de Langue Française*. Paris, 13-15 Octobre.

Courrieu, P. (1980). La lecture des mots en contexte restreint. *Journées d'étude de l'Ecole Pratique des Hautes Etudes "Codage Phonétique et Codage Phonologique"*. Paris, Octobre.

Courrieu, P. (1982). Intégration perceptive des chaînes de caractères et accès au lexique. *Journées d'étude de l'Ecole Pratique des Hautes Etudes "Accès au Lexique"*. Paris, 30 Septembre-1er Octobre.

Courrieu, P. (1984). Analyse-t-on le mot de gauche à droite ? *Colloque "Hommage à Georges Noizet"*. Aix-en-Provence, 18-19 Octobre.

Courrieu, P. (1986). Paradigmes temps réel interactifs. *Colloque Société Française de Psychologie "Activités cognitives: modèles de processus et niveaux d'observation"*. Aix-en-Provence, 13-14 Mars.

Courrieu, P. (1989). Anagram effects in word recognition. *Workshop on Language Comprehension*. Aix-en-Provence, September 15-16.

Courrieu, P. (1991). A convergent generator of neural networks. *Second Workshop on Language Comprehension*. Aix-en-Provence, April 26-27.

Courrieu, P. (1993). Raisonnements et connexionnisme: réponse à Alain Grumbach. *Congrès de la Société Française de Psychologie*. Poitiers, 13-15 Mai.

Courrieu, P. (1998). Réalisme cognitif dans les modèles de lecture: une brève confrontation du modèle de Fukushima avec quelques données empiriques. *Journée Thématique "Lecture" du GRCE*. Paris, ENST, 26 Novembre.

Courrieu, P., Dô, P. (1985). Perceptual analysis of words in Arabic. *Third European Conference on Eye Movements*. Dourdan, September 24-27.

Pynte, J., Besson, M., Courrieu, P. (1991). Etude comportementale et électrophysiologique de la compréhension du langage. *Assises Régionales des Sciences de la Cognition "Cognisud"*. Marseille, 24-26 Janvier.

Pynte, J., Courrieu, P., Frenck, C. (1988). Psycholinguistic management in immediate memory and the motor programming of handwriting. *International Workshop on Writing*. Padova, December 1988.

Pynte, J., Courrieu, P., Kennedy, A., Murray, W.S. (1987). On the role of spatialisation in reading ambiguous sentences. *Fourth European Conference on Eye Movements*. Göttingen, September 21-24.

Rey, A.*,* Brand-d'Abrescia, M., Peereman, R., Spieler, D., & Courrieu, P. (2010). The nanopsycholinguistic approach: Item performance in disyllabic word naming. Oral communication presented at the *51st Annual Meeting of the Psychonomic Society*, St Louis, USA, November 18-21.

Ripoll H., Baratgin J., Laurent E., Courrieu P. & Ripoll T. (2001). Mechanisms underlying the activation of knowledge basis in identification of basketball play configurations by experts and non-experts players. *10th World Congress of Sport Psychology*. Skiathos Island (Greece), May 28-June 2.

AUTRES ACTIVITES

Participation à des contrats de recherche

Besson, M., Courrieu, P., Frenck-Mestre, C., Jacobs, A., Pynte, J. (1992). *Approche électrophysiologique et simulation de la compréhension du langage*. M.R.T.-Sciences de la Cognition 1992.

Ducrot, S., Lété, B., Nguyen, N., Pynte, J., Habib, M., Rey, V., Asmussen, C., Bastien, C., Bastien-Toniazzo, M., Courrieu, P., Colé, P., Rey, A. (2002-2004). *Le developpement des capacités perceptives visuo-attentionnelles au cours de l'acquisition de la lecture (normale et pathologique)*. Projet co-financé par Conseil Général des Bouches-du-Rhône, Conseil Régional PACA, CNRS-Université de Provence.

Gonzalez, M., Courrieu, P., Pélissier, A. (1988-1992). *Etudes d'une expertise humaine en détection sous-marine*. Convention DCAN-CNRS C 87 48 813 517, tranches I, II, III.

Pynte, J., Courrieu, P., Frenck, C., Vion, M., Cavé, C., Di Cristo, A., Hirst, D., Roméas, P., Besson, M., Jacobs, A., Nazir, T., Ide, N., Véronis, J., Harie, S., Habib, M., Colé, P., Hamon, J.-F., Léonard, F., Magnan, A., Mendoza, J.-L. J. (1992). *Les unités de traitement dans la perception de la parole et la lecture*. Réseau Cognisciences "Cognisud", N°18.

Ripoll, H., Baratgin, J., Laurent, E., Kehlhoffner, E., Cauzinille, E., Courrieu, P., Pélissier, A., Ripoll, T., Drogoul, A., Landau, S., Munoz, A., Zucker, J.-D., Bredeche, N. (1999-2002). *Les déterminants cognitifs de l'organisation spatiale du footballeur: application à l'homme et au robot*. A.C.I. Cognitique (thème 1: Cognition Spatiale), N° 90.

Touratier, C., Courrieu, P., Piolat, A., Pynte, J., Véronis, J. (1987). *Convergence des modèles linguistiques et psychologiques de l'orthographe dans la production écrite*. A.T.P. CNRS "Nouvelles Recherches sur le Langage", N° 1099.

Expertises d'articles soumis aux revues suivantes:

. L'Année Psychologique
. CPC: European Bulletin of Cognitive Psychology
. IEEE Transactions on Neural Networks
. Journal of Computational & Applied Mathematics
. Computers and Mathematics with Applications
. Applied Numerical Mathematics
. Bulletin of the Malaysian Mathematical Sciences Society
. Computing

Participation à l'organisation de colloques scientifiques

Workshop on "Language Perception and Comprehension: Multidisciplinary Approaches", Marseille, July 14-18th 1992. Organisé par M. Besson, P. Courrieu, C. Frenck-Mestre, A. Jacobs, & J. Pynte.

Enseignement

1984-1988: enseignement de l'informatique (programmation Basic et Pascal) en second cycle de psychologie (37h1/2 / an), interventions en DESS d'Ergonomie Cognitive (3h / an), Université de Provence.

1991-2006: enseignements sur les réseaux de neurones artificiels et la reconnaissance des formes en 2ème 3ème cycles de psychologie (6h / an), Université de Provence.

Encadrement de mémoires de recherche

Aicart-De Falco, S. (1987). *Apprentissage et reconnaissance de caractères minuscules scripts chez des enfants d'école maternelle*. Mémoire de Maîtrise de Psychologie, Université de Provence.

Lequeux, M. (1986). *Des roles des lettres sans position et des anagrammes dans la perception des mots*. Mémoire de Maîtrise de Psychologie, Université de Provence (codir. C. Bastien).

Lequeux, M. (1988). *Effet du nombre d'anagrammes lexicales en fonction de leurs fréquences d'usage dans la perception des mots*. Mémoire de D.E.A. de Psychologie, Université de Provence (codir. C. Bastien).

Sabancioglu, F. (2002). *A la recherche des invariants dans la perception visuelle des lettres et des mots: les transformations par changement d'échelle et par double symétrie*. Mémoire de Maîtrise de Psychologie, Université de Provence (codir. T. Ripoll).

16/11/84

MINISTÈRE DE L'ÉDUCATION NATIONALE

FICHIER CENTRAL DES THÈSES

UNIVERSITÉ PARIS X - NANTERRE
200, avenue de la République
92001 NANTERRE
Tél. 725.92.34

NUMÉRO D'ENREGISTREMENT : 8118939I

(Ce numéro doit être rappelé dans toute correspondance avec le Fichier Central des Thèses).

MR. COURRIEU PIERRE

1088 AVENUE RENOIR

83500 LA SEYNE SUR MER

ATTESTATION D'ENREGISTREMENT AU FICHIER CENTRAL DES THÈSES

PREMIÈRE INSCRIPTION PRISE EN DECEMBRE 1980

POUR LA PRÉPARATION D'UN DOCTORAT DE 3EME CYCLE

A L'UNIVERSITÉ (ou autre établissement agréé) UNIV.AIX MARS1

SOUS LA DIRECTION DE MR BASTIEN CLAUDE

DISCIPLINE PSYCHOLOGIE

THESE SOUTENUE EN DECEMBRE 1983

TITRE DE LA THÈSE

L'IDENTIFICATION DES MOTS AU COURS DE LA LECTURE.

CETTE ATTESTATION DOIT ÊTRE CONSERVÉE

AUCUN DUPLICATA NE SERA DÉLIVRÉ

00000013

14

T 550 3601

IN 4 061144 C 83 D

SECONDE PARTIE
Dossier de travaux

Une grande partie des travaux que j'ai réalisés au début de ma carrière, dans la suite de ma thèse de troisième cycle, consistait en des études expérimentales du codage orthographique des mots chez le lecteur adulte. Je n'ai pas complètement abandonné cette thématique, mais si je dois y revenir, ce sera dans une perspective assez différente, plus étroitement liée à la modélisation numérique. J'ai donc choisi de ne pas présenter ici ces travaux anciens, et le lecteur intéressé pourra se reporter aux références apparaissant sur ce sujet dans la liste de mes publications. En particulier, trois de ces références comportent des liens vers des documents librement accessibles (Courrieu, 1985; Courrieu, 1988; Courrieu & Lequeux, 2009). La dernière de ces références est en fait un manuscrit non publié que j'ai récemment mis en ligne, et dont la rédaction initiale remonte à 1988 (révisée en 2004). L'échantillon de travaux présenté dans ce qui suit couvre essentiellement la période 2001-2010, à l'exception de deux références antérieures à cette période qui m'ont paru avoir quand même leur place ici.

II.A Perception des lettres

Les lettres de l'alphabet sont des formes visuelles à part entière, assez simples à première vue, mais aussi très variables dans leurs réalisations. Les lettres peuvent apparaître au sein de formes plus complexes telles que des mots, et les lecteurs adultes sont experts dans leur reconnaissance. Sur quoi se fonde cette habileté extrême à reconnaître les lettres, et comment est-elle acquise? Ayant constaté le relatif dédain dans lequel l'Intelligence Artificielle tenait la reconnaissance de formes simples comme les lettres (ce qui contrastait singulièrement avec la réelle complexité de cette performance), Douglas Hofstadter (1995) n'a pas hésité à déclarer que "the toughest challenge facing AI workers is to answer the question: What are the letters 'A' and 'I'?".

D'assez nombreux modèles de reconnaissance des caractères ont été proposés dans le domaine de la Reconnaissance des Formes (Pattern Recognition), mais finalement, assez peu de principes ont été retenus par la psychologie de la perception. Reconnaissance globale et "template matching" ne semblent plus avoir beaucoup d'adeptes, bien que les outils de modélisation utiles dans ces approches aient fortement progressé ces dernières années (j'y

reviendrai dans la section II.C). L'idée qui résiste le mieux aux critiques et aux mises à l'épreuve expérimentales est celle d'une organisation hiérarchique de la perception visuelle, partant de la détection de traits (segments) simples qui sont ensuite combinés en des éléments plus complexes au niveau supérieur, et ainsi de suite jusqu'à atteindre le niveau d'éléments reconnaissables comme des caractères, des combinaisons de lettres, ou des mots. Cette théorie hiérarchique est directement inspirée des observations neurophysiologiques (Hubel & Wiesel, 1970), et est assez largement adoptée en psychologie cognitive (Dehaene, 2007).

Une autre idée a été proposée par Courrieu et De Falco (1989, article ci-joint), se fondant sur le double constat suivant. Il est, d'une part, relativement facile de reconnaître automatiquement des caractères lorsque ceux-ci sont représentés sous la forme de courbes paramétriques (position (x, y) fonction du temps (t)), c'est-à-dire sous la forme d'une représentation abstraite du geste d'écriture. C'est ce qu'on appelle la reconnaissance "online" de caractères (Connell & Jain, 2001). D'autre part, il est aisé pour un lecteur adulte de reconstituer a posteriori une approximation du geste d'écriture à partir d'une trace écrite, laquelle est une image en deux dimensions (niveau de gris (g) fonction de la position (x, y)). Il existe de plus des méthodes de suivi automatique de tracé dans une image (Baruch, 1988; Chouinard & Plamondon, 1992), ce qui confirme la faisabilité de la chose. La question était donc posée: est-ce qu'apprendre à tracer des lettres facilite leur reconnaissance en lecture? Pour tenter de répondre à cette question, Courrieu et De Falco (1989) ont appris à des enfants pré-scolaires à reconnaître des lettres avec différentes méthodes d'apprentissage, permettant de privilégier soit une reconnaissance globale, soit une analyse des lettres en segments simples, soit une représentation dynamique du geste d'écriture. Les résultats ont fait clairement apparaître l'efficacité de l'analyse des lettres en segments simples, mais il n'y avait pas d'effet détectable de l'apprentissage du geste d'écriture sur la reconnaissance. Cependant, d'autres auteurs ont récemment repris cette idée, et ont observé que l'apprentissage du geste d'écriture facilite la reconnaissance des caractères chez des enfants un peu plus âgés (Longcamp, Zerbato-Poudou, & Velay, 2005), ainsi que chez des adultes (Longcamp, Boucard, Gilhodes, Anton, Roth, Nazarian, & Velay, 2008). L'idée n'était donc pas si mauvaise, et il semble bien que la construction de représentations à partir du geste d'écriture puisse contribuer à la reconnaissance des lettres, mais seulement à partir d'un certain niveau de maturation du système cognitif. Certains auteurs ont baptisé "signature proprioceptive" le pattern d'activation motrice caractéristique de chaque symbole graphique reconnaissable (Roll, Albert, Ribot-Ciscar, & Bergenheim, 2004; Vinter & Chartrel, 2008), mais il s'agit là d'une forme plus "incarnée" (donc moins abstraite) de l'hypothèse initiale de représentations

en courbes paramétriques. Par ailleurs, comme les représentations en segments simples sont également efficaces (Courrieu & De Falco, 1989), et même plus précocement, on peut supposer que la reconnaissance des lettres s'appuie en fait sur différentes ressources, et qu'a priori, rien de ce qui peut être utile à la perception n'est à exclure.

Une autre façon d'aborder la question des représentations à l'oeuvre dans la perception des lettres consiste à analyser des données de similitude perceptive des différentes lettres, de façon à en extraire un ensemble de facteurs déterminants. C'est ce qu'ont fait Courrieu, Farioli, & Grainger (2004, article ci-joint) en mesurant et analysant des temps de discrimination perceptive des 26 lettres de l'alphabet latin, plus le caractère d'espacement. La méthodologie employée consistait à appliquer à la matrice des temps de discrimination inverses une technique de "plongement monotone euclidien" (Courrieu, 2002, voir section II.B), qui est une technique particulière de "multidimensional scaling" permettant d'attribuer à chaque caractère une position dans un espace euclidien, puis à compléter par une analyse factorielle. Nous avons ainsi obtenu un ensemble de 25 facteurs, d'importances inégales, interprétables comme des descripteurs perceptifs ("features"). Certains de ces descripteurs correspondaient à la présence d'une forme de lettre entière, comme par exemple "n" dans "n" ou "h", "i" dans "i" ou "j", "v" dans "v" ou "y". D'autres descripteurs correspondaient à des patterns particuliers tels que "4 coins et une diagonale d'un carré" comme dans "x" et "z" (mais aussi dans "%"), ou à des segments simples comme "l'arc de cercle supérieur gauche" commun à "c" et "r". Ces descripteurs sont compatibles avec la théorie hiérarchique évoquée plus haut. Cependant, d'autres descripteurs, plus abstraits, sont difficilement interprétables dans ce cadre. Certains descripteurs correspondaient à une caractéristique assez générale comme "petite forme curvilinéaire" dans "a, c, e, o, s", ou à une caractéristique plus spécifique comme "forme sigmoïde" dans "s" et "z". D'autres descripteurs correspondaient à une certaine combinaison de segments simples, mais indépendamment de leur disposition comme "petit cercle et grande barre verticale" dans "b, d, p, q", ce qui suggère des invariances par symétrie ou rotation. Les invariances par rotation ont par ailleurs été confirmées dans une expérience de priming masqué de lettres (Courrieu, Ripoll, & Sabancioglu, 2009). On voit donc que la théorie hiérarchique, dans sa formulation courante, ne peut sans doute pas rendre compte à elle seule de l'ensemble des descripteurs utilisés par la perception.

Une étude ultérieure de Pelli, Burns, Farell, & Moore-Page (2006) estime que le nombre de descripteurs visuels dont la détection permet d'identifier un caractère serait de l'ordre de 7 (±2), mais aucun descripteur n'est spécifié, et les auteurs semblent avoir totalement ignoré le travail de Courrieu et al. (2004). Une autre étude de Fiset, Blais, Ethier-

Majcher, Arguin, Bub, & Gosselin (2008) propose un ensemble de 10 descripteurs visuels obtenus grâce une technique dite "de bulles", mais l'étude de Courrieu et al. (2004) ne semble pas non plus avoir attiré l'attention de ces auteurs. La cause de l'étrange "transparence" de cette publication est peut-être à rechercher dans le fait que l'article a bizarrement disparu de certaines bases documentaires électroniques très utilisées, et que de multiples courriers adressés à des responsables potentiels, pour tenter de corriger cette anomalie, sont restés sans réponse et sans effet. Une autre explication possible est qu'avec les mots clés "alphabetic character perception", Google Scholar retourne bien la référence de l'article dans les premiers rangs, mais avec les mots clés "letter perception", il n'en est rien! Comme il n'est pas dans mes compétences d'enseigner la notion de synonymie à Google Scholar, je veillerai une prochaine fois à choisir plus stratégiquement le titre de mes articles...

Références

Baruch, O. (1988) Line thinning by line following. *Pattern Recognition Letters, 8*, 271-276.

Chouinard, C., & Plamondon, R. (1992). Thinning and segmenting handwritten characters by line following. *Machine Vision and Applications, 5*, 185-197.

Connell, S.D., & Jain, A.K. (2001). Template-based online character recognition. *Pattern Recognition, 34*, 1-14.

Courrieu, P. (2002). Straight monotonic embedding of data sets in Euclidean spaces. *Neural Networks, 15*, 1185-1196.

Courrieu, P., De Falco, S. (1989). Segmental vs. dynamic analysis of letter shape by preschool children. *CPC: European Bulletin of Cognitive Psychology, 9*, 189-198.

Courrieu, P., Farioli, F., Grainger, J (2004). Inverse discrimination time as a perceptual distance for alphabetic characters. *Visual Cognition, 11(7)*, 901-919.

Courrieu, P., Ripoll, T., & Sabancioglu, F. (2009). *Affinely Invariant Features in Visual Perception of Letters and Words*. Unpublished manuscript (2002): http://hal.archives-ouvertes.fr/hal-00429562/fr/, 14 p.

Dehaene, S. (2007). *Les Neurones de la Lecture*. Paris, Odile Jacob, 478 p.

Fiset, D., Blais, C., Ethier-Majcher, C., Arguin, M., Bub, D., & Gosselin, F. (2008). Features for identification of uppercase and lowercase letters. *Psychological Science, 19(11)*, 1160-1167.

Hubel, D.H., & Wiesel, T.N. (1970). Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology, 195*, 215-243.

Hofstadter, D.R. (1995). On seeing A's and seeing As. *Stanford Humanities Review: Constructions of the Mind, 4(2)*, 109-121.

Longcamp, M., Boucard, C., Gilhodes, J-C., Anton, J-L., Roth, M., Nazarian, B., & Velay, J-L. (2008). Learning through Hand- or Typewriting Influences Visual Recognition of New Graphic Shapes: Behavioral and Functional Imaging Evidence. *Journal of Cognitive Neuroscience 20:5*, pp. 802–815.

Longcamp, M., Zerbato-Poudou, M-T., & Velay, J-L. (2005). The influence of writing practice on letter recognition in preschool children: A comparison between handwriting and typing. *Acta Psychologica, 119*, 67–79.

Pelli, D.G., Burns, C.W., Farell, B., & Moore-Page, D.C. (2006). Feature detection and letter identification. *Vision Research, 46*, 4646–4674.

Roll, J-P., Albert, F., Ribot-Ciscar, E., & Bergenheim, M. (2004). ‘‘Proprioceptive signature” of cursive writing in humans: a multi-population coding. *Experimental Brain Research, 157*, 359–388.

Vinter, A., & Chartrel, E., (2008). Visual and proprioceptive recognition of cursive letters in young children. *Acta Psychologica, 129*, 147–156.

C.P.C. : Cahiers de Psychologie Cognitive
European Bulletin of Cognitive Psychology
1989, Vol. 9, n° 2, 189-198.

SEGMENTAL VS. DYNAMIC ANALYSIS OF LETTER SHAPE BY PRESCHOOL CHILDREN

Pierre Courrieu and Suzy De Falco

Centre de Recherche en Psychologie Cognitive (UA CNRS 182)
Université de Provence, 29 avenue Robert Schuman
13621 Aix-en-Provence Cedex, France
e-mail: CREPCO@FRMOP11.BITNET

Abstract

Using a technique appropriate to preschool children, a confusion matrix of Roman letters written in printed script was generated for the purpose of determining what classes of letters are likely to be confused by preschoolers. An experiment in learning to visually perceive the "difficult" letters was then conducted using a different sampling of children from the same population. All participants took a letter-discrimination test before and after the learning period. In some cases, the learning phase involved analysis of letters into simple segments, and in others it involved dynamic tracing of each letter in imitation of the writing gesture. The results show that analysis into simple segments has a significant beneficial effect on learning to perceive letters, while dynamic representation is ineffective. These findings support the hypothesis that perception processes are analytic, and do not rely on the parametric representation of the written trace.

Key words: Letter recognition, learning to read.
Mots clés : Reconnaissance des lettres, apprentissage de la lecture.

The problem

It is well known today that the perception of written words by adult readers is done by extraction of all or part of the letters in the words being read (Johnson & McClelland, 1980; McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982; Paap, Newsome, McDonald, & Schvaneveldt, 1982). The above authors also unanimously agree that letters themselves are detected by the integration of simpler units, such as line segments and curves, the so-called "primitives" that are extracted from the stimulus image (see also Rumelhart, McClelland, and the PDP Research Group, 1986). Although this hypothesis seems likely in light some of the well-known findings in neurophysiology (Hubel & Wiesel, 1963, 1968), no decisive experimental proof exists as to its adequacy in accounting for the perceptual processes involved in reading (see however Oden, 1979, 1984). Another idea is suggested by research in the field of automatic writing recognition, where two main types of models have been proposed: the so-called "on-line" systems (Berthod & Ahyan, 1980) where the signal is detected on-line from a digitizing tablet and data is created in the form of points on a plane that are strictly ordered in time, and the so-called "off-line" systems (Srihari & Bozinovic, 1987) where the signal is simply an image, i.e. a distribution of points on a plane that vary in luminosity and are not ordered in any particular way. When "on-line" data is available, we can parametrically represent the curve defined as a letter is being written (the writer's gesture), making for relatively easy processing and efficient recognition. "Off-line" data is much more difficult to process, however, and no satisfactory methods are yet available, even though the "off-line" situation is the one that *a priori* corresponds to the ordinary reading situation of human beings. The following observation might nevertheless be made: human beings can usually infer a good approximation of a dynamic process solely from a (static) curve, even when the object represented is unknown. For example, it is very easy to take a pen and trace over the strokes made by a writer, or even by a "scribbler", unless the trace is unreasonably complex. The question naturally raised then is: Could this obvious ability to produce the "on-line" from an "off-line" image be used by readers to transform the two-dimensional images of letter shapes into a parametric representation that would be more "efficiently" recognized? Note that this hypothesis does not imply equivalency of perceptual representations and motor representations. It is simply based on the assumption that there exists an amodal representation that is abstract enough to serve as a common reference for various different modal representations.

One way to approach this problem would be to experimentally vary the type of representation readers store of the objects to be recognized (letters), and then to test the effectiveness of the induced representations on recognition

performance. In the case of letters, such experimental variation can only be done on novice populations such as illiterates or preschoolers. The latter was chosen here because of the potential pedagogical value of such a study, in addition to the fact that the question of whether learning to write script is an aid in learning to read has already in fact been raised by teachers.

Preschoolers were first tested for their ability to perceptually discriminate Roman letters written in printed script so as to determine which letters they needed to learn, i.e. which letters were the most likely to be confused. The following pre-experiment was conducted for that purpose.

Pre-experiment

1. Subjects: 52 children between the ages of 3;1 and 6;2 participated in the experiment. All had normal eyesight and were enrolled in a preschool.

2. Material: Each child was given a series of 26 worksheets containing 78 characters from the Roman alphabet carefully written in lowercase printed script, and one reference character enlarged to twice the normal size, located in a priming area at the top of the worksheet. There was one worksheet for each reference letter. The 78 characters on the lower part of the worksheet consisted of three occurrences of the complete Roman alphabet in random order. The reference letter thus occurred three additional times on each worksheet.

3. Task: The children were asked to search among the "little letters" to find the ones identical to the "big letter" (i.e. the reference letter) and to circle them with a pencil. Erasing and correcting was allowed.

4. Procedure: The data was gathered across four 20-minute sessions spread over one week. The worksheet presentation order was varied using the latin squares "rank-neighborhood" method with two subjects for each of the 26 orders in the latin square thus defined.

5. Results: The results are given in table 1 in a form analogous to a letter confusion matrix, with a maximum total per cell of 156 (3 occurrences x 52 subjects).

If we exclude the rarely-made errors, we can see that the confusion matrix is quite empty and basically symmetrical, which makes it easy to interpret and eliminates the necessity of complex matrix computations. The main confusion classes are ((b,d), (p,q)), (f,t), and (n,u), where the inner parentheses delimit subclasses with a higher probability of confusion than those within the outer parentheses. Notice that the high confusion frequencies systematically pertain to letters that differ only by affine transformations, such as symmetries. This

TABLE 1. Confusion matrix of the 26 lowercase letters in the Roman alphabet written in printed script for preschool children (maximum per cell: 156). The row entries are the reference characters and the columns are the responses.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	123	3	3	5	0	0	0	0	0	0	0	0	0	0	5	1	2	0	0	0	0	0	0	0	0	0
b	0	99	0	88	0	0	4	0	0	0	0	1	0	0	0	32	21	0	0	0	0	0	0	0	2	0
c	1	0	133	0	3	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
d	0	96	0	121	0	0	0	0	0	0	0	0	0	0	0	24	22	0	0	0	0	0	0	0	0	0
e	0	1	5	0	132	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	0	0	0	0	0	129	0	0	0	3	0	0	0	0	0	0	0	0	0	47	0	0	0	0	0	0
g	0	2	0	3	0	0	128	0	0	2	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0
h	0	0	0	1	0	0	0	117	0	0	0	0	0	8	0	0	0	0	0	0	4	0	0	0	0	0
i	0	0	0	0	0	0	0	0	135	6	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	6	131	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	2	0	0	8	0	0
l	0	0	0	0	0	0	0	1	7	4	0	128	0	0	0	0	0	1	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	138	0	0	0	0	0	0	0	0	0	7	0	0	0
n	1	0	0	0	0	0	0	7	0	0	0	0	0	112	0	0	0	0	0	0	22	0	0	0	0	0
o	4	0	13	0	0	0	0	0	0	0	0	0	0	0	126	0	0	0	0	0	0	0	0	0	0	0
p	0	22	0	19	0	0	0	0	0	0	0	0	0	0	0	102	71	0	0	0	0	0	0	0	0	0
q	1	20	0	21	0	0	0	1	0	0	0	0	0	0	0	88	99	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	3	0	0	0	1	0	0	0	0	0	0	0	120	0	3	0	0	0	0	0	0
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	0	0	0	0	0	0	5
t	0	0	0	0	0	40	0	0	0	2	0	0	0	0	0	0	0	0	0	122	0	0	0	0	0	0
u	1	0	0	0	0	0	0	3	0	0	0	0	2	24	0	0	0	2	0	0	114	0	0	0	0	0
v	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	2	125	0	0	0	0
w	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	2	0	133	0	0	0
x	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	141	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	143	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	5	0	0	0	0	3	0	129

TABLEAU 1. Matrice de confusion des 26 lettres minuscules script de l'alphabet latin chez des enfants d'âge pré-scolaire (maximum par case : 156). Les entrées lignes correspondent aux étalons et les entrées colonnes aux réponses.

suggests that children simply apply the same rules to letters as they do to ordinary objects in their environment, which can be viewed at different angles and do not lose their identity by affine transformations. There are two facts here that do not fit into this interpretation, however. First, letters such as (b,d) and (p,q) that are symmetrical with respect to the vertical axis were confused much more often than horizontally symmetrical letters like (b,p) and (d,q). Second, if we look at the confusions generally made by adult subjects (i.e. subjects who know the alphabetic code), we find a similar phenomenon. Moreover, except for the broader class (b,d,p,q), which no doubt can be interpreted as stated above, all the main confusion classes for these children are subclasses of the confusions made by adults. It suffices to compare these results to those obtained by Bouma (1971), who defined seven main confusion classes: (a,s,z,x), (e,o,c), (n,m,u), (r,v,w), (d,h,k,b), (t,i,l,f), (g,p,j,y,q). It thus seems likely that perceptual determinants other than the simple lack of knowledge of the alphabetic code are factors of letter confusion in both children and adults.

Experiment

1. Subjects: 52 children between the ages of 3;4 and 6;3 participated in the experiment. All had normal eyesight and were enrolled in a preschool.

2. Experimental setup: The experiment consisted of seven sessions held over a two-week period. The first and last sessions were devoted to a perceptual discrimination test of the eight printed script letters used in the experiment (b,d,p,q,f,t,n,u). The second session was used to familiarize the children with the training they would be given in sessions 3, 4, 5, and 6. The letters used as examples in session 2 (s,x) were not in the test letter set. Two letters were taught per session. The pairs, chosen so as not to oppose any two letters likely to be confused, were as follows: (d,u) for session 3, (b,t) for session 4, (p,n) for session 5, and (q,f) for session 6.

3. Perceptual letter-discrimination test: The pre-test and post-test were similar, and consisted of presenting eight worksheets (in a variable, random order). Each worksheet had one of the reference letters printed in double size in the middle of the priming area at the top of the worksheet. On the lower part of the worksheet, 5 occurrences of each of the test letters (40 characters in all) were shown in random order (except for the fact that the same letter never occurred twice in a row). The subject's task was to find and circle the "little letters" like the "big letter" at the top of the worksheet (erasing was allowed). Each session took twenty minutes.

The index used to assess a given subject's performance on a given worksheet (reference letter) was the ratio of the number of correctly circled characters to the total number of characters circled per worksheet. This index, which varied between 0 and 1, seemed to correctly indicate how well subjects could perceptually discriminate letters while eliminating certain strategy effects, such as the tendency of some children to circle many letters and others to circle only a few. The Signal Detection Theory index, d', which has analogous characteristics, could not be used since the distribution of "noise" was absolutely not Gaussian due to the fact that each reference letter had distractors that were systematically strong or weak.

4. Letter learning conditions:

- *Non-segmental, non-dynamic learning.* Each child was given one worksheet per letter with a priming area at the top containing a model of the letter, a concrete word starting with that letter, and a drawing illustrating the word. The child also had four paper cutouts of the letter. Below the priming area, the illustration word was written four times with its first letter replaced by a small arrow. The child's task was to correctly position the cutout letters over the arrows. The

experimenter only allowed the child to stick a letter in place if it was correctly positioned. After three unsuccessful trials by the child, the experimenter put the letter in the correct position. In this condition the children were thus led to manipulate each letter globally.

- *Segmental, non-dynamic learning.* The worksheet given to the children in this condition was similar to the above, with the following exceptions: in addition to the normal model, the letter was also shown broken down into three simple segments. The segments were the straight and curved portions of the letter, shown separated and in any order. The child was given two sets of these segments cut out of paper. Below the priming area on the worksheet, a sample of the illustration word was written four times with the first letter incomplete. In the first three samples, one of the segments was missing (a different one each time); in the last sample, all three segments were missing, and the first letter was replaced by two dotted lines indicating the upper and lower limits of the body of the letter. In each case, the child was to correctly complete the word using the cutout segments. The experimenter intervened at the same point as above.

- *Non-segmental, dynamic learning.* In this condition, instead of showing the letter broken down into segments, the worksheet priming area contained a very particular graphic representation of the letter. The letter was depicted by two solid lines filled in with dots whose density was inversely proportional to the distance (curvilinear abcissa) from the origin of the trace made by writing the letter in the ordinary fashion. At the points where the trace crossed itself, the density of the portion farthest away from the origin dominated. This seemed to be the best way to indicate the dynamics of the writing gesture. The children were informed of this rule during session 2 (devoted to familiarization). The lower part of the worksheet contained four samples of the illustration word. In the first three, the first letter was represented dynamically, and in the last, dotted lines replaced the missing first letter as above. The child's task was to learn to draw the letter by placing the pen on the darkest part to start and then moving it gradually towards the lightest part on the first three samples. Then he or she was supposed to draw the letter without a graphic guide in the appropriate place on the fourth sample. After three unsuccessful attempts, the child was given the necessary help.

- *Dynamic, segmental learning.* This learning condition is the combination of the preceding two conditions. The priming area contained the standard items (a normal letter, a word, and a drawing), plus a dynamic representation of the letter and a segmental representation of the letter. The segmentation was done in the same way as in the second condition, but the segments were ordered in their natural order of occurrence in the letter. In addition, the segmented

version had no overlapping portions since they were shown separately. The lower part of the worksheet contained four samples of the illustration word. In the first three samples, the first letter was represented dynamically with a new segment missing each time; in the fourth example, the first letter was missing altogether and its location was marked by two dotted lines as above. Again, the child's task was to learn to draw the letter by following the graphic guides, filling in the missing segments along the way. He or she was then to draw the letter on the fourth sample without a graphic guide, receiving the necessary assistance after three consecutive failures.

5. Data analysis: The independent variable is the perceptual discrimination index defined above for the pre-test and post-test. The data were processed in an analysis of variance with the following design: S13 <G2 * D2> * R8 * T2 where the factors are defined as follows:
S: Subject (random).
G: (Segmental learning, non-segmental learning).
D: (Dynamic learning, non-dynamic learning).
R: Reference letter (b,d,p,q,u,n,f,t).
T: (Pre-test, post-test).

The four letter-learning conditions described above were obtained by crossing factors G and D. Note that the four groups of children used were balanced as well as possible as to age, sex, and place of schooling. However, the effect of these variables is simply incorporated into the random variance, since this study was not aimed at analyzing developmental or differential variables. R is a secondary factor, but it cannot be considered as random since its modalities were chosen systematically.

6. Predictions: Our predictions can be easily deduced from the problem statement.
- *P1:* If letter recognition is based on analysis into segments, then the letter discrimination index will increase to a greater extent between the pre-test and the post-test when the training is segmental than when it is not (G-by-T interaction).
- *P2:* If letter recognition is based on a parametric representation of the writing gesture, then the letter discrimination index will increase more between the pre-test and the post-test when the training is dynamic than when it is not (D-by-T interaction).

7. Results: Table 2 gives the mean values of the letter discrimination index for the main experimental conditions, as well as its mean increase between the pre- and post-tests.

TABLE 2. Mean values of the letter discrimination index on the pre- and post-tests, and the mean increases between them for each letter-teaching method proposed to the children.

	Non-segmental			Segmental					
	Pre.	Post.	Incr.	Pre.	Post.	Incr.	Pre.	Post.	Incr.
Non-Dynamic	.612	.728	.116	.603	.771	.168	.607	.750	.143
Dynamic	.668	.773	.105	.550	.786	.236	.609	.780	.171
	.640	.751	.111	.577	.778	.201	.608	.765	.157

TABLEAU 2. Valeurs moyennes de l'indice de discrimination des lettres au pré-test et au post-test, et progressions moyennes de cet indice suivant le type d'apprentissage perceptif des lettres auquel a été soumis l'enfant.

The analysis of variance showed that there is a significant overall increase in the letter discrimination index between the pre- and post-tests ($F(1,48) = 81.31$, $p<.001$), and thus that as a whole, the children did benefit from the training. There is a significant interaction between the test and segmental/non-segmental factors ($F(1,48) = 6.87$, $p<.01$) indicating more progress in the segmental learning conditions; this confirms prediction P1. On the other hand, the interaction between the test and dynamic/non-dynamic factors is not significant ($F<1$), nor is the second-order interaction between test, segmental/non-segmental learning, and dynamic/non-dynamic learning ($F(1,48) = 1.32$, NS). Thus, prediction P2 did not prove to be true, and the introduction of dynamic gesture representation does not seem to systematically favor the perceptual learning of letters by children. Finally, the main factors do not interact significantly with the reference-letter factor.

Discussion

Our results clearly show that breaking letters down into segments is a valid approach, since a global representation is obviously less efficient than analytic representation. This argues in favor of models that define perceptual analysis as a hierarchical order involving infra-categorical levels, i.e. the letter may not be the smallest perceptual unit extracted from the signal, but rather a composite unit.

On the other hand, the idea that letter traces are represented parametrically was not confirmed here; the child's knowledge of the process by which letters are generated (the writing gesture) does not seem to systematically influence his/her perceptual performance. Of course, hasty generalizations should be

avoided, and it still may be true that adults develop representations and processing procedures that are better adapted to handling very irregular signals, such as adult cursive script. These representations and procedures, however, should be considered as a sort of expertise process that develops as experience with increasingly difficult signals is acquired. If this is the case, children may ordinarily not be able to read natural adult handwriting before having several years of reading experience. In the meantime, the basic representations involved in elementary learning do in fact appear to be analytical and not parametric.

RESUME

On a déterminé une matrice de confusion des lettres latines script à l'aide d'une technique appropriée chez des enfants d'âge pré-scolaire de façon à repérer les classes de lettres à forte probabilité de confusion. Puis on a réalisé une expérience d'apprentissage perceptif des lettres "difficiles" sur un autre échantillon d'enfants de la même population, les enfants étant soumis à un pré-test, puis à un post-test de discrimination des lettres, respectivement avant et après l'apprentissage. L'apprentissage pouvait introduire ou non une analyse des lettres en segments simples, et il pouvait d'autre part introduire ou non une représentation dynamique du tracé de chaque lettre correspondant au geste d'écriture. Les résultats montrent que l'analyse en segments simples a un effet significativement bénéfique sur l'apprentissage perceptif, alors que l'introduction de représentations dynamiques reste inopérante. Ceci milite en faveur de l'hypothèse de processus perceptifs analytiques n'utilisant pas de représentation paramétrique du tracé.

REFERENCES

Berthod, M., & Ahyan, S. (1980). On line cursive script recognition: A structural approach with learning. In *Proceedings Fifth International Conference on Pattern Recognition* (pp. 723-725). Miami Beach, FL.

Bouma, H. (1971). Visual recognition of isolated lower-case letters. *Vision Research, 11,* 459-474.

Hubel, D.H., & Wiesel, T.N. (1963). Shape and arrangement of columns in cat's striate cortex. *Journal of Physiology, 165,* 559-568.

Hubel, D.H., & Wiesel, T.N. (1968). Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology, 195,* 215-243.

Johnston, J.C., & McClelland, J.L. (1980). Experimental tests of a hierarchical model of word identification. *Journal of Verbal Learning and Verbal Behavior, 19,* 503-524.

McClelland, J.L., & Rumelhart, D.E. (1981). An interactive activation model of context effects in letter perception: Part 1. An account of basic findings. *Psychological Review, 88* (5), 375-407.

Oden, G.C. (1979). A fuzzy logical model of letter identification. *Journal of Experimental Psychology: Human Perception and Performance, 5* (2), 336-352.

Oden, G.C. (1984). Dependence, independence, and emergence of word features. *Journal of Experimental Psychology: Human Perception and Performance, 10* (3), 394-405.

Paap, K.R., Newsome, S.L., McDonald, J.E., & Schvaneveldt, R.W. (1982). An activation-verification model of letter and word recognition: the Word-Superiority-Effect. *Psychological Review, 89* (5), 573-594.

Rumelhart, D.E., & McClelland, J.L. (1982). An interactive activation model of context effects in letter perception: Part 2. The contextual enhancement effect and some tests and extensions of the model. *Psychological Review, 89* (1), 60-94.

Rumelhart, D.E., McClelland, J.L., & The PDP Research Group (1986). *Parallel distributed processing: Explorations in the microstructure of cognition.* Cambridge: The MIT Press (2 T.).

Srihari, S.N., & Bozinovic, R.M. (1987). A multi-level perception approach to reading cursive script. *Artificial intelligence, 33,* 217-255.

Received September 8, 1988
Accepted March 17, 1989

VISUAL COGNITION, 2004, *11* (7), 901–919

Inverse discrimination time as a perceptual distance for alphabetic characters

Pierre Courrieu, Fernand Farioli, and Jonathan Grainger

CNRS & University of Provence, Aix-en-Provence, France

Reaction times to discriminate lower-case letters were collected in an experiment. The inverse discrimination times were used to build metrics on the space of letters. These metrics were found to be significantly correlated with various well-known letter confusability measures, and a meaningful dimensional analysis of the alphabet was performed. This methodology is mathematically well founded, it requires fewer data than common methods, and it appears to be highly sensitive to visual similarity between letters, which allows visual letter features to be effectively analysed.

A number of prior studies have attempted to analyse the perceptual relations between lower-case letters of the Roman alphabet by collecting subjective similarity data (Dunn-Rankin, 1968; Dunn-Rankin, Leton, & Shelton, 1968; Kuennapas & Janson, 1969) or confusion data (Bouma, 1971; Courrieu & de Falco, 1989). Understanding these relations is important from a theoretical point of view, in order to build appropriate models of letter recognition and reading. This is also important in order to build suitable measures of the orthographic similarity between letter strings, because visual similarity of characters probably plays a role in the confusability of strings (are the orthographic neighbours ''wine'', ''wire'', and ''wide'' equally similar to each other?). In addition, one can also consider that this is a particular (and quite convenient) example of visual shape processing whose study can lead to more general conclusions concerning visual pattern recognition.

Direct subjective similarity rating methods are handicapped by the fact that one does not know whether the similarity criteria actually used by subjects are those that are involved in perceptual processing. On the other hand, methods for collecting confusion data are quite time consuming since they require many trials and subjects in order to obtain sensitive measures. As pointed out by

Please address all correspondence to: Pierre Courrieu, Laboratoire de Psychologie Cognitive, CNRS (UMR 6146), Université de Provence, 29 avenue Robert Schuman, 13621 Aix-en-Provence cedex 1, France. Email: courrieu@up.univ-mrs.fr

http://www.tandf.co.uk/journals/pp/13506285.html DOI:10.1080/13506280444000049

Podgorny and Garner (1979), experimenters must degrade the viewing conditions in order to obtain a high enough confusion rate, while the similarity structure is known to depend on the type of degradation used. Moreover, these methods usually collect data on interval or ordinal scales, with the consequence that transforming the resulting measures into a metric is problematic both theoretically and practically, since a metric requires a ratio level of scaling (Lee, 2001). The strong structure of metric spaces (particularly Euclidean ones) allows one to apply powerful data analysis methods (e.g., multidimensional scaling), while nonmetric topological data are much harder to analyse given the relatively weak (and usually unknown) structure of the space they belong to.

In a seminal study, Podgorny and Garner (1979) found strong empirical evidence that the time required to discriminate upper-case letters is an increasing function of the perceptual similarity of these letters. The authors pointed out that response time based similarity measures potentially allow for solving many of the problems encountered with other types of similarity measure. An alternative approach is to measure saccade latencies to letters presented in peripheral vision (Jacobs, Nazir, & Heller, 1989). In Jacobs et al., a given target letter was presented centrally, followed by two letters, one of which was the target, in peripheral vision (one to the left, the other to the right of fixation). The participants had to move their eyes to the location of the target letter. Saccade latencies to the parafoveal target were used to generate a discrimination matrix for all pairs of lower-case letters. Thus, the idea of using reaction times for measuring the similarity of visual stimuli is not new. However, a reaction time is not a metric, and a true metric would be useful for methodological as well as theoretical purposes.

The goal of the present study is to propose a simple and sensitive measure of the similarity of stimulus pairs based on response times. The sensitivity of this measure should help economize on data collection, and its ratio scale will facilitate the building of metrics, including Euclidean ones, by using recently proposed mathematical tools (Courrieu, 2002). In the following study, we present two metrics using measures of the time taken to discriminate lower-case letters from the Roman alphabet. The first metric is as simple as possible, while the second one is Euclidean and allows for sophisticated similarity analysis methods to be applied, which will be illustrated with a dimensional analysis of the alphabet. We first describe a method for building a metric based on discrimination response times, and then an experiment that collected such response times before presenting the resulting metric.

BUILDING DISCRIMINATION TIME-BASED METRICS

This study was designed to examine whether discrimination time can provide the basis of a reliable measure of the visual similarity of lower-case letters embedded in character strings, as they usually appear in reading, with lateral

masking effects. Given its importance in the perception of printed words, the ''space'' character was included in the ''completed alphabet'', and it will be symbolically denoted '' in this paper. Our main goal was to find a simple continuous and strictly monotonic function of discrimination time, with this function being a true metric on the completed alphabet, and the obtained distances actually reflecting the visual dissimilarity of characters. One must remember that a metric, or distance, associated with a set S (e.g., an alphabet) is a numerical function d on the set of pairs of elements of S (e.g., letters) such that, for any three elements a, b, and c of S, one has the following four relations:

1. $d(a, b) = 0 \Leftrightarrow a = b$, that is, the distance is zero between identical elements only.
2. $d(a, b) \geq 0$, that is, a distance cannot be negative (positiveness).
3. $d(a, b) = d(b, a)$, that is, the distance between two elements does not depend on the ordering of these elements (symmetry).
4. $d(a, c) \leq d(a, b) + d(b, c)$, that is, between any two elements, there is no ''path'' shorter than their distance (triangle inequality).

Whenever the elements of a set are described in some coordinate system, one can compute a distance between two elements using an appropriate formula. For example, in a set of character strings of a given length, the Hamming distance between two strings is the number of positions where the characters of the two strings are different, while the special case of a Hamming distance of 1 corresponds to ''orthographic neighbours'' (Coltheart, Davelaar, Jonasson, & Besner, 1977). Another example is that of Minkowskian metrics in a real n-dimensional space:

$$d(X, Y) = \left(\sum_{i=1..n} |x_i - y_i|^r\right)^{1/r},$$

where the parameter r characterizes a particular Minkowskian distance, for example r = 1 for the ''city-block'' distance, r = 2 for the Euclidean distance. However, standard formulas cannot be used when no coordinates are provided. In such cases, one can attempt to empirically measure distances, for example by using a ruler, or measuring the latency of an echo. Sometimes, the empirical measures themselves are not metrics (i.e., they do not satisfy relations 1–4 above), but a true metric can be regularly derived from them by a simple transformation, which is what we are going to do in the present study.

Following Podgorny and Garner (1979), our working hypothesis is that discrimination time is an increasing function of the perceptual similarity of stimuli. If one excludes false discriminations, then the discrimination time of identical stimuli is a priori infinite. Hence, the discrimination time and the metric we seek to establish must be inversely related since a metric has a value of zero for identical elements (1). On the other hand, if two stimuli are distinct, then there is certainly a finite delay after which any subject with normal vision is able to

detect a difference. This implies that a measure inversely related to the discrimination time has a nonzero value for distinct stimuli, and this measure is positive (provided that the measure of discrimination time is also positive!), which satisfies (2). The symmetry property (3) can easily be obtained using an experimental measure where the stimuli to be compared in a pair are not a priori ordered, or eventually taking the mean discrimination time for the two possible ordered pairs. Of course, this is relevant provided that the original data matrix is not basically asymmetric. The most difficult property to obtain is triangle inequality (4). Fortunately, one knows that if a function d satisfies relations 1–3, then one can find a strictly positive real number γ such that d^{γ} also satisfies triangle inequality, and then is a metric (Courrieu, 2002, Lemma 2). Moreover, there is another strictly positive real number α, such that $\alpha \leq \gamma$ and d^{α} is a Euclidean metric for the set S (Courrieu, 2002, Theorem 3). Mathematical considerations leading to these results are quite technical, and the reader can find them in the cited reference, together with practical algorithms for applications. Obtaining a Euclidean metric allows one to embed the set S in a real vectorial Euclidean space (this is a multidimensional scaling procedure). The Euclidean metric is the only metric that is invariant through any orthogonal transformation of coordinates (e.g., rotation), which allows meaningful embedded solutions to be found by suitably rotating coordinate axes. This is not possible using non-Euclidean metrics, such as the ''city-block'' metric, for example, that is also frequently used in psychological modelling. Finally, the following remark will turn out to be helpful. Assume that a function d satisfies the relations 1–3, that the minimum of d for pairs of distinct stimuli is min(d), and the maximum is max(d). Then, if $\max(d) \leq 2 \min(d)$, then d necessarily satisfies the triangle inequality (4), that is d is a metric.

Summarizing the above considerations, we can note that one of the simplest way of building the desired metric d from discrimination time t consists of taking $d = t^{-\gamma}$, while $\gamma = 1$ (that is $d = 1/t$) is allowed provided that one obtains $\max(d) \leq 2 \min(d)$, where the min is taken for all pairs of distinct stimuli.

We now need to define an appropriate measure of discrimination time t. Let T be the total reaction time in a discrimination task where stimuli are presented in pairs and the subject must press a key as quickly as possible whenever the two presented stimuli are distinct, and must not respond whenever the two presented stimuli are similar (go/no-go task). Then the reaction time T is greater than the discrimination time t since in addition to t, there is at least a motor reaction time and possibly some other irrelevant partial times. Hence, we have $T = t + t0$, where t0 denotes the sum of the motor time and other irrelevant partial times. Now, assume an extreme case where t0 is much larger than t, in such a way that the contribution of t to T becomes negligible. Then one would obtain T close to t0 for any pair of stimuli, and given that certainly $1/t0 \leq 2/t0$, one would obtain a true metric that is not related to the similarity of stimuli. For that reason, it is essential to estimate t0 as precisely as possible in order to obtain an appropriate

measure of the discrimination time $t = T - t0$. In addition, one must of course verify that the obtained metric is related to some other well-known measures of similarity, and that it actually reflects the similarity of the stimuli. Finally, we note that if d is a metric, then for any real number $k > 0$, the quantity kd is the same metric with a different scale. This means that the scale of the metric can be chosen in such a way that its values be distributed around a reference value (for example 1 or 100). Then one obtains the general formulation of the metric model:

5. $t = a(T - t0)$,
6. $d = t^{-\gamma}$,

where $a > 0$ provides the desired scale, and $\gamma > 0$ is determined in order to provide triangle inequality for the whole set S (here the completed alphabet). Now, if it is required that the metric be Euclidean, then one replaces (6) by:

7. $d = t^{-\alpha}$,

where $0 < \alpha \leq \gamma$, and α is determined according to Courrieu (2002) in such a way that (S, d) is a Euclidean metric set, which means that there is a set of points in a real vectorial space, each point corresponding to an element of S (i.e., a character), and the Euclidean distances between these points are equal to the values of metric d.

DATA COLLECTION METHOD

Participants

Forty-two psychology students at the University of Provence participated in the experiment. They were all native speakers of French, aged 18–22 years, and reported normal or corrected-to-normal vision.

Design and materials

Stimuli were all lowercase letters of the Roman alphabet plus a blank space. All possible combinations of these stimuli were constructed giving 27×27 stimulus pairs. For a given group of participants half of the stimulus pairs were presented in alphabetic order (e.g., a b), and the other half in the opposite order (e.g., d c). Two independent groups of participants were presented with either of two different stimulus lists such that a given stimulus pair was seen in one order in one group of participants and in the other order in the other group of participants. Each participant saw all possible pairs of different characters once ($27 \times 26/2 = 351$ pairs), and 13 occurrences of each of the 27 identical pairs (no-go trials),

giving a total of 702 experimental trials. The order of trials was random with a different order for each participant.

Procedure

Letters were presented in lower-case Arial bold, 12 point. Each trial consisted in the presentation of a given pair of stimuli accompanied by three hash marks. First, five hash marks (#####) were presented for 250 ms, followed by three hash marks with two test characters (e.g., #a#b#). Stimulus presentation and recording of participant responses was done using the DMDX software (Forster & Forster, 2003). Participants were requested to respond as rapidly and as accurately as possible if the two test characters were different. They were instructed not to respond to pairs of identical stimuli (go/no-go procedure). The screen was cleared when subjects responded. A timeout of 2 s was used and the timeout data were ignored in the analysis. In a preliminary phase, simple detection RT was measured for each participant by asking him/her to respond as rapidly as possible whenever two fixed characters (#$#£#) appeared among the hash marks. In this phase, the prior five hash marks were presented for a randomly variable time (uniformly sampled between 600 and 1000 ms), in order to avoid time-based strategies. The mean of the 20 shortest RTs out of 25 trials was used to calculate the baseline detection RT (that is t0) for each participant. Participants were then given a set of 20 practice trials using pairs of numbers, of which half were identical and half composed of different numbers. Similarly to the letter stimuli in the main experiment, participants had to respond only when the numbers were different (e.g., #5#8#). After being informed that the stimuli would now consist of letters and sometimes spaces, the participants moved on to the main experiment. The 702 experimental trials were broken into three blocks of 234 trials with a short rest between blocks.

COMPUTATION OF THE METRICS

Discrimination time matrix

The additional time t0 was estimated for each subject as described above in the simple detection task. This value was subtracted from each response time of the subject in the discrimination task, and the resulting times were divided by their mean in order to obtain a mean discrimination time equal to 1 for each subject. Then each cell of the 27×27 discrimination time matrix received the mean of the discrimination times provided by all subjects who responded to the corresponding pair of characters. Given that the left–right disposition of characters in a pair was reversed for half the subjects, the data concerning cells of symmetrical positions in the matrix were not provided by the same subjects.

Symmetrization of the discrimination time matrix

As a measure of symmetry, a linear correlation coefficient was computed between the 351 upper out-diagonal values of the discrimination time matrix and the corresponding lower out-diagonal values of the same matrix. The resulting correlation was $r = .654$, $p < .001$. As one can see, the symmetry is not perfect, however the values of symmetric data cells vary in a significantly similar way. Hence, the matrix was symmetrized by taking the mean value of the symmetric cells of the original matrix. This provided the matrix of the t values, while the diagonal coefficients (corresponding to identical pairs) are a priori assumed to be infinite.

Determining the metrics

Taking $d = 1/t$ (i.e., $\gamma = 1$), one obtained $\min(d) = 0.6544$ and $\max(d) = 1.295$. Then the relation $\max(d) \leq 2 \min(d)$ was satisfied, that is $d = 1/t$ is a metric for the completed alphabet. However, this metric was not Euclidean since the computation of the number α as the power for obtaining a Euclidean metric (equation 7) provided $\alpha = .7321$. The values of the (non-Euclidean) metric d (multiplied by 100) for the completed alphabet are reported in Table 1.

COMPARISON WITH OTHER DATA

A possibility was that the subjects did not actually compare the characters but compared their name after verbal encoding. In order to test this eventuality, the mean discrimination time of each letter was compared to the corresponding letter naming time obtained in another experiment (Petit & Grainger, 2004). In Petit and Grainger's study, delayed naming latencies were subtracted from online naming latencies (both obtained on the same trial) in order to extract purely articulatory influences on the time taken to name letters. It turned out that only a small and nonsignificant correlation (.11) existed between the two sets of data. Thus there is no evidence for a verbal encoding mediation in the discrimination task.

Next we examined whether the elaborated metric is actually related to the visual similarity of letters. This was first done by comparing the data of Table 1 to a letter confusion matrix (Table 2) obtained with preschool children just before they began learning to read (Courrieu & de Falco, 1989). These children had just sufficient maturity for beginning to read, however their perception was not influenced by language factors such as the name and frequency of letters, or grapheme–phoneme correspondences. This is therefore a way of measuring pure visual letter confusability. The methodology was of course adapted to young prereaders, without any time constraint, and the obtained confusion matrix is in fact quite ''empty'' (many zeros), as one can see in Table 2. The linear correlation between the comparable data of Table 1 and Table 2 (325 out-diagonal

TABLE 1
Matrix of the metric built from letter discrimination times

	″	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
a	81																				
b	85	110																			
c	86	104	110																		
d	78	106	69	107																	
e	78	97	111	99	111																
f	80	107	106	106	91	107															
g	91	110	107	115	100	117	105														
h	75	113	98	112	104	115	95	112													
i	77	104	108	104	103	104	85	99	105												
j	78	108	106	110	97	96	97	117	101	78											
k	83	115	108	128	105	117	104	118	88	94	96										
l	73	99	98	99	95	92	77	104	90	70	93	87									
m	83	103	102	107	100	97	106	101	99	100	109	102	105								
n	88	105	105	112	103	117	109	103	82	98	103	113	99	79							
o	88	96	99	91	95	103	102	116	112	106	106	129	100	94	104						
p	83	109	87	112	101	94	101	94	119	106	102	102	94	102	103	102					
q	85	112	105	108	94	121	107	74	103	96	93	105	106	107	104	96	68				
r	77	106	103	96	111	109	99	103	106	91	101	102	94	97	107	118	108	104			
s	76	98	117	108	120	102	112	129	118	99	100	108	101	97	110	100	114	119	93		
t	65	101	106	103	105	102	78	113	101	88	94	88	71	104	106	110	102	108	84	101	
u	89	111	105	99	104	106	97	100	97	109	104	100	110	87	92	98	113	107	96	120	101
v	78	107	107	115	116	111	86	110	119	102	103	98	82	89	104	121	120	120	101	111	101
w	85	99	93	100	106	99	106	97	101	103	100	92	95	84	95	100	108	103	95	97	98
x	83	112	114	113	108	114	110	113	103	97	91	93	87	103	102	111	114	113	93	97	101
y	86	128	116	124	116	111	109	109	117	100	97	107	82	93	118	119	109	109	106	121	98
z	77	107	112	110	114	111	102	119	111	101	107	109	90	106	108	107	109	103	98	103	101

	u	v	w	x	y
v	98				
w	104	86			
x	97	97	102		
y	109	105	92	93	
z	96	102	98	82	104

The column entry ″ corresponds to the space character. The values of the metric were multiplied by 100 and rounded for readability. Dividing the values in the table by 100 and raising them to the power 0.7321, one obtains a Euclidean metric on the alphabet.

coefficients) is $r = -.355$, $p < .01$. The linear correlation between the Euclidean version of the metric and the children's confusion data is $r = -.366$, $p < .01$. The fact that the correlations are negative simply means that the confusability of letters increases as their distance decreases. Hence, our main hypothesis is confirmed, and it is clear that inverse discrimination time is not only a true metric on the completed alphabet, but also significantly affected by visual similarity.

TABLE 2
Letter confusion matrix of preschool children (data from Courrieu & de Falco, 1989)

a	123																				
b	2	*99*																			
c	2	0	*133*																		
d	2	92	0	*121*																	
e	0	0	4	0	*132*																
f	0	0	0	0	0	*129*															
g	0	3	0	2	0	0	*128*														
h	0	0	0	0	0	0	0	*117*													
i	0	0	0	0	0	0	0	0	135												
j	0	0	0	0	0	2	1	0	6	*131*											
k	0	0	0	0	0	0	0	0	0	0	*120*										
l	0	0	0	0	0	0	0	0	6	2	0	*128*									
m	0	0	0	0	0	0	0	0	0	0	0	0	*138*								
n	0	0	0	0	0	0	0	8	0	0	0	0	0	*112*							
o	4	0	8	0	0	0	0	0	0	0	0	0	0	0	*126*						
p	0	27	0	22	0	0	1	0	0	0	0	0	0	0	0	*102*					
q	2	20	0	22	0	0	1	0	0	0	0	0	0	0	0	80	*99*				
r	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	*120*			
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*118*		
t	0	0	0	0	0	44	0	0	0	1	0	0	0	0	0	0	0	2	0	*122*	
u	0	0	0	0	0	0	0	4	0	0	1	0	1	23	0	0	0	1	0	0	*114*
v	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
w	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	1
x	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
y	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u

v	125				
w	0	*133*			
x	0	0	*141*		
y	0	0	0	*143*	
z	0	0	2	0	*129*
	v	w	x	y	z

The linear corelation between the 325 out-diagonal coefficients of this matrix and the corresponding coefficients in Table 1 is $r = -.355$, $p < .001$, while the correlation with the Euclidean version of the metric is $r = -.366$, $p < .001$.

In addition, we compared our discrimination time based metrics with the letter confusion data obtained by Bouma (1971) with skilled adult readers. Bouma's matrices were symmetrized by averaging the values of out-diagonal symmetric position cells. The linear correlation between Bouma's ''distance vision'' data and our metric (325 out-diagonal coefficients) was $r = -.267$, $p < .01$, while the correlation of these data with the Euclidean version of the metric was $r = -.271$, $p < .01$. The linear correlation between Bouma's ''eccentric

vision'' data and our metric (only 300 out-diagonal coefficients, since Bouma omitted the stimulus ''y'') was $r = -.322$, $p < .01$, while the correlation of these data with the Euclidean version of the metric was $r = -.327$, $p < .01$. As one can see, the Euclidean version of the metric is at least as good as the basic metric in all cases.

Finally, we compared our metrics with the saccade latencies obtained by Jacobs et al. (1989). The observed correlation with the simple metric was $r = -.265$, $p < .01$, while the correlation with the Euclidean metric was $r = -.269$, $p < .01$. However, if one attempts to interpret the saccade latencies in the experiment of Jacobs et al. as discrimination times, one expects a strong positive relation to the inverse of our simple metric. The obtained correlation was $r = .291$, $p < .01$, which is significant but not as strong as expected. A possibility is that the perception of letters in peripheral vision, which was required in Jacobs et al.'s experiment, has some differences with the foveal perception required in our experiment. If this is the case, one might expect a strong relation between the saccade latencies and the ''eccentric vision'' data of Bouma (1971). The obtained correlation is $r = .334$, $p < .01$, which is significant but not particularly strong. Hence, we conclude that, although these various methods obviously captured some common features, they are far from being equivalent.

DIMENSIONAL ANALYSIS OF THE SIMILARITY

Another way of verifying that discrimination time actually reflects the visual similarity of letters is to assess whether a set of meaningful dimensions can account for the observed variations in discrimination time. Note however that our purpose here is not to find an optimal set of features in order to build or to test a special theory of letter perception. We need only show that a set of obvious visual similarity factors exists which account for a substantial part of the data. Obviously, visual similarity of letters will depend to some extent on the particular type font tested (Arial, 12 point, bold font, in the present experiment).

Computational method

First we used a monotonic embedding method (Courrieu, 2002) in order to compute the number α of equation 7, which provided the result $\alpha = .7321$, as mentioned above. This is a special multidimensional scaling method, which provides a set of points in the form of a singular triangular matrix, and the dimension of the Euclidean embedding space is minimized as a consequence of the appropriate determination of α. The dimension of the embedding space was found to be 25. This first solution also allowed us to compute the centre of gravity of the 27 representative points and then to centre the cluster. However, this solution depends on the order of the elements of the set, which in the present case is the (arbitrary) alphabetical order. Since we are now in a Euclidean space, we can apply any orthogonal transformation to the coordinates without changing

the Euclidean distances between the points. A solution that is mathematically of special interest is the solution whose coordinate axes are the eigendimensions of the cluster.[1] In order to obtain this special solution, we computed the 27 × 27 symmetric matrix A = V′V, where V is the matrix whose 27 column vectors are the centred coordinates of the representative points previously found, and the ′ denotes the transposition operator. The matrix A is positive definite (matrix of scalar products) and of rank 25, which means that A has exactly 25 strictly positive real eigenvalues and two zero eigenvalues. Since A is symmetric, it can be written in the form A = QDQ′, where Q is an orthogonal matrix whose columns are the eigenvectors of A, and D is the diagonal matrix of eigenvalues of A. Since A has only nonnegative eigenvalues, the solution is the matrix E = $QD^{1/2}$, which has 25 nonzero columns and whose 27 row vectors are the coordinates of the representative points on the 25 eigendimensions. The computation of the matrices Q and D was performed using the well-known Jacobi's algorithm applied to the symmetric matrix A. The obtained matrix E is reported in Table 3.

Similarity factors

An inspection of Table 3 suggests that certain eigendimensions (particularly the first ones) would be interpretable, however the matrix as a whole is not easy to read. This results from the fact that successive eigendimensions maximize a purely quantitative criterion (explained inertia), which frequently leads eigenaxes to adopt an intermediate position between several important, but distinct, groups of elements (characters). Hence, we applied a series of orthogonal rotations to the matrix of eigenvectors in order to obtain a more readable solution where each dimension mainly accounts for the similarity of only one group of characters.[2] The result of this orthogonal transformation is reported in Table 4, from which one can verify that the Euclidean distances between the points representative of the characters remain unchanged and are equal to the power $\alpha = .7321$ of the initial metric (Table 1). Low absolute value coordinates are poorly reliable since they can vary substantially with small rotations and the data are not noise free. Hence, in Table 5, we provide a qualitative reading of the dimensions, where we take into account only coordinates whose absolute value is at least equal to 0.20. For each dimension (X01 to X25), we provide the main similarity class of characters associated to that dimension, the main contrasting

[1] Eigendimensions (or ''principal components'', according to Hotelling's, 1936, terminology) form an orthogonal coordinate system whose origin is the centre of gravity of the cluster, and with the particularity that the first k dimensions account for a maximum part of the total inertia (sum of squared coordinates), for any k lower or equal to the number of dimensions.

[2] A computer program, whose technical detail is of minor interest here, has been developed to help finding suitable rotations. Interested readers can contact the first author for more details.

TABLE 3
Monotonic embedding of the completed alphabet in a Euclidean space of dimension 25 (eigendimensions)

	E01 *9.93%*	*E02* *8.52%*	*E03* *7.45%*	*E04* *6.58%*	*E05* *5.87%*	*E06* *5.72%*	*E07* *5.34%*	*E08* *4.97%*	*E09* *4.66%*	*E10* *4.37%*	*E11* *4.09%*	*E12* *3.82%*	*E13* *3.68%*
″	−.063	.052	−.095	−.055	.030	.024	−.027	−.053	−.029	−.052	−.022	.005	.058
a	.080	.368	−.058	−.057	−.019	.042	−.282	−.105	.232	−.180	.001	−.262	.069
b	.312	−.079	.010	−.230	−.131	.263	.201	−.307	−.047	.143	−.013	.040	.141
c	.149	.386	−.042	−.034	−.044	−.319	.076	.014	−.260	.198	−.048	−.166	−.093
d	.337	−.120	−.084	−.278	−.072	.104	.237	−.131	.124	.149	.142	−.026	.083
e	.014	.332	−.188	.109	−.188	.203	−.006	.230	−.271	−.214	.116	−.123	.217
f	−.070	−.100	−.185	−.276	−.260	−.248	−.014	.062	.119	−.108	.017	.180	−.126
g	.340	−.284	.074	.276	−.115	−.220	−.259	−.039	.065	.071	.011	−.160	.091
h	.012	−.180	.210	−.413	.234	.031	−.073	.167	−.171	−.046	−.219	−.157	.002
i	−.142	−.090	−.246	−.013	.071	−.117	−.137	.202	.184	.220	.076	.019	.068
j	−.102	−.069	−.212	−.005	.201	.104	.089	.235	.024	.117	.384	−.097	−.047
k	−.255	−.320	.010	−.133	.136	.246	−.125	−.128	−.155	−.154	.092	−.070	−.254
l	−.225	−.095	−.239	−.098	−.141	−.014	.049	.084	.106	.004	−.251	−.079	.081
m	.047	.087	.390	.121	−.125	.174	−.074	.154	−.006	.044	.011	.249	.009
n	.128	−.024	.378	−.130	.167	.021	−.169	.237	.165	.014	−.101	.121	.215
o	.323	.354	−.006	−.020	.043	−.025	.249	.144	.165	.007	−.103	.093	−.271
p	.349	−.144	−.267	.229	.023	.159	−.042	−.056	−.102	−.259	−.047	.209	.070
q	.399	−.266	−.138	.252	.241	−.117	−.078	−.021	.036	−.043	−.001	−.013	−.198
r	−.168	.038	.009	.066	.071	−.218	−.160	−.280	−.250	.247	.048	.150	.164
s	−.206	.412	−.069	.081	.274	.193	−.139	−.124	.072	.117	.010	.226	−.091
t	−.231	−.042	−.231	−.151	−.085	−.098	−.111	−.015	−.160	−.028	−.160	.155	−.056
u	.052	−.003	.371	−.052	−.033	−.275	.139	.055	−.207	−.206	.263	.061	−.052
v	−.295	−.038	.192	.022	−.396	−.012	−.077	−.121	.227	−.063	.189	.003	−.077
w	−.043	.054	.229	.157	−.146	.222	−.079	−.112	−.038	.161	−.131	−.219	−.239
x	−.315	−.047	.120	.114	.271	−.044	.263	−.084	.108	−.009	.071	−.111	.153
y	−.234	−.245	.006	.361	−.172	.109	.269	.215	−.066	.142	−.178	−.005	.006
z	−.194	.066	.062	.153	.165	−.191	.278	−.222	.134	−.273	−.158	−.024	.076

	E14 *3.47%*	*E15* *3.01%*	*E16* *2.93%*	*E17* *2.71%*	*E18* *2.49%*	*E19* *2.27%*	*E20* *2.06%*	*E21* *1.81%*	*E22* *1.57%*	*E23* *1.21%*	*E24* *0.94%*	*E25* *0.54%*
″	−.055	−.093	.155	−.131	−.177	−.065	.207	−.004	.108	−.217	.056	.064
a	−.251	−.098	−.200	−.024	.000	−.146	−.104	.090	.015	.016	−.017	.003
b	.107	−.047	−.073	.069	.148	−.089	.125	.085	.119	.050	−.034	−.058
c	.204	.152	−.065	−.090	−.206	−.007	−.052	.119	.052	.023	.031	−.030
d	−.187	.025	.076	.003	−.219	.145	−.091	−.042	−.134	−.005	−.026	.035
e	.034	.024	.155	.091	.042	.083	.002	−.120	−.032	−.008	−.078	−.065
f	.007	−.062	.189	−.010	.113	.060	−.246	.089	.069	−.042	−.005	−.032
g	−.140	.096	.164	−.014	.106	.127	.120	.021	.041	.052	.128	−.035
h	.017	−.054	.214	−.043	.072	−.154	−.004	−.007	.048	.029	−.014	.028
i	.056	.099	−.062	.347	−.027	−.050	.080	.041	.101	−.099	−.071	.028
j	.110	−.251	−.060	−.087	.049	−.001	−.012	−.013	.021	.099	.128	.021
k	−.055	.218	−.091	.122	−.091	.006	−.029	.004	−.049	−.041	.078	−.093
l	.081	.194	−.050	.005	.057	−.046	.037	−.045	−.190	.143	.029	.085
m	−.109	.057	.044	.087	−.197	−.102	−.090	−.090	.145	.129	.040	.046
n	.186	−.051	−.169	−.079	−.033	.160	−.019	.023	−.110	−.069	−.000	−.073
o	−.098	.075	−.057	.017	.168	−.068	.075	−.180	−.023	−.080	.077	−.040
p	.174	.094	−.109	−.081	.035	−.022	−.098	.075	.036	−.059	.060	.084
q	.060	−.079	.040	−.049	−.087	−.087	.005	−.087	−.045	.063	−.191	−.024
r	−.041	−.085	−.028	.033	.107	−.140	−.137	−.162	−.123	−.050	.033	−.010
s	−.036	.068	.216	−.050	.063	.080	.074	.184	−.079	.081	−.035	−.004
t	−.164	−.126	−.212	−.107	−.047	.207	.145	−.101	.119	.079	−.042	−.007
u	−.114	.015	−.107	.088	.110	.019	.115	.133	−.097	.008	−.057	.081
v	.238	.025	.030	−.165	−.042	−.127	.106	−.069	−.027	.001	−.039	−.026
w	.123	−.137	−.015	.091	.096	.201	−.086	−.034	.012	−.077	−.032	.094
x	−.066	.243	−.020	−.189	.099	.073	−.093	−.076	.166	−.036	−.070	.015
y	−.199	−.109	−.038	−.069	−.019	−.106	−.034	.170	−.062	−.048	−.021	−.060
z	.116	−.192	.072	.234	−.119	.048	.006	−.006	.010	.058	.071	−.027

The Euclidean distances between the points representative of the characters are equal to the values in Table 1, divided by 100 and raised to the power $\alpha = .7321$.

TABLE 4
A meaningful orthogonal rotation of the eigendimensions reported in Table 3

	X01 *9.03%*	*X02* *7.81%*	*X03* *7.73%*	*X04* *5.24%*	*X05* *4.06%*	*X06* *4.07%*	*X07* *5.00%*	*X08* *3.57%*	*X09* *4.24%*	*X10* *3.91%*	*X11* *3.92%*	*X12* *2.38%*	*X13* *3.68%*
″	−.053	.068	−.063	.012	−.025	−.025	.007	−.140	−.082	−.025	.007	−.033	.013
a	.008	.351	.137	−.110	.099	−.145	.031	.000	−.149	.320	−.118	−.055	−.064
b	.337	−.024	.000	.456	−.115	.164	−.109	−.001	.003	.053	.139	.061	−.041
c	.034	.381	.166	−.110	−.072	.013	−.007	−.087	.196	−.071	.380	−.024	−.187
d	.342	.042	−.099	.449	−.087	−.103	.019	−.107	.091	.005	−.117	.008	−.127
e	.044	.362	.000	−.124	−.148	−.116	−.096	−.071	−.141	−.149	−.133	−.015	.094
f	.076	−.016	−.216	−.090	−.046	.356	.071	.045	.179	.081	−.140	−.125	.132
g	.430	−.165	−.042	−.233	.087	−.174	.000		.057	.320	.001	.029	.025
h	−.004	−.254	.096	.063	−.246	.035	.447	−.059	−.167	.055	.164	−.175	−.018
i	−.188	−.015	−.271	−.007	.048	−.112	.097	−.018	.263	.105	−.086	.207	.166
j	−.191	−.029	−.227	.077	.020	−.165	.030	−.066	.261	−.175	−.009	−.109	−.062
k	−.177	−.282	−.172	−.028	−.076	−.136	−.030	−.121	−.157	−.040	−.057	−.114	−.189
l	−.145	.001	−.275	.066	.032	.015	.085	.001	−.158	.135	−.014	.209	.033
m	.054	−.103	.385	−.024	.106	−.011	−.108	−.175	−.000	.040	−.091	−.084	.173
n	.030	−.184	.329	.004	.027	−.068	.447	−.071	.150	−.052	−.065	.073	.114
o	.173	.371	−.198	.115	.046	.109	.111	.199	−.022	−.144	−.084	.108	−.035
p	.331	.083	−.269	−.147	−.040	−.084	−.067	−.033	−.110	−.160	.104	−.043	.130
q	.331	−.056	−.211	−.166	.135	−.131	.079	.061	.013	−.215	.154	−.039	.039
r	−.207	−.027	.010	−.071	.018	.119	−.108	.063	.097	.181	.380	.175	−.055
s	−.364	.348	.121	.059	.050	.045	.003	−.065	−.099	−.004	−.003	.018	.256
t	−.193	.035	−.243	−.146	−.150	.356	.017	−.122	.001	.107	−.134	−.008	−.180
u	.072	−.160	.326	−.214	−.247	.009	−.095	.130	.158	−.121	−.128	.173	−.271
v	−.077	−.170	.120	−.039	.335	.090	−.271	−.136	.143	.057	−.109	.002	.062
w	−.004	−.135	.221	.037	.133	.125	−.139	−.162	−.214	−.008	.022	−.043	−.027
x	−.357	−.072	.051	.170	−.020	−.197	−.079	.311	.040	.033	−.051	−.241	−.099
y	−.120	−.237	−.137	.006	.335	.066	−.098	.150	−.244	−.144	−.065	.006	−.143
z	−.182	−.110	.067	−.008	−.199	−.035	−.235	.313	−.108	−.180	.055	.040	.258

	X14 *3.40%*	*X15* *2.60%*	*X16* *2.60%*	*X17* *2.81%*	*X18* *3.52%*	*X19* *3.87%*	*X20* *2.48%*	*X21* *2.68%*	*X22* *3.82%*	*X23* *3.01%*	*X24* *2.55%*	*X25* *2.04%*
″	.006	.084	−.001	.105	−.079	−.076	.040	.071	.020	.140	−.088	−.344
a	.128	.000	−.234	.025	−.059	.103	−.109	−.027	.353	.000	.000	.000
b	.288	.001	−.059	.039	.078	−.048	−.100	−.109	.000	−.000	.000	−.000
c	−.222	−.012	.020	−.111	−.078	−.084	−.003	−.228	−.033	−.088	.090	−.062
d	−.249	.029	−.038	.084	−.008	.041	−.056	.130	.073	−.088	−.053	−.007
e	.089	−.056	−.059	−.104	−.118	−.272	−.146	.163	−.179	.183	.196	.136
f	−.231	−.268	−.026	.054	.006	−.120	.029	.040	.177	.002	−.001	.003
g	−.079	.062	.090	−.090	−.054	.089	−.071	−.007	−.256	.030	−.136	−.079
h	−.001	−.116	−.052	.132	−.000	.027	.045	−.038	−.032	.179	.144	−.041
i	.051	−.088	−.076	−.122	.004	.078	.008	−.067	−.070	−.120	.239	−.159
j	.148	−.097	−.217	−.007	−.217	.087	.081	.028	−.072	.129	−.146	.197
k	.000	−.067	−.124	−.025	.494	.000	.000	.000	.000	.000	.000	.000
l	−.141	.002	.120	−.030	−.003	−.190	.087	−.195	.063	.074	.110	.164
m	.009	.108	−.053	.131	.044	−.037	.169	.121	−.161	−.201	.175	.091
n	.122	.174	.086	−.070	−.012	−.088	−.077	−.002	.114	−.102	−.089	.082
o	−.000	−.033	−.030	−.102	.018	.153	.419	−.000	.000	.000	.000	−.000
p	.294	.030	.193	.002	.091	−.074	.064	.043	.081	−.216	−.131	.094
q	−.000	.103	.088	.066	.019	.432	.000	.000	.000	.000	.000	−.000
r	−.000	.034	.031	.036	.040	.031	−.132	.386	.000	−.000	.000	.000
s	.020	−.068	.000	.132	.163	.148	−.110	−.007	−.198	−.042	−.294	−.032
t	.000	.304	−.000	.000	−.000	−.000	−.000	−.000	−.000	−.000	−.000	.000
u	.000	−.170	.030	.199	.000	−.000	.000	−.000	−.000	−.000	.000	−.000
v	−.044	−.051	.091	.042	.094	−.202	−.007	−.135	.163	.334	.000	.000
w	−.002	−.103	−.030	−.427	−.040	.138	−.120	−.046	−.069	.006	−.052	.051
x	−.000	.018	.341	.000	.000	−.000	−.000	−.000	−.000	.000	.000	−.000
y	−.030	.009	−.024	.074	−.281	−.178	.022	−.017	−.229	−.227	.074	−.001
z	−.156	.171	−.067	−.031	−.100	.045	−.032	−.106	.254	.011	−.035	−.105

The first 14 dimensions (X01 to X14) are visual similarity factors, while the next 11 dimensions plausibly correspond to remaining distinctive features of individual characters.

TABLE 5
Main character similarity classes, contrasting classes, and possible critical features corresponding to the dimensions reported in Table 4

X01. Main similarity class: { b, d, g, p, q }
Main contrasting class: – { r, s, x }
Common feature: a circle with an ascender or descender. Note that in the font used, the graphic realisation of the letter ''g'' is the one that looks like ''q'' and the digit ''9''.

X02. { a, c, e, o, s }, – { h, k, y }
Small curvilinear shapes.

X03. { m, n, u, w }, – { f, i, j, l, p, q, t }
Repeated small vertical strokes.

X04. { b, d }, – { g, u }
Ascender subclass of X01.

X05. { v, y }, – { h, u }
The v-shape.

X06. { f, t }, – {}
An ascender with a horizontal bar.

X07. { h, n }, – { v, z }
The n-shape.

X08. { x, z }, – {}
Four corners and a diagonal of a square (as in ''%'').

X09. { i, j }, – { w, y }
The i-shape.

X10. { a, g }, – { q }
In the used font, one can obtain the character ''g'' by reversing and stretching the character ''a''.

X11. { c, r }, – {}
An upper-left arc of circle.

X12. { i, l }, – { x }
The i-shape is similar to an l-shape with an interruption.

X13. { s, z }, – { u }
Sigmoid shape.

X14. { b, p }, – { c, d, f }
The circle-at-right subclass of X01.

The next 11 dimensions plausibly concern remaining distinctive features of individual characters:
X15. { t }, – { f }
X16. { x }, – { a, j }
X17. { w }, – { u }
X18. { k }, – { j, y }
X19. { q }, – { v }
X20. { o }, – {}
X21. { r }, – { c }
X22. { a }, – { g, y }
X23. { v }, – { m, p, y }
X24. { s }, – { i }
X25. { ″ }, – {}

class (opposite sign), and we comment on the likely common feature of the main similarity class. The contrasting class is a set of characters that do not share the feature common to the characters of the similarity class, and can therefore help to identify this feature.

The first 14 dimensions, that are obviously visual similarity factors, account for 68.02% of the total inertia (sum of squared coordinates), while the 11 remaining dimensions are not meaningless. We note that the similarity factors widely account for the letter confusion classes observed with preschool children (Courrieu & de Falco, 1989), which was of course expected from the significant correlation between the two sets of data. There are also strong similarities between our results and other well-known factor analyses of letter similarity data (e.g., Dunn-Rankin, 1968; Dunn-Rankin et al., 1968; Kuennapas & Janson, 1969).

Comments on features

Our purpose is not to develop a theory of letter perception on the basis of this analysis; however, our results have some implications for existing or potential theories. We observe that certain features reduce to the presence of a certain shape, without any transformation, as part of the characters (X05, X06, X07, X09, X11, X12). Such features could be compatible with a simple ''template matching'' theory of letter recognition (see Neisser, 1967). However, one knows that this type of theory cannot account for the observed capability of the visual perception to recognize objects under various transformations. Hence, the concept of ''prototype'' has been proposed in order to introduce more flexibility in the recognition process (Posner & Keele, 1968), and there are to date neurocomputational models that are able to extract (from strings) and to recognize characters with strong invariance to transformations such as translation, scale change, stretching, and a limited amount of random distortion (Fukushima, 1992; Fukushima & Imagawa, 1993; Fukushima, Miyake, & Ito, 1983). However, we note that several of our similarity factors assume features invariant through affine transformations that are not supported by these models, such as symmetries (mirrors) with respect to the vertical and/or the horizontal axes (X01, X04, X10, X13, X14). We note also that certain features are more abstract than usually assumed by the notion of prototype (curvilinearity for X02, repetition for X03), however this notion itself seems quite flexible. Another observation is that certain features are hierarchically dependent: X04 and X14 provide partitions of X01. Finally, certain details appear methodologically important for the use of masked priming techniques: While the space character (X25) seems to be poorly confusable, there are some nonalphabetical characters that contain detectable letter features, such as X08 in %. In fact, the character % is known to provide strong masking effects on target letter strings (Peressotti & Grainger, 1999).

CONCLUSION

The present study examined whether inverse discrimination time (equations 5 and 6), as well as its monotonic transform into a Euclidean metric (equation 7), could account for the visual similarity relations among lower-case letters. This was verified by comparing the two metrics to known letter confusion data, and by a dimensional analysis of the factors governing the variations of the Euclidean version of the metric. The proposed methodology can certainly be used for other types of stimuli, and the available data seem sufficiently reliable to be reused in other studies. The main advantage of discrimination time-based metrics is that they provide a sensitive measure of perceptual similarity, while requiring much less experimental data than the usual confusion based approaches. Another advantage is that the method provides a ratio level of data scaling, whereas, as pointed out by Lee (2001), usual methods collect similarity data on interval or ordinal scales. As a consequence, the transformation of data into a metric is highly simplified and theoretically well founded with the present approach, whereas this is frequently problematic with other commonly used methods. Finally, the dimensional analysis of the alphabet clearly illustrates the sensitivity of the method and provides useful empirical data for testing models of letter recognition. An important conclusion is that letter recognition models must account for features invariant to a number of affine transformations, including symmetries (if not rotations). Finally, we note that letters are quite simple shapes, and it remains to be seen whether all detected features are relevant in the context of letter string and word perception.

REFERENCES

Bouma, H. (1971). Visual recognition of isolated lower-case letters. *Vision Research*, *11*, 459–474.

Coltheart, M., Davelaar, E., Jonasson, J. T., & Besner, D. (1977). Access to the internal lexicon. In S. Dornic (Ed.), *Attention and performance VI* (pp. 535–555). London: Academic Press.

Courrieu, P. (2002). Straight monotonic embedding of data sets in Euclidean spaces. *Neural Networks*, *15*, 1182–1193.

Courrieu, P., & de Falco, S. (1989). Segmental vs. dynamic analysis of letter shape by preschool children. *CPC: European Bulletin of Cognitive Psychology*, *9*(2), 189–198.

Dunn-Rankin, P. (1968). The similarity of lowercase letters of the English alphabet, *Journal of Verbal Learning and Verbal Behavior*, *7*, 990–995.

Dunn-Rankin, P., Leton, D. A., Shelton, V. F. (1968). Congruency factors related to visual confusion of English letters. *Perceptual Motor Skills*, *26*, 659–666.

Forster, K. I., & Forster, J. C. (2003). DMDX: A Windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, and Computers*, *35*, 116–124.

Fukushima, K. (1992). Character recognition with neural networks. *Neurocomputing*, *4*, 221–233.

Fukushima, K., & Imagawa, T. (1993). Recognition and segmentation of connected characters with selective attention. *Neural Networks*, *6*, 33–41.

Fukushima, K., Miyake, S., & Ito, T. (1983). Neocognitron: A neural network model for mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, *13*, 826–834.

Hotelling, H. (1936). Simplified calculation of principal components. *Psychometrika*, *1*, 27–35.

Jacobs, A. M., Nazir, T. A., & Heller, O. (1989). Perception of lower-case letters in peripheral vision: A discrimination matrix based on saccade latencies. *Perception and Psychophysics*, *46*, 95–102.

Kuennapas, T., & Janson, A. J. (1969). Multi-dimensional similarity of letters. *Perceptual Motor Skills*, *28*, 3–12.

Lee, M. D. (2001). Determining the dimensionality of multidimensional scaling representations for cognitive modeling. *Journal of Mathematical Psychology*, *45*, 149–166.

Neisser, U. (1967). *Cognitive psychology*. New York: Appleton-Century-Crofts.

Peressotti, F., & Grainger, J. (1999). The role of letter identity and letter position in orthographic priming. *Perception and Psychophysics*, *61*, 691–706.

Petit, J. P., & Grainger, J. (2004). Letter naming latency estimated using a new trial delay procedure. *Manuscript in preparation.*

Podgorny, P., & Garner, W. R. (1979). Reaction time as a measure of inter- and intraobject visual similarity: Letters of the alphabet. *Perception and Psychophysics*, *26*, 37–52.

Posner, M. I., & Keele, S. W. (1968). On the genesis of abstract ideas. *Journal of Experimental Psychology*, *77*, 353–363.

Original manuscript received February 2003
Revised manuscript received February 2004

II.B Modèles de codage de données

De très nombreux problèmes cognitifs peuvent être décrits comme (ou ramenés à) des problèmes d'approximation de fonctions numériques, même lorsque les problèmes sont initialement présentés sous forme symbolique, et pourvu que l'on encode convenablement les données (Courrieu, 1994b). Une fonction numérique suppose définies trois entités: un espace numérique d'entrée, un espace numérique de sortie, et une "machinerie" (par exemple un réseau de neurones) établissant un lien fonctionnel entre les éléments de l'espace d'entrée et les éléments de l'espace de sortie. La machinerie fonctionnelle requise sera par ailleurs d'autant moins complexe que les topologies des espaces d'entrée et de sortie sont plus compatibles. Pour faire simple, disons que les topologies sont compatibles si des éléments proches dans l'espace d'entrée ont toujours des images proches dans l'espace de sortie, et la compatibilité est encore plus grande (équivalence) si la réciproque est également vraie. Par ailleurs, la topologie des espaces d'entrée et de sortie est largement dépendante du codage que l'on adopte pour les données correspondantes. On voit donc qu'il est essentiel de déterminer des codages appropriés lorsqu'on veut modéliser des fonctions cognitives, sans quoi il peut s'avérer pratiquement impossible d'approcher convenablement les fonctions à modéliser. Il faut bien reconnaître que le codage est souvent réalisé de manière empirique et intuitive par les modélisateurs, avec plus ou moins de succès suivant les cas. J'ai pour ma part consacré beaucoup d'efforts à la recherche de méthodes de codage appropriées pour différentes sortes de données de base. Le cas particulier, mais particulièrement important, des données de type image sera examiné à la section II.C. Je vais ici présenter le cas des nuages et séquences de points, ainsi que le cas des données de similitude. J'avais par ailleurs rapidement examiné le cas des données symboliques dans (Courrieu, 1994b), et il existe une abondante littérature sur le sujet, mais je n'en parlerai pas ici.

Il arrive que les données se présentent sous la forme non pas de simples points (vecteurs), mais d'ensembles de plusieurs points d'un espace réel multidimensionnel. Lorsque les points sont naturellement ordonnés, il s'agit d'une séquence de points, et cela ne pose pas de problème de codage difficile. Dans ce cas, on aura simplement une matrice au lieu d'un unique vecteur. Il n'en va pas de même lorsque les points, a priori, ne sont pas ordonnés, et l'on parle dans ce cas de nuages de points. A titre d'exemple, supposons que l'on représente l'état d'un footballeur par deux coordonnées de position (x, y) sur le terrain, et un vecteur de déplacement (angle, vélocité), ce qui donne 4 coordonnées réelles par joueur. Supposons que chaque donnée à considérer consiste en une configuration de jeu d'une équipe de 11

footballeurs. A priori, les différents joueurs ne sont pas ordonnés, sauf si l'on considère les numéros de dossards, mais cela n'est pas pertinent dans la plupart des problèmes à traiter en psychologie du sport. Comment pouvons nous représenter une configuration de jeu sans induire un ordre arbitraire des joueurs, ce qui conduirait à considérer comme différentes des configurations identiques à une permutation des joueurs près? J'ai proposé deux solutions dans (Courrieu, 2001, article ci-joint), la seconde solution étant en fait plus particulièrement adaptée aux séquences de points. La première solution encode un nuage de points par les coefficients d'un polynôme multivarié particulier dont les zéros coïncident exactement avec les points du nuage, ce qui donne bien un codage invariant par permutation des points. Cette méthode de codage a été appliquée en psychologie du sport, dans la modélisation d'une tâche de discrimination de configurations de jeu de basket-ball par des joueurs experts ou novices (Baratgin, Courrieu, Ripoll, Laurent, & Ripoll, 2002; Ripoll, Baratgin, Laurent, Courrieu, & Ripoll, 2001).

Les données de similitude constituent un autre type de données fréquemment rencontrées en psychologie (matrices de confusion ou de comparaison par paires). Ces données sont souvent traitées par des méthodes de "multidimensional scaling" (Shepard, 1962), ce qui permet de représenter les objets comparés par des points d'un espace métrique, même lorsque les données de similitude ne sont pas vraiment réductibles à des mesures de distance exactes. A strictement parler, il s'agit de méthodes d'analyse de données, mais on peut aussi les considérer comme des méthodes de codage permettant de construire l'espace de sortie de modèles fonctionnels à partir de données de similitude empiriques (Baratgin et al., 2002). J'ai défini une méthode simple permettant de transformer, de façon strictement monotone, des mesures de similitude empiriques (pourvu qu'elles soient symétriques) en une métrique euclidienne, et de réaliser un multidimensional scaling direct sur ces mesures transformées (Courrieu, 2002, article ci-joint). Cette méthode, appelée "plongement monotone euclidien", a été appliquée dans l'étude de Courrieu et al. (2004) décrite dans la section II.A précédente. Au passage, l'élaboration de cette méthode m'a conduit à étendre la méthode de factorisation de Cholesky à des matrices singulières, ce qui s'est par la suite avéré utile dans d'autres contextes (voir section II.E).

Références

Baratgin, J., Courrieu, P., Ripoll, T., Laurent, L., & Ripoll, H. (2002). *Similarity Judgements of Basketball Game Configurations by Experts and Novices: A Model and Some Experimental Tests*. Unpublished manuscript (2002): http://www.ergos-perf.com/Docs/Model_PSE.pdf, 54 p.

Courrieu, P. (1994b). Connexionnisme et fonctions symboliques. *In* J.P. Caverni, C. George, & G. Politzer, *Raisonnements: Conjoncture et Prospective. Psychologie Française (numéro spécial), 39-2*, 231-236.

Courrieu, P. (2001). Two methods for encoding clusters. *Neural Networks, 14*, 175-183.

Courrieu, P. (2002). Straight monotonic embedding of data sets in Euclidean spaces. *Neural Networks, 15*, 1185-1196.

Courrieu, P., Farioli, F., Grainger, J (2004). Inverse discrimination time as a perceptual distance for alphabetic characters. *Visual Cognition, 11(7)*, 901-919.

Ripoll, H., Baratgin, J., Laurent, E., Courrieu, P., & Ripoll, T. (2001). Mechanisms underlying the activation of knowledge basis in identification of basketball play configurations by expert and non-expert players. *In* A. Papaioannou, M. Goudas, & Y. Theodorakis (Eds.), *In the Dawn of the New Millennium: Proceedings of the 10th World Congress of Sport Psychology, Skiathos, Greece, 28th May - 2nd June, Vol. 2*, pp. 283-285. Thessaloniki: Christodoulidi Publications.

Shepard, R.N. (1962). The analysis of proximities: multidimensional scaling with an unknown distance function (I & II). *Psychometrika, 27(2)*, 125-140, & *27(3)*, 219-246.

PERGAMON

Neural Networks 14 (2001) 175–183

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

Two methods for encoding clusters

Pierre Courrieu*

Laboratoire de Psychologie Cognitive, UMR CNRS 6561, Université de Provence, 29 avenue Robert Schuman, 13621 Aix-en-Provence cedex 1, France

Received 17 February 2000; accepted 28 October 2000

Abstract

This paper presents two methods for generating numerical codes representing clusters of R^n, while preserving various topological properties of data spaces. This is useful for networks whose input, or eventually output, consists of unordered sets of points. The first method is the best one from a theoretical point of view, while the second one is more usable for large clusters in practice.

Keywords: Cluster codes; Neural network input and output representations; Pseudo-sequences; Unordered sets; Encoding problem

1. Introduction

There are applications in which the input, or eventually the output, of a neural network is not a point but a cluster, that is an unordered, countable finite set of points of R^n. A frequent difficulty is to find an appropriate representation for such an input, given that there is no natural order of points in R^n for $n > 1$. Any permutation of a cluster's points is a priori equivalent to other ones, but distinct permutations provide distinct input vectors, and learning the equivalence of permutations is a very complex problem given that for a set of m points there are $m!$ possible permutations. One can always build an artificial order of points in R^n, for example using a Peano–Hilbert scanning. Unfortunately, this type of procedure never allows for preserving the topology of data spaces, and a small variation of data points can lead to a large variation in the representation. This results in major difficulties for learning regular functions on the space of such representations. A quite similar problem occurs with pseudo-sequences of points, that is when data points are ordered with respect to a natural variable, such as a time coordinate, but the underlying process (generating data points) is not actually sequential or is a random mixture of several sequences. In this case, small and possibly random variations of time coordinates can modify the order of points, resulting in large variations of the input representation.

The problem we address here is: find a mapping from the set of clusters of R^n to a set of real vectors (or matrices), referred to as 'cluster codes', such that (1) any cluster has a unique code, (2) distinct clusters have distinct codes, and (3) to any continuous movement of points in a cluster corresponds a continuous variation of code components. Such a mapping would be appropriate for encoding input clusters. Now, if the application requires that one can decode output cluster codes, we also need that (4) the mapping is inversible (which implies (2)). A mapping with the four above properties is in general an homeomorphism between the data space and the code space. However, one must take care that the code space can be a special part of R^k, which is not as simple as R^k itself. To date, we know of no solution which can be applied to all practical problems. However, there are various solutions for various families of problems. Two of these solutions are presented hereafter. Note that the problem addressed here is to find a finite exact representation of data, in a format which ensures that representations of various clusters can be compared. This is a problem different (and much less studied in the literature) from that of approximating data distribution by a density of probability (Husmeier & Taylor, 1998; Specht, 1990; Traven, 1991) or an attractor (Barnsley, 1993; Diaconis & Freedman, 1999). The problem is in some way related to the 'encoding problem' in neural self associators (Rumelhart, Hinton & Williams, 1986). However, such encoders require prior learning, and they cannot guarantee property (1), since a permutation of a cluster's points generally results in a change of the internal representation.

2. First method: polynomial encoding of clusters

For encoding a cluster in R or R^2 only, one can consider a real or complex polynomial of the form $P(z) = \prod_{j=1}^{m}(z - z_j)$,

* Tel.: +33-4-4295-3728; fax: +33-4-4220-5905.
E-mail address: crepco@newsup.univ-mrs.fr (P. Courrieu).

0893-6080/01/$ - see front matter
PII: S0893-6080(00)00096-4

which has exactly m real or complex roots, depending on the dimension. These roots are obviously the z_j's, which are the coordinates of the cluster's points. The product in real or complex algebra is commutative and associative, then $P(z)$ does not depend on the order in which the m roots are taken into account. As a consequence, $P(z)$ is unequivocally associated to the unordered set of roots (that is the cluster to be encoded). One can expand the polynomial $P(z)$, this expansion resulting in a sequence of $(m + 1)$ real or complex coefficients which is a possible cluster code satisfying all requirements (1)–(4). Unfortunately, as a consequence of the well known Frobenius theorem, one knows that there is no commutative and associative algebra in dimension larger than 2. There is an associative algebra in dimension 4 (quaternions algebra), but this algebra is not commutative. Then, the above sketched method cannot be extended for encoding clusters in dimension larger than 2.

The method described hereafter satisfies requirements (1)–(4) for any cluster of R^n, and for any n. So, it is theoretically a very general method, and it can be useful for theoretical purpose. However, there are restrictions in practice due to the fact that the generated code size increases rapidly with the cluster dimension (n) and the cluster size (m). Moreover, decoding is a complex operation, and when the code is only an approximation the result of decoding can be very approximate. Therefore, the author recommends the practical use of this method only for encoding small clusters in low dimension spaces, and when the application does not require decoding. A possible application field is the encoding of game configurations, and the method is well adapted for encoding configurations which contain several distinct clusters of various sizes and dimensions.

2.1. Data projection on the half unit sphere of R^{n+1}

Let $\{X_j \in R^n;\ 1 \le j \le m\}$ be the set of points of a cluster, where one assumes that no point has infinite coordinates. Choose an origin point $O \in R^n$ and a real $r > 0$ such that, for any j, $\|X_j - O\| \le r$, where $\|.\|$ denotes the Euclidean norm. Then the projection on the half unit sphere of R^{n+1} of the point $X_j = (x_{1j}, \ldots, x_{nj})$ is given by:

$$u_{ij} = (x_{ij} - o_{ij})/r, \qquad 1 \le i \le n,$$

and

$$u_{0j} = (1 - \sum_{i=1}^{n} u_{ij}^2)^{1/2}.$$

The vector $U_j = (u_{0j}, u_{1j}, \ldots, u_{nj})$ is such that $\|U_j\| = 1$, and the projection is obviously inversible by $x_{ij} = r u_{ij} + o_i$, $1 \le i \le n$.

One can choose standard O and r, and then project any cluster which is inside the sphere of R^n of center O and radius r. This sphere will be referred as the 'encoding sphere', and one can remark that projected coordinates are always continuous real functions of original coordinates inside the encoding sphere. One can also take for O the center of gravity of the cluster, which provides translation invariance, and/or take $r = s \max_{l \le j \le m} \|X_j - O\|$, $s \ge 1$ fixed, which provides scale invariance. In the latter case, any cluster of R^n is inside its own encoding sphere.

The main interest of the projection on the half unit sphere of R^{n+1} is that all vectors U have unit norm, and then:

$$\frac{1}{2}\|U - U_j\|^2 = 1 - U.U_j \qquad \text{and}$$

$$1 - U.U_j = 0 \Leftrightarrow U + U_j \Leftrightarrow X = X_j.$$

2.2. Polynomial associated with a cluster

Consider the polynomial defined by:

$$P(U) = \prod_{j=1}^{m} (1 - U.U_j)$$

where the U_j's are the projections of the cluster's points on the half unit sphere of R^{n+1}, U is the projection of the variable, and '.' denotes the inner (dot) product of vectors.

This is a real polynomial of degree m with $n + 1$ variables, and after Section 2.1, the set of roots of this polynomial is exactly the set of projections of the m cluster's points on the half unit sphere of R^{n+1}. Given that the (real) product is commutative and associative, this polynomial does not depend on the order in which data points are taken into account. Since the projection on the half unit sphere is inversible, $P(U)$ is unequivocally associated with the cluster in a given encoding sphere. Then, expanding $P(U)$, one obtains a sequence of coefficients which is a possible cluster code. The expansion of this algebraic polynomial is finite, and it is of the form:

$$P(U) = \sum_{k=0}^{m} \sum_{a_0 + a_1 + \cdots + a_n = k} (-1)^k c[a_0, a_1, \ldots, a_n] u_0^{a_0} u_1^{a_1} \cdots u_n^{a_n},$$

where the a_i's are integer exponents of the (projected) variable's coordinates. The real coefficients $c[.]$ are the code components, and they can be described in the following way. Let $P[m, k]$ denote the set of parts with k elements of the integer set $\{1, 2, \ldots, m\}$, and let $Q\{a_0, a_1, \ldots, a_n\}$ denote the set of all distinct permutations of the set which contains a_0 integer values 0, a_1 integer values 1, …, a_n integer values n. With $k = a_0 + a_1 + \cdots + a_n$, one has:

$$c[0, 0, \ldots 0] = 1,$$

$$c[a_0, a_1, \ldots, a_n] = \sum_{(j_1, \ldots j_k) \in P[m,k]} \sum_{(i_1, \ldots i_k) \in Q[a_0, \ldots, a_n]} \prod_{s=1}^{k} u_{i_s j_s}.$$

It should be noted that the coefficients of the expansion comprise only products and sums of the projected coordinates of the cluster's points, which guarantees that these coefficients are continuous functions of the coordinates of points, as long as clusters remain inside the encoding sphere.

Example. —With $n = 2$ and $m = 3$, one must expand the polynomial:

$$P(u_0,\ u_1,\ u_2) = (1 - u_{01}u_0 - u_{11}u_1 - u_{21}u_2)\ (1 - u_{02}u_0 - u_{12}u_1 - u_{22}u_2)\ (1 - u_{03}u_0 - u_{13}u_1 - u_{23}u_2).$$

The expansion by hand is quite tedious but easy, and one obtains:

$$\begin{aligned}P(u_0,u_1,u_2) = 1 &- (u_{01} + u_{02} + u_{03})u_0 - (u_{11} + u_{12} + u_{13})u_1 - (u_{21} + u_{22} + u_{23})u_2 + (u_{01}u_{12} + u_{11}u_{02} + u_{01}u_{13} + u_{11}u_{03} + u_{02}u_{13} + u_{12}u_{03})u_0u_1 + (u_{01}u_{22} + u_{21}u_{02} + u_{01}u_{23} + u_{21}u_{03} + u_{02}u_{23} + u_{22}u_{03})u_0u_2 + (u_{11}u_{22} + u_{21}u_{12} + u_{11}u_{23} + u_{21}u_{13} + u_{12}u_{23} + u_{22}u_{13})u_1u_2 + (u_{01}u_{02} + u_{01}u_{03} + u_{02}u_{03})u_0^2 \\ &+ (u_{11}u_{12} + u_{11}u_{13} + u_{12}u_{13})u_1^2 + (u_{21}u_{22} + u_{21}u_{23} + u_{22}u_{23})u_2^2 - (u_{01}u_{12}u_{23} + u_{11}u_{02}u_{23} + u_{01}u_{22}u_{13} + u_{11}u_{22}u_{03} + u_{21}u_{02}u_{13} + u_{21}u_{12}u_{03})u_0u_1u_2 - (u_{01}u_{02}u_{13} + u_{01}u_{12}u_{03} + u_{11}u_{02}u_{03})u_0^2u_1 - (u_{01}u_{02}u_{23} + u_{01}u_{22}u_{03} + u_{21}u_{02}u_{03})u_0^2u_2 - (u_{11}u_{12}u_{23} + u_{11}u_{22}u_{13} + u_{21}u_{12}u_{13})u_1^2u_2 - (u_{01}u_{12}u_{13} + u_{11}u_{02}u_{13} + u_{11}u_{12}u_{03})u_0u_1^2 - (u_{01}u_{22}u_{23} + u_{21}u_{02}u_{23} + u_{21}u_{22}u_{03})u_0u_2^2 - (u_{11}u_{22}u_{23} + u_{21}u_{12}u_{23} + u_{21}u_{22}u_{13})u_1u_2^2 - u_{01}u_{02}u_{03}u_0^3 \\ &- u_{11}u_{12}u_{13}u_1^3 - u_{21}u_{22}u_{23}u_2^3.\end{aligned}$$

Fortunately, there is another way of computing the coefficients of the expansion, as we shall see in the next section.

2.3. Encoding procedure

2.3.1. Computing the coefficients

The following procedure provides the same coefficients than those of Section 2.2, however, its recurrent form makes it much easier to implement on standard computers. The encoding procedure is presented in pseudo-Pascal notation:

```
c[0, 0, …, 0]: = 1;
for j: = 1 to m do
  for k: = j downto 1 do
    for a0: = 0 to k do
    for a1: = 0 to (k − a0) do
    …
    for an−1: = 0 to (k − Σ_{i=0}^{n−2} ai) do
      begin
      an: = k − Σ_{i=0}^{n−1} ai;
      if k = j then c[a0, a1, …, an]: = 0;
      c[a0, a1, …, an] := c[a0, a1, …, an]
        + Σ_{ai>0} uij c[a0, …, ai − 1, …, an];
      end;
```

Note. For readers unfamiliar with pseudo-Pascal notation, one can say that this is simply English mixed with mathematical notations. The semicolon is the statement separator, while 'x: = y' means that x takes the value of the expression y. The sequence of statements between 'begin' and 'end' must be repeated in each iteration governed by the 'for' statements.

2.3.2. Addressing the coefficients in a vector

For clarity in the procedure (2.3.1) statement, we used a multidimensional array notation for $c[.]$, where we have in fact to obtain a vector. Moreover, the size of this multidimensional array would be of $(m + 1)^{n+1}$ real numbers, while the total number of coefficients of the expansion is only $K_{n+2}^m = \binom{m+n+1}{[m]}$. The combinatorial K_i^j can be recursively computed using the relations:

$$K_i^0 = K_1^j = 1,$$

and

$$K_i^j = K_{i-1}^j + K_i^{j-1}.$$

Using this combinatorial, one can compactly address the coefficients in a one dimension array as follows:

$$k := a_0 + a_1 + \cdots + a_n;$$

$$\text{index}(a_0, a_1, \ldots, a_n) := \{k > 0\}K_{n+2}^{k-1} + \{a_0 > 0\}\sum_{j=0}^{a_0-1} K_n^{k-j} + \{a_1 > 0\}\sum_{j=0}^{a_1-1} K_{n-1}^{k-a_0-j} + \{a_2 > 0\}\sum_{j=0}^{a_2-1} K_{n-2}^{k-a_0-a_1-j} + \cdots + \{a_{n-1} > 0\}\sum_{j=0}^{a_{n-1}-1} K_1^{k-a_0-\cdots-a_{n-2}-j},$$

where we note that $\{\text{false}\} = 0$, and $\{\text{true}\} = 1$.

In practical implementations, the multi-index $[a_0, a_1, \ldots, a_n]$ which appears in Section 2.3.1 must be replaced by the scalar integer function $\text{index}(a_0, a_1, \ldots, a_n)$ defined above. Indexed this way, the coefficients are ordered

Table 1
Mean distances (and standard deviations) in the space of codes ($d(C, C')$, first method), and in the space of data (Hausdorff), as a function of the modification scale δ

δ	$d(C, C')$		Hausdorff	
0.0001	000.11	(0.04)	0.0002	(0.0000)
0.001	001.10	(0.43)	0.0015	(0.0001)
0.01	010.71	(4.12)	0.0154	(0.0009)
0.02	021.33	(8.27)	0.0309	(0.0019)
0.04	043.64	(17.3)	0.0615	(0.0041)
0.08	087.96	(36.0)	0.1232	(0.0086)
0.16	163.79	(57.6)	0.2450	(0.0182)
0.32	331.29	(129)	0.4814	(0.0375)
0.64	602.82	(262)	0.8909	(0.0783)
1.28	886.72	(243)	1.2303	(0.1692)

in increasing order of the total degree of the terms they weight. Then the total degree corresponding to the last non zero coefficient provides the number m of points of the cluster. If the index of this coefficient is larger than K_{n+2}^{k-1} but no larger than K_{n+2}^{k} then the degree is k, and $m = k$. One can theoretically use an infinite vector for encoding any cluster of R^n, the unused components being set to zero.

2.4. Decoding

Properties (1) and (2) of the Introduction imply that there is a bijection between the set of clusters of R^n, inside a given encoding sphere, and the set of exact cluster codes. This guarantees that any exact cluster code is, in some way, decodable with an exact result. Now, in practice, codes to be decoded are in general approximations which are not necessarily exact cluster codes, since the set of exact cluster codes is only a subset of the set of real vectors. We sketch hereafter a method which theoretically allows for approximately decoding a real vector into a cluster. Of course, an exact result is accessible if the vector is an exact cluster code.

First one can find the number of points (m) from the last non zero coefficient of the vector V to be decoded (see Section 2.3.2). Let Q be the set of clusters of size m in a given encoding sphere, consider a cluster $q \in Q$, and denote $C(q)$ the cluster code associated with q by the encoding method (Sections 2.1 and 2.3). Then a solution, denoted q^*, to the decoding problem is provided by solving the following global optimization problem:

$$q^* = \arg\min_{q \in Q} \|C(q) - V\|.$$

Since the search domain is bounded (encoding sphere) and the functional to be minimized is continuous (as a consequence of property (3)), there are global optimization algorithms whose convergence to a global minimizer is guaranteed (Courrieu, 1997, Theorem 1; Ingber & Rosen, 1992; Solis & Wets, 1981). This theoretically guarantees the solvability of the decoding problem. However, it was observed in practice that the solving time increases very fast with the dimension (n) and the size (m) of clusters. So, the above described decoding method cannot reasonably be recommended as a practical one, except for very small problems.

2.5. Computational test

We know that the encoding preserves basic topological properties of the data space. The object of the computational test is to see if metric relations are also preserved, and if they correctly reflect variations in data generating processes. For the computational test, we generated 20 uniformly random clusters of $m = 12$ points in $[-1, 1]^4$ (i.e. $n = 4$). Ten modified versions of each cluster were generated by adding a random quantity, uniformly sampled in $[-\delta, \delta]$, to each coordinate of each point. The modification scale δ was experimentally varied from 0.0001 to 1.28, and each added quantity was eventually resampled until the modified coordinate was in $[-1, 1]$. The distance between the code C of a cluster and the code C' of a modified version was defined as the Euclidean distance of codes: $d(C, C') = \|C - C'\|$. We also computed, for comparison, the Hausdorff distance (associated to the Euclidean distance in R^n) between clusters in the data space (see Barnsley, 1993). Let q and q' be the two clusters to be compared, then their Hausdorff distance is defined by:

$$h(q,\ q') = \max(\max_{X \in q} \min_{Y \in q'} \| X - Y \|,$$

$$\max_{Y \in q'} \min_{X \in q} \| Y - X \|).$$

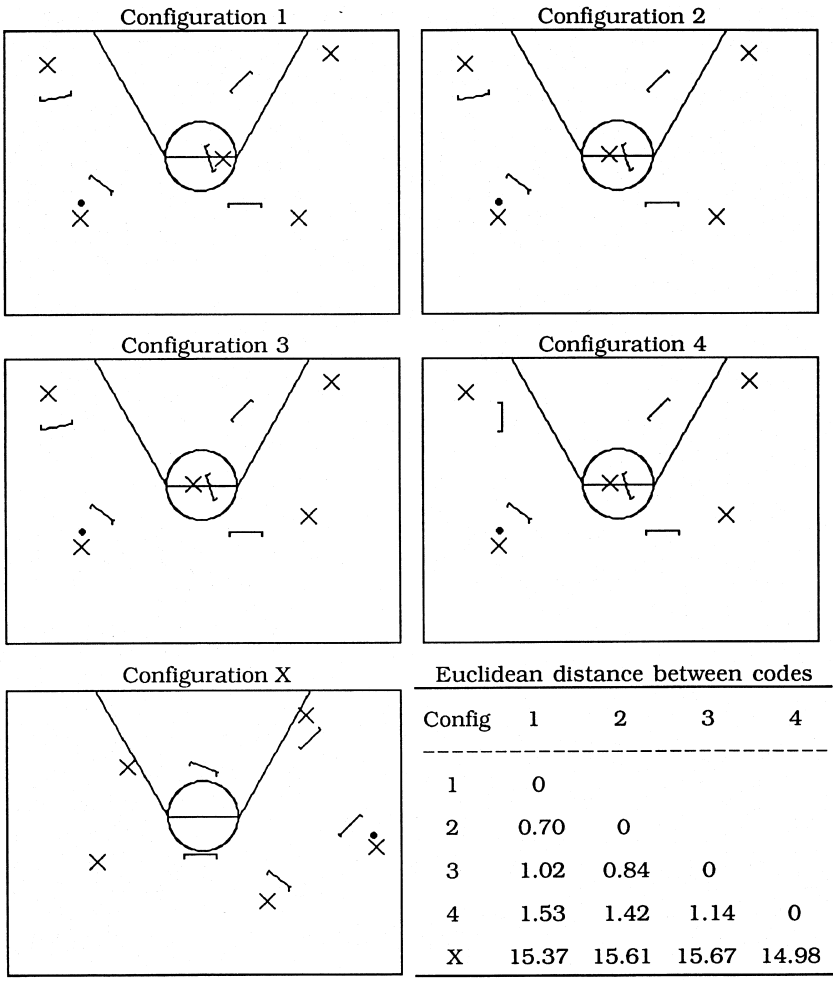
Mean distances (with standard deviations on 19 degrees of freedom) are reported in Table 1.

As one can see in Table 1, the distance of codes is, on the average, a monotone increasing function of the modification scale, and is approximately proportional to the Hausdorff distance ($d(C, C') \simeq 700$ Hausdorff). Hence, it seems that the encoding method generates a code space whose metric properties are compatible with those of the data space.

Out of curiosity, the decoding problem was tested on exact codes of very small clusters, $1 \leq m \leq 4$, in $[-1, 1]^4$. Decoding problems were exactly solved using the so called 'Hyperbell algorithm' (Courrieu, 1997). It turned out that the number of steps of the algorithm for finding exact solutions was about 455×10^m, where each step required encoding a cluster. Clearly, this is not a practical solution.

2.6. An example of application

The above described encoding method was recently used in sport's science area for encoding and comparing basketball play configurations (Courrieu, Ripoll, Ripoll, Baratgin & Laurent, submitted; Baratgin, Ripoll, Ripoll, Courrieu & Laurent, submitted). Basketball play configurations can be schematized as illustrated in Fig. 1. In this type of representation, ground marks provide a reference, the dot stands for the ball, crosses represent attackers, and segments represent



Euclidean distance between codes

Config	1	2	3	4
1	0			
2	0.70	0		
3	1.02	0.84	0	
4	1.53	1.42	1.14	0
X	15.37	15.61	15.67	14.98

Fig. 1. Five schematized basketball play configurations and the Euclidean distances between their associated codes (first method applied by Courrieu et al., submitted; Baratgin et al., submitted).

defenders with their orientation. Coaches commonly use such representations for training basketball teams. A play configuration can be described as a set of three clusters: the ball which is a cluster of one point in the plane, the attacker team, which is a cluster of five points in the plane, and the defender team, which is a cluster of five points in a four dimension space since each defender is represented by his position in the plane (two coordinates) and his orientation (two more coordinates). A convenient way of representing a defender consists of taking his "left hand" and "right hand" plane coordinates. Then, it suffices to encode each of the three clusters as described above, and to concatenate the resulting codes, in fixed order, in a global vector (312 real coefficients). This vector unequivocally represents the multicluster play configuration. Fig. 1 shows four close configurations (numbered from 1 to 4), each of them being obtained from the previous one by changing the position of one player, while the fifth configuration (X) is very different from the other ones. Euclidean distances between codes associated to these configurations were computed and they are shown at the bottom-right of Fig. 1. As one can see, these distances reflect the differences between play configurations in a quite natural way. More advanced results on this topic are reported in Courrieu et al. (submitted) and Baratgin et al. (submitted).

3. Second method: cluster codes depending on a separating variable

This method satisfies requirements (1) and (3) for any cluster, and requirements (2) and (4) in most cases, but not all. Its advantages are that the generated code is concise (nm real coefficients), and that decoding, when possible, is very simple. Moreover, a simple inspection of a code vector immediately shows whether or not decoding is allowed. This method is particularly adapted for encoding and decoding pseudo-sequences of points. A possible application field is the encoding of sets of seismic events, such events being rarely simultaneous if the time is measured with relatively high precision. Another possible application field is the

analysis of sequences of regularly sampled physical measures (e.g. acoustical or electrophysiological measures).

3.1. The one variable case

The one variable case is very simple since points of R are naturally ordered. Then it is sufficient to order data points in increasing order (or in fact any fixed order) of their values to obtain a cluster code which trivially satisfies requirements (1)–(4) in all cases. In particular, decoding is simply the identity mapping. In the case where all points of a cluster of R have distinct coordinates, we say that the variable separates the points of this cluster. The set of clusters for which the variable is not separating (i.e. clusters which have repeated points) is a set of zero measure. To be convinced of this, note that for any cluster size $m > 1$, the set of clusters with repeated points is the union of hyperplanes of R^m whose equations are $x_i - x_j = 0,\ 1 \le i < j \le m$. This is obviously a zero measure subset of R^m.

3.2. Encoding clusters of R^n

Let t be the first coordinate of R^n, and consider a cluster of m points $\{X_j = (t_j, x_{2j}, \ldots, x_{nj});\ 1 \le j \le m\ \}$ in R^n, where points have been ordered in such a way that $t_j \le t_{j+1}$, $1 \le j \le (m-1)$. Let $g(t)$ be a stricly monotonic continuous function, and form the square matrix $S = (s_{ij})$, $1 \le i \le m$, $1 \le j \le m$, where $s_{ij} = [g(t_j)]^{i-1}$. All coefficients in the first row of the matrix S have value $[g(.)]^0 = 1$, and remaining rows are successive integer powers of the second row coefficients. A matrix with such a structure has a determinant, called 'Vandermonde determinant', which is:

$$\det S = \prod_{1 \le j < k \le m} (s_{2k} - s_{2j}).$$

It is clear that this determinant is different from zero if and only if all coefficients in the second row have different values. Given that g is strictly monotonic, one obtains:

$$\det S \ne 0 \Leftrightarrow t_j \ne t_k, 1 \le j < k \le m.$$

Now, consider the vectors $T = (t_j)$, $1 \le j \le m$, and $V_i = (x_{ij})$, $1 \le j \le m$, $2 \le i \le n$. The $m \times n$ matrix C whose first column is $C_1 = T$, and whose ith column is $C_i = SV_i$, $2 \le i \le n$, is a code which satisfies requirements (1) and (3) in all cases. The first column encodes only one variable (t), then after Section 3.1, the conventional increasing order of C_1 components is appropriate. On the other hand, the columns C_i, $i > 1$, do not depend on any ordering of points: let p be an $m \times m$ permutation matrix, then a permutation of the points gives $C_i = Spp'V_i = SV_i$, where p' denotes the transposed of p, which is equal to its inverse given that any permutation matrix is orthogonal. This guarantees the property (1), while the property (3) results from the fact that g is continuous and the encoding only implies products and sums. Moreover, if $\det S \ne 0$, then S is inversible and $V_i = S^{-1}C_i$, $2 \le i \le n$, where S, and then S^{-1}, can be computed from the column C_1. In other words, this encoding satisfies requirements (2) and (4) if and only if the t variable separates the cluster's points. After Section 3.1, the set of clusters for which t is not separating is a set of zero measure, however it is not an empty set. Now, a simple inspection of the column C_1 of the code immediately shows whether it contains equal coefficients, that is, if decoding is allowed or not. When a code was obtained as the output of an approximation process (neural network), it can happen that the components of C_1 are not in increasing order. This is not problematic, since the set of points provided by decoding will be the same, in permuted order (since $(Sp)^{-1}\ C_i = p'S^{-1}C_i = p'V_i$). However, if comparisons between codes are required, then it is necessary to reorder C_1 (and only C_1!).

The code has another property which can be useful in certain applications. Let V be the matrix whose columns are the V_i's, let W be the matrix whose columns are the C_i's, $2 \le i \le n$, and let B be any $(n-1) \times (n-1)$ real matrix. Then, the cluster whose coordinate matrix is (T, VB) has the code matrix (T, WB), since the matrix S only depends on T, and $W = SV$ implies that $SVB = WB$. In other words, applying a linear transform to the data coordinates, without changing the first coordinate, results in applying the same transform to the code.

3.3. Practical choice of g(t)

After Section 3.2, the only theoretical requirements for the g function are that the function is continuous and strictly monotonic. Now, depending on this function, successive powers which are used in the matrix S can rapidly go to very high or very low values. Hence, one must choose the g function in such a way that none of the coefficients of S tends to infinity as the power (i.e. the row number) increases. This means that $-1 \le g(t) \le 1$, for all possible values of t. A solution is $g(t) = 2F(t) - 1$, where $F(t)$ is a continuous strictly monotonic approximation of the cumulative probability function of t. Another advantage of this solution is that the distribution of $g(t)$ is approximately uniform in $[-1, 1]$. Note that one can as well choose $g(t) = F(t)$, and then the variation interval is $[0, 1]$.

3.4. Summary of the method

For the whole application:

- choose one of the variables, which is the most probably separating, as the variable t,
- determine a function $g(t) = 2F(t) - 1$, or $g(t) = F(t)$, where $F(t)$ is a strictly monotonic continuous approximation of the cumulative probability function of t.

Encoding a cluster:

- form the matrix $S = (s_{ij})$, $s_{ij} = [g(t_j)]^{i-1}$,
- the first column of the code is $C_1 = T$, in increasing order of t values,
- the ith column of the code is $C_i = SV_i$, $2 \le i \le n$.

Decoding a code matrix:

- $T = C_1$,

- if all components of T have distinct values then the code is decodable, and one forms the matrix S as for the encoding,
- $V_i = S^{-1}C_i$, $2 \le i \le n$.

Note: If certain values of t are very close, then the matrix S is ill-conditioned, which implies that a small approximation error of C_i can result in a quite large approximation error in the decoded V_i.

3.5. Computational test

Given that the above described encoding satisfies requirements (1) and (3) in all cases, various topological properties of data spaces are preserved in code spaces. However, since this encoding does not satisfy requirement (2) everywhere, there is a potential drawback which is that two distinct clusters can have the same code, and combined with (3), there are quite different clusters which have close codes. Now, since the set of unseparated clusters is of zero measure, one can hope that the drawback is limited and has statistically weak effect. For the computational test, we generated 20 uniformly random clusters of $m = 128$ points in $[-1, 1]^{16}$ (i.e. $n = 16$). Ten modified versions of each cluster were generated by adding a random quantity, uniformly sampled in $[-\delta, \delta]$, to each coordinate of each point. The modification scale δ was experimentally varied from 0.0001 to 1.28, and each added quantity was eventually resampled until the modified coordinate was in $[-1, 1]$. The distance between the code C of a cluster and the code C' of a modified version was defined as the Euclidean matricial norm of the difference of codes: $d(C, C') = \|C - C'\|$, which is equivalent to the Euclidean distance in R^{nm}. We also computed, for comparison, the Hausdorff distance between clusters in the data space. Mean distances (and standard deviations) are reported in Table 2.

As one can see in Table 2, the distance of codes is, on the average, a monotone increasing function of the modification scale, as is the Hausdorff distance (however the relations are not linear). Hence, it seems that the encoding method has reasonable properties for practical use. Now, it is clear that when one chooses learning examples for a neural algorithm, it is desirable to avoid those examples of clusters whose first coordinate is not separating. When decoding an approximated output code, one has to consider with caution clusters whose points are very irregularly spaced on the first coordinate (in fact, what is important is the spacing of $g(t)$ values).

3.6. An example of pseudo-sequence problem

One can suspect that many natural processes are in fact pseudo-sequences, that is sequences of randomly ordered events or random mixtures of several distinct processes. In order to show how the above cluster encoding method can help to solve pseudo-sequence problems, we take now the example of sequences made of a random mixture of two independent processes: a process generated by the well-known logistic map, and a process generated by the so called '*kappa* map' (Husmeier & Taylor, 1998). A logistic process will be defined by:

$$y(t' + 1) = \alpha y(t')[1 - y(t')], y(0) = 0.5, \alpha \in [3, 4],$$

while a kappa process will be defined by:

$$z(t'' + 1) = 1 - z(t'')^{\kappa}, z(0) = 0.5, \kappa \in [0.5, 1.25].$$

Taking $t = t' + t''$, one can define a pseudo-sequence by:

$$x(t + 1) = y(t' + 1) \text{ with probability } p(t', t'', \Delta_{\max}),$$

$$x(t + 1) = z(t'' + 1) \text{ with probability } 1 - p(t', t'', \Delta_{\max}),$$

where the probability of processes is given by:

$$p(t', t'', \Delta_{\max}) = \max(0, \min(1, 0.5 + 0.5(t'' - t')/\Delta_{\max})),$$

where $\Delta_{\max} > 0$ is an integer number.

Note that if $\Delta_{\max}$ tends to infinity then $p = 0.5$ at any step, independently of previous events, while taking a low value for $\Delta_{\max}$ constrains t' and t'' to remain close to each other. This is a way of limiting the random variability of the mixture. However, the global probability of each of the two processes is always 0.5.

A sequence can be considered as a cluster $\{(t, x(t)); 1 \le t \le m\}$ in R^2, where t is obviously a separating variable uniformly distributed in the interval $[1, m]$. Then one can take $g(t) = t/m$ and apply the cluster encoding method. In this study, we take $m = 200$, and we can draw the sequences given that they are in R^2. Note, however, that the same methodology can as well be applied with higher dimension (vector sequences), but this would be hard to draw.

Each of the two component processes of a sequence depends on a parameter, and each pair of parameters (α, κ) corresponds to a family of random mixtures of the same processes, while changing the parameters leads to another family (see Fig. 2). The question is: does the cluster code reflect the family the sequence belongs to? If this is the case, then the distance between codes of sequences belonging to the same family must be lower than the distance between codes of sequences belonging to distinct families,

Table 2
Mean distances (and standard deviations) in the space of codes ($d(C, C')$, second method), and in the space of data (Hausdorff), as a function of the modification scale δ

δ	$d(C, C')$		Hausdorff	
0.0001	00.09	(0.06)	0.0003	(0.0000)
0.001	00.89	(0.47)	0.0029	(0.0001)
0.01	06.31	(2.78)	0.0290	(0.0009)
0.02	10.84	(5.18)	0.0580	(0.0018)
0.04	16.30	(5.09)	0.1157	(0.0029)
0.08	23.72	(7.48)	0.2317	(0.0063)
0.16	29.26	(7.05)	0.4610	(0.0105)
0.32	36.33	(7.78)	0.9187	(0.0253)
0.64	43.57	(5.73)	1.8009	(0.0607)
1.28	57.36	(6.54)	2.5288	(0.0908)

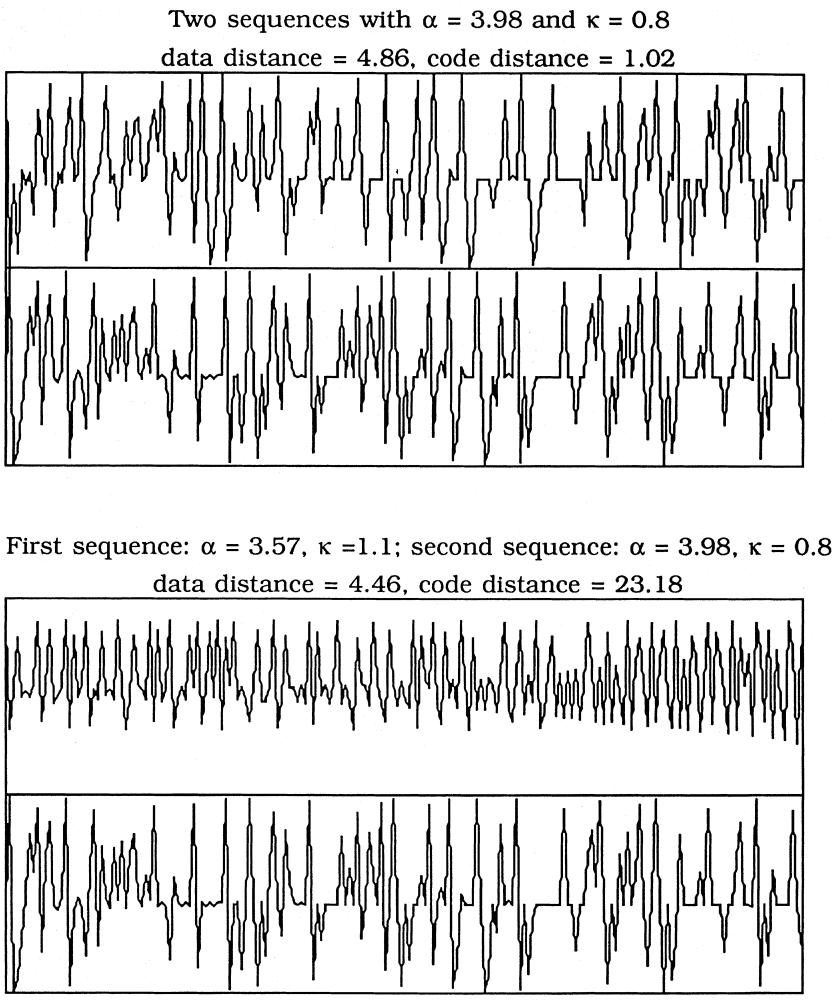


Fig. 2. Two pairs of sequences generated by a random mixture of independent logistic and kappa processes ($\Delta_{max} = 200$). Codes were computed using the second method.

and the encoding method can be used for building robust sequence classifiers (for example). Of course, cluster codes are useful only if they reflect the component processes better than data themselves.

A code distance can be defined, as in Section 3.5, by the Euclidean norm of the difference of codes, while a data distance can be defined in the same way, replacing the code components by the (sequentially ordered) data points. Note that, in the present case, distances do not depend on the values of t, since these values are the same for all sequences.

A computational experiment was designed as follows. Four values of Δ_{max} were selected (2, 3, 4, and 200). For each of these values, a uniform random sample of 80 pairs of parameters (α, κ) was generated in the sampling intervals of these parameters. 40 of these pairs were used for generating 40 pairs of sequences ($m = 200$), the two sequences in a pair being distinct random mixtures of the same two processes. The remaining 40 pairs of parameters were randomly associated to the previous ones for generating 40 other pairs of sequences, the two sequences in a pair being random mixtures of (randomly) distinct processes. The data distance and the code distance were computed for each pair of sequences. Mean distances (and standard deviations on 39 degrees of freedom) for all experimental conditions are reported in Table 3. In addition, Student t tests (with 78 degrees of freedom) were used for comparing the distances between sequences generated with similar processes to the distances between sequences generated with distinct processes. Statistical significance was tested according to the usual decision threshold $p < 0.05$ (while 'n.s.' means that the tested difference is statistically non significant).

As one can see in Table 3, with $\Delta_{max} = 2$, the data distance was just significantly lower for similar processes than for distinct processes, while for higher values of Δ_{max}, the data distance clearly does not allow for detecting the similarity of processes. Contrasting with this result, the code distance was always largely and significantly lower for similar processes than for distinct processes. Hence, the proposed encoding method clearly appears as an efficient tool for solving pseudo-sequence problems. Moreover, this

Table 3
Mean data distance and code distance (with standard deviation) between sequences made of distinct random mixtures of independent logistic and kappa processes. The component processes of two compared sequences can be similar or randomly distinct. When processes are similar, lowering Δ_{max} results in bringing the ranks of similar values of the two sequences nearer

Δ_{max}	2	3	4	200
Data distance				
Similar (α, κ)	3.27 (0.90)	4.15 (1.15)	3.79 (1.10)	3.55 (0.99)
Distinct (α, κ)	3.66 (0.75)	4.10 (0.82)	4.07 (0.88)	3.80 (0.96)
Student $t(78)$, significance	2.10, $p < 0.05$	0.21, n.s.	1.23, n.s.	1.12, n.s.
Code distance				
Similar (α, κ)	2.66 (2.63)	4.06 (3.60)	3.15 (2.98)	3.76 (1.73)
Distinct (α, κ)	9.24 (5.33)	9.57 (6.32)	10.07 (6.03)	8.89 (4.52)
Student $t(78)$, significance	7.00, $p < 0.001$	4.79, $p < 0.001$	6.52, $p < 0.001$	6.70, $p < 0.001$

result suggests that the tool allows for robust comparisons of sequences in general.

4. Conclusion

Two methods for encoding clusters were presented. In applications, cluster codes can be used for comparisons or as arguments of various functions (Spline functions, Radial Basis functions, etc.), and decodable codes can also be used as output of approximation processes. The first method is the best one from a theoretical point of view since it satisfies all requirements specified in the Introduction, for all clusters. This method has limited applications in practice since, due to its relative complexity, it is reserved for small clusters in low dimension spaces, and applications which do not require decoding. However, it is well adapted for encoding multicluster configurations, and an example of application in sport's science area was provided. The second method has some theoretical drawbacks, but it is very simple and concise, which makes it usable in most practical cases. There is no special limitation concerning the size or the dimension of clusters and practical examples were provided. A reviewer asked whether the presented coding schemes are local or global. The answer is that the two coding schemes have the unusual property of being simultaneously local and global. They are global in that sense that any code component depends on all points of the cluster. On the other hand, they are local in that sense that any point of the cluster can be exactly deduced from the code. Finally, one can note that there are probably other possible approaches to the problem of encoding clusters, in particular in the field of algebraic geometry, and this is a matter for further research.

Acknowledgements

This work was partially supported by a grant from Ministère de l'Education Nationale, de la Recherche et de la Technologie—ACI 'Cognitique' (1999, #90).

References

Baratgin, J., Ripoll, T., Ripoll, H., Courrieu, P. & Laurent, E., submitted. Similarity judgment of basketball play configurations by experts and novices. Part 2: first experimental tests.

Barnsley, M. F. (Academic PreP). *Fractals Everywhere*, (2nd ed.).

Courrieu, P. (1997). The Hyperbell algorithm for global optimization: a random walk using Cauchy densities. *Journal of Global Optimization*, *10*, 37–55.

Courrieu, P., Ripoll, T., Ripoll, H., Baratgin, J. & Laurent, E., submitted. Similarity judgment of basketball play configurations by experts and novices. Part 1: theoretical approach.

Diaconis, P., & Freedman, D. (1999). Iterated random functions. *SIAM Review*, *41*, 45–76.

Husmeier, D., & Taylor, J. G. (1998). Neural networks for predicting conditional probability densities: improved training scheme combining EM and RVFL. *Neural Networks*, *11*, 89–116.

Ingber, L., & Rosen, B. (1992). Genetic algorithms and very fast simulated reannealing: a comparison. *Mathl. Comput. Modelling*, *16*, 87–100.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland, *Parallel Distributed Processing*, vol. 1. Cambridge, MA: MIT Press.

Solis, F. J., & Wets, R. J. -B. (1981). Minimization by random search techniques. *Mathematics of Operations Research*, *6*, 19–30.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks*, *3*, 109–118.

Traven, H. G. C. (1991). A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density functions. *IEEE Transactions on Neural Networks*, *2*, 366–377.


PERGAMON

Neural Networks 15 (2002) 1185–1196

Neural Networks

www.elsevier.com/locate/neunet

Straight monotonic embedding of data sets in Euclidean spaces

Pierre Courrieu*

Laboratoire de Psychologie Cognitive, CNRS-UMR 6146, Université de Provence, 29 avenue Robert Schuman, 13621 Aix-en-Provence Cedex 1, France

Received 12 July 2001; accepted 15 May 2002

Abstract

This paper presents a fast incremental algorithm for embedding data sets belonging to various topological spaces in Euclidean spaces. This is useful for networks whose input consists of non-Euclidean (possibly non-numerical) data, for the on-line computation of spatial maps in autonomous agent navigation problems, and for building internal representations from empirical similarity data. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Non-Euclidean data sets; Monotonic embedding; Multidimensional scaling; Singular Cholesky factor; Spatial maps

1. Introduction

The main concern of this paper is the spatial representation in Euclidean spaces of data sets belonging to metric or non-metric topological spaces. In a previous study (Courrieu, 2001), the particular case of spaces of clusters was treated, a cluster being defined as a finite unordered set of points. Two methods were proposed for encoding clusters in an Euclidean code space, while preserving various topological properties of data spaces. The present work concerns other types of data spaces, and is related to a well known field in data analysis, i.e. 'multidimensional scaling' (Cox & Cox, 1994; Kruskal, 1964a,b; Shepard, 1962a,b; Young & Torgerson, 1967). Usual multidimensional scaling algorithms search for a set of points of R^n whose mutual distances (for a given metric) approximate a given 'dissimilarity' measure between the corresponding elements of the data space, while the dissimilarity measure is not necessarily a metric. Most of these algorithms attempt to minimize an error criterion using a gradient descent type procedure, which gives rise to quite slow computation and frequently causes the search process to be trapped into local minima. A recent variant of gradient descent allows for improving the performance of these methods (Demartines & Hérault, 1997). From a mathematical point of view, multidimensional scaling is closely related to the well known 'isometric embedding' problem (Blumenthal, 1936; Fréchet, 1910; Micchelli, 1986; Schoenberg, 1937, 1938). This connection will be widely exploited in this paper.

The Neural Networks community is concerned with this problematic in several ways. First, neural computation research has developped powerful methods for approximating continuous mappings on compact subsets of Euclidean spaces, from finite sets of data points. The Euclidean nature of the support space is particularly clear for Radial Basis Function Networks and Radial Spline systems (Girosi & Poggio, 1990; Poggio & Girosi, 1990), since Radial Basis Functions are functions of an Euclidean distance on the input space. Moreover, fundamental properties of networks, such as their approximation and regularization capabilities, critically depend on the hypothesis concerning the support space. However, as concerns practical applications, one can note that available data spaces are frequently not Euclidean, and even that they are frequently not real vectorial spaces. This results in difficulties for engineers faced with implementing artificial neural network applications, since they must transform given data spaces into real vectorial spaces empirically, and in general without any guarantee concerning the relevance of this operation. Certain numerical data sets can be artificially considered as Euclidean spaces, however, using an Euclidean metric on such data sets frequently leads to very disappointing results. A well known example of this is the case of numerical time series: one can always consider time series as vectors and compute an Euclidean distance between two vectors, however, this is rarely relevant. Dissimilarity measures provided by Dynamic Programming methods ('elastic template matching') are in general much more appropriate, however, the

* Corresponding author. Tel.: +33-4-42-95-37-28; fax: +33-4-42-20-59-05.

E-mail address: courrieu@up.univ-mrs.fr (P. Courrieu).

PII: S0893-6080(02)00091-6

resulting space is not Euclidean, a priori, and in fact it is not necessarily metric if the measure used is not a true metric (Okochi & Sakai, 1982; Vinstuk, 1968). There are also data spaces which are not numerical, such as spaces of symbol strings, for example. However, there are well-known methods for defining dissimilarity measures on symbol string spaces, these measures being called 'edition distances' (Lowrance & Wagner, 1975; Wagner & Fischer, 1974). One can find many other examples of data spaces which are not Euclidean, or even not numerical, but on which one can define a dissimilarity measure which has at least some properies of a metric. In general, triangle inequality is the most difficult property to obtain. Now, assume that one can define a simple continuous strictly increasing transform of a dissimilarity measure, and the useful part of the data space with this monotonically transformed dissimilarity measure is a metric set isometrically embeddable in an Euclidean space (this is a 'monotonic embedding'). It is clear that this would help to solve a number of practical problems of data encoding for neural network applications. Another field of interest concerns autonomous agents (alive or artificial) which must locate objects and themselves in a given environment, using approximative evaluations of distances for building a 'spatial cognitive map'. The interest of multidimensional scaling in this navigational context is obvious, however, this requires on line computation, that is simple and fast algorithms. The problem is clear for robotic applications, while the principle of an on line computation of spatial cognitive maps also seems relevant from a neurobiological point of view if one consider the behavior of hippocampal place cells (Cressant, Muller, & Poucet, 1997, 1999; Muller, 1996; O'Keefe & Nadel, 1978; Poucet, Save, & Lenck-Santini, 2000). Finally, multidimensional scaling methods were widely used in the area of psychological science for approximating the so-called 'psychological spaces' from empirical subjective similarity data (Nosofsky, 1992; Shepard, 1987). This can be viewed as a generalization of spatial cognitive maps to more abstract spaces.

In this paper, we present a particular approach of multidimensional scaling which, in a sense, is less general than usual multidimensional scaling methods, since it does not allow for embedding data sets in any real metric space. However, the method developed here is particularly appropriate to neural network applications since it allows for an exact monotonic embedding of data sets in Euclidean spaces, using a straight, fast and incremental algorithm. The incremental character of the algorithm implies that one can embed any new item without recomputing or modifying the embedding of previously embedded items. This is absolutely necessary for embedding the current input of a network, or of a navigation system, without modifying the embedding of the learning set (and hence the network itself), or of the landmarks. A straight fast procedure is required for on-line computation, while the exactitude of the embedding is not the most relevant characteristic in the case of noisy data, however, this is a way of obtaining fast computation. The present approach lies on mathematical fundations which extend the earlier mentioned classical results concerning the isometric embedding problem. Surprisingly, these classical results were much more widely used in the study of positive definite functions than in the field of multidimensional scaling, despite the fact that multidimensional scaling is basically an isometric (or monotonic) embedding problem.

2. Conventions and background

2.1. Notations

The identity matrix is denoted I. The transposed row vector of a collumn vector v is denoted v', while the transposed matrix of a matrix Q is denoted Q'. For specifying the content and size of a function vector or a function matrix, it will be convenient to use notations of the form:

$$v = [v_i]_{i=1,\ldots,n},$$

where v_i can be an expression,

$$D = [d_{ij}]_{i,\,j=0,\ldots,n},$$

where d_{ij} can be an expression.

For example, let A, B and C be three $(n+1)\times(n+1)$ matrices, then

$$A + B - C = [a_{ij} + b_{ij} - c_{ij}]_{i,\,j=0,\ldots,n}.$$

2.2. Usual definitions

2.2.1. Topological and metric spaces

A 'metric' or 'distance' associated to a set S, is a real valued function d on $S\times S$ such that, for any $a, b, c \in S$, one has the four following properties: (1) $d(a,b) \geq 0$ and $d(a,a) = 0$, (2) $d(a,b) = d(b,a)$, (3) $d(a,b) \leq d(a,c) + d(b,c)$, (4) if $a \neq b$ then $d(a,b) > 0$. The triangle inequality (3) can also be writen as $d(a,b) \geq |d(a,c) - d(b,c)|$.

A function which satisfies only requirements (1)–(3) is called a 'semi-metric'. A function d' which satisfies only requirements (1) and (2) is sufficient for inducing a topology on S. An open (resp. closed) 'ball' of center $x \in S$, and of radius $r \geq 0$, is the set of points $\{y \in S; d'(x,y) < r(\text{resp. } \leq r)\}$. The set of all balls is a neighbourhood system and then, (S, d') is a topological space. Now, if d is a true metric, then (S, d) is called a 'metric space', or 'metric set'. If S is a finite set of $n+1$ elements, then one can associate to $(S,d)a$ 'distance matrix' of the form $D = [d_{ij}]_{i,j=0,\ldots,n}$, and one can do the same even if d is not a true metric (while avoiding the term 'distance matrix' in this case).

2.2.2. Minkowskian metrics

A Minkowskian metric is a metric associated to R^n of the form

$$M_q(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^q\right)^{1/q}, \qquad q \geq 1.$$

The most usual Minkowskian metrics are the 'city-block' metric M_1, the Euclidean metric M_2, and the 'dominance' metric $M_\infty(X,Y) = \max_i |x_i - y_i|$. While all Minkowskian metrics are invariant by changing the origin or the sign of coordinates, the Euclidean metric is the only one which is also invariant by any orthogonal transform of the coordinates (e.g. rotation).

2.2.3. Embeddings

A metric space (S',d') is said to be 'isometrically embeddable' in a metric space (S,d) if there is a mapping f from S' to S such that, for any $a,b \in S'$, one has $d(f(a),f(b)) = d'(a,b)$.
A topological space (S',d') will be said to be 'monotonically embeddable' in a metric space (S,d) iff there is a continuous strictly increasing function g on $[0,\infty)$ such that $(S',g(d'))$ is a metric space isometrically embeddable in (S,d).

2.2.4. Properties of symmetric matrices

The 'Rayleigh's ratio' of a symmetric matrix B by a non-zero vector v is the ratio $R_B(v) = v'Bv/(v'v)$. The lowest eigenvalue of B is equal to $\inf_v R_B(v)$, while the greatest eigenvalue of B is equal to $\sup_v R_B(v)$.
A real symmetric matrix B of order $n \times n$ is said to be 'positive definite' iff for any vector $v \in R^n$, one has: $v'Bv \geq 0$. It is equivalent to say that none of the eigenvalues of B is negative.
A real symmetric matrix B of order $n \times n$ is said to be 'strictly positive definite' iff for any non-zero vector $v \in R^n$, one has: $v'Bv > 0$. It is equivalent to say that all eigenvalues of B are strictly positive.
A real symmetric matrix B of order $n \times n$ is said to be 'almost negative definite' (Donoghue, 1974; Micchelli, 1986) iff for any vector $v \in R^n$ such that $\sum_{i=1}^n v_i = 0$, one has: $v'Bv \leq 0$.
A real symmetric matrix B of order $n \times n$ will be said to be 'almost strictly negative definite' iff for any non-zero vector $v \in R^n$ such that $\sum_{i=1}^n v_i = 0$, one has: $v'Bv < 0$.

2.3. Useful theorems

Bessel–Parseval's inequality. Let H be a pre-Hilbert space, and $(e_i)_{i=1,\ldots,m}$ be an orthonormal family of H. Then, for any vector $x \in H$, the family $(|\langle x,e_i\rangle|^2)_{i=1,\ldots,m}$ is summable and $\sum_{i=1}^m |\langle x,e_i\rangle|^2 \leq \|x\|^2$, where $\langle\cdot,\cdot\rangle$ denotes the scalar product of H, and $\|\cdot\|$ is the corresponding norm. The orthonormal family $(e_i)_{i=1,\ldots,m}$ is complete for x iff $\sum_{i=1}^m |\langle x,e_i\rangle|^2 = \|x\|^2$ (Parseval's identity).

Theorem 1 (Gerschgorin). *Let B be a square $n \times n$ matrix. Then all eigenvalues of B belong to the union of the intervals defined by*

$$|x - b_{ii}| \leq \sum_{j \neq i} |b_{ij}|, \qquad 1 \leq i \leq n.$$

Lemma 1 (Fréchet, 1910; Schoenberg, 1938). Any finite metric set (S,d) of $n+1$ points may be embedded isometrically in (R^n, M_∞).Solution:

$$[x_{ij}]_{i=1,\ldots,n,j=0,\ldots,n} = [d_{ij}]_{i=1,\ldots,n,j=0,\ldots,n}.$$

Theorem 2 (Schoenberg, 1937, 1938; Micchelli, 1986). *A necessary and sufficient condition for the isometric embeddability of a finite metric set (S,d) of $n+1$ elements in an Euclidean space is that one of the two following statements be true*:

(i) *the matrix* $[d_{0i}^2 + d_{0j}^2 - d_{ij}^2]_{i,j=1,\ldots,n}$ *is positive definite*,
(ii) *the matrix* $[d_{ij}^2]_{i,j=0,\ldots,n}$ *is almost negative definite*.

Moreover one has (i) $\Leftrightarrow$ (ii).

Corollary 1. *The two following statements are equivalent*:

(iii) *the matrix* $[d_{0i}^2 + d_{0j}^2 - d_{ij}^2]_{i,j=1,\ldots,n}$ *is strictly positive definite*,
(iv) *the matrix* $[d_{ij}^2]_{i,j=0,\ldots,n}$ *is almost strictly negative definite*.

Proof. It suffices to replace ' $\geq$ ' and ' $\leq$ ' by ' $>$ ' and ' $<$ ', respectively, in the inequalities stated by Schoenberg (1938, pp. 525–526). □

Corollary 2. *If the finite metric set (S,d) is isometrically embeddable in an Euclidean space, then the required minimum dimension of this space is equal to the rank of the matrix* $[d_{0i}^2 + d_{0j}^2 - d_{ij}^2]_{i,j=1,\ldots,n}$.

Proof. Considering d as an Euclidean distance and taking the element of S with the index 0 as the origin (that is $X_0 = 0$), one has:

$$d_{0i}^2 + d_{0j}^2 - d_{ij}^2 = \|X_i\|^2 + \|X_j\|^2 - \|X_i - X_j\|^2 = 2X_i \cdot X_j,$$

that is

$$[d_{0i}^2 + d_{0j}^2 - d_{ij}^2]_{i,j=1,\ldots,n} = 2X'X.$$

Since $X'X$ is symmetric, one has $X'X = Q\Delta Q'$, where Q is the orthogonal matrix of eigenvectors, and Δ is the diagonal matrix of eigenvalues of $X'X$. Since these eigenvalues are not negative (after Theorem 2), on can take $X = \Delta^{1/2}Q'$ as

an embedding solution. Then the number of dimensions is equal to the number of non-zero eigenvalues, and this completes the proof. □

3. Monotonic embeddability in metric spaces

In this section, one examines general conditions of monotonic embeddability of data sets in a real metric space, whenever it is not required that this space be Euclidean.

Lemma 2. *Let S be any finite set comprised of $n+1$ distinct elements, and let μ be a real valued function on $S \times S$ such that, for any $a, b \in S$, one has $\mu(a,a) = 0$, and if $a \neq b$ then $\mu(a,b) = \mu(b,a) > 0$, $\mu(a,b) < \infty$. Then there is a real $\gamma(S) > 0$ such that for any stricly positive real $p \leq \gamma(S)$, the space (S, μ^P) is a metric set isometrically embeddable in (R^n, M_∞).*

Proof. Consider the matrix $[\mu_{ij}]_{i,j=0,\ldots,n}$ associated to the (non-metric) space (S, μ), let $\inf \mu = \min_{i \neq j} \mu_{ij} > 0$, and $\sup \mu = \max_{i \neq j} \mu_{ij}$. Then, for a real $p > 0$, the space (S, μ^p) is a metric set provided that for any i, j, k, one has the triangle inequality $\mu_{ij}^p \leq \mu_{ik}^p + \mu_{kj}^p$, which is guaranteed if p is such that $(\sup \mu)^p \leq 2(\inf \mu)^p$, that is $p \leq \ln(2)/\ln(\sup \mu/\inf \mu) \leq \gamma(S)$. Then the isometric embeddability of (S, μ^p) in (R^n, M_∞) results from Fréchet–Schoenberg's lemma (i.e. Lemma 1), which proves Lemma 2. □

Given Lemma 1, it is quite natural to ask whether any finite metric set is isometrically embeddable in an Euclidean space. The answer to this question is easy, since one can find many decisive examples.

Lemma 3. *There are finite metric sets which are not isometrically embeddable in an Euclidean space.*

Proof. Consider finite metric sets of $n+1$ elements, $n \geq 4$, whose distance matrices have the following structure. Chose a real vector $z \in R^n$, $n \geq 4$, such that

$$\|z\| = 1, \qquad |z_i| \leq 1/2, \qquad 1 \leq i \leq n, \qquad (z_i - z_j)^2 \leq 3/4,$$

$$1 \leq i \leq j \leq n.$$

Define the distances by

$$d_{ii} = 0, \qquad 0 \leq i \leq n, \qquad d_{0i} = d_{i0} = \sqrt{1/2 - z_i^2},$$

$$1 \leq i \leq n, \qquad d_{ij} = d_{ji} = \sqrt{1 - (z_i - z_j)^2},$$

$$1 \leq i < j \leq n.$$

Using this definition, one obtains

$$1/2 \leq d_{ij} = d_{ji} \leq 1, \qquad 0 \leq i < j \leq n,$$

and then, for any i, j, k, one has

$$d_{ij} \leq d_{ik} + d_{kj},$$

which is triangle inequality. Since d obviously has the other properties of a distance, it is clear that this is actually a metric for the considered set.

Now, one has also

$$2d_{0i}^2 = 1 - 2z_i^2,$$

$$d_{0i}^2 + d_{0j}^2 - d_{ij}^2 = -z_i^2 - z_j^2 + (z_i - z_j)^2 = -2z_i z_j,$$

that is

$$[d_{0i}^2 + d_{0j}^2 - d_{ij}^2]_{i,j=1,\ldots,n} = I - 2zz',$$

where the right member is, by definition, a Householder's matrix. Since one has

$$z'(I - 2zz')z = -1,$$

it is clear that the matrix $[d_{0i}^2 + d_{0j}^2 - d_{ij}^2]_{i,j=1,\ldots,n}$ is not positive definite, and then, following Theorem 2, the finite metric set whose distance matrix is $[d_{ij}]_{i,j=0,\ldots,n}$ is not isometrically embeddable in an Euclidean space, which proves Lemma 3. □

The above proof concerns metric sets of at least five points. Metric sets of one ore two points are trivially isometrically embeddable in an Euclidean space. Metric sets of three points are also embeddable since the triangle inequality always allows for building a triangle in R^2 with appropriate side lengths. The case of metric sets of four points has been studied by Blumenthal (1936), who found that raising the metric to a positive power, lower or equal to 1/2, compels the embeddability.

4. Monotonic embedding in Euclidean spaces

We have seen in Section 3 that, under quite general conditions, data sets can be monotonically embedded in at least one real metric space in a simple way (Lemma 2), while this does not guarantee the embeddability in an Euclidean space (Lemma 3). In this section, one states general conditions of monotonic embeddability of data sets in Euclidean spaces.

Definition 1. Let S be any set. A 'dissimilarity' function associated to S is a real valued function μ on $S \times S$ such that, for any $a, b \in S$, one has $\mu(a,a) = 0$, and if $a \neq b$ then $\mu(a,b) = \mu(b,a) > 0$. The set of dissimilarity functions associated to S is denoted $\delta(S)$.

Note that metrics are dissimilarities (which, in addition,

satisfy triangle inequality), while semi-metrics are not dissimilarities since they allow distinct elements to have a zero distance.

Definition 2. One denotes G the set of functions $g(\mu;p)$ of one real variable $\mu \geq 0$, and one real parameter $p > 0$ such that:

$g(\mu;p)$ is a continuous strictly increasing function of μ,

$g(0;p) = 0$,

for any $\mu > 0$, for any real $\epsilon > 0$, there is a real $a > 0$ such that

$p \leq a \Rightarrow |g^2(\mu;p) - 1| \leq \epsilon$.

Example 1. Power function $g(\mu;p) = \mu^p \in G$, with the restriction that $\sup \mu < \infty$:Without loss of generality, one can assume that $\epsilon < 1$, then

$$a = \begin{cases} \ln(\epsilon + 1)/(2 \ln \mu), & \text{if } \mu > 1, \\ \ln(1 - \epsilon)/(2 \ln \mu), & \text{if } \mu < 1, \\ > 0, & \text{if } \mu = 1. \end{cases}$$

Example 2. Weibull's function

$g(\mu;p) = 1 - \exp(-\mu^r/p) \in G$, with $r > 0$:

with $\epsilon < 1$ one has

$$a = \frac{-\mu^r}{\ln(1 - \sqrt{1 - \epsilon})}.$$

Lemma 4. *Let $T = [t_{ij}]_{i,j=0,\ldots,n}$ be a trivial distance matrix associated to $n + 1$ distinct elements, that is $t_{ii} = 0$, and if $i \neq j$ then $t_{ij} = 1$. Consider a symmetric matrix of the form $[g^2(\mu_{ij};p)]_{i,j=0,\ldots,n}$, where $\mu \in \delta(S)$, for a given set S of $n + 1$ distinct elements* (Definition 1), *and $g \in G$* (Definition 2). *Then for any real $\epsilon > 0$, there is a real $a > 0$ such that*

$0 < p \leq a \Rightarrow \|[g^2(\mu_{ij};p)]_{i,j=0,\ldots,n} - T\|_\infty \leq \epsilon$,

where the matricial norm $\|B\|_\infty = \max_i \Sigma_j |b_{ij}|$.

Proof. The diagonals of the two symmetric matrices $[g^2(\mu_{ij};p)]_{i,j=0,\ldots,n}$ and T are zero, which implies that there are at most n non-zero differences per row of the difference matrix. For each non-diagonal cell of the first matrix, there is a real $a_{ij} > 0$ such that $p \leq a_{ij} \Rightarrow |g^2(\mu_{ij};p) - 1| \leq \epsilon/n$, by definition of the function g. This implies that there is an appropriate real number a such that $a \geq \min_{0 \leq i < j \leq n} a_{ij} > 0$, which completes the proof. □

Theorem 3. *Let S be any finite set comprised of $n + 1$ distinct elements, consider a dissimilarity $\mu \in \delta(S)$, and a function $g \in G$. Then the following two statements are true*:

(i) *There is a real $\alpha(S) > 0$ such that for any stricly positive real $p \leq \alpha(S)$, the space $(S, g(\mu;p))$ is a metric set isometrically embeddable in an Euclidean space whose dimension is at most n.*
(ii) *There is a real $\beta(S) > 0$, $\beta(S) \leq \alpha(S)$, such that for any stricly positive real $p < \beta(S)$, the space $(S, g(\mu;p))$ is a metric set isometrically embeddable in an Euclidean space whose dimension is exactly n.*

Proof. Step 1. Let $T = [t_{ij}]_{i,j=0,\ldots,n}$ be a trivial distance matrix, whose diagonal coefficients equal 0, and other coefficients equal 1. Then for any vector $v \in R^{n+1}$ such that $\sum_{i=0}^n v_i = 0$, one obtains

$$v'Tv = v'\left[\left(\sum_{j=0}^n v_j\right) - v_i\right]_{i=0,\ldots,n} = v'(-v) = -\|v\|^2 \leq 0,$$

which proves that T is almost negative definite.
Step 2. Consider the symmetric matrix $D_p = [g^2(\mu_{ij};p)]_{i,j=0,\ldots,n}$ associated to the space $(S, g(\mu;p))$, for a given $p > 0$. One can write

$D_p = T + (D_p - T)$,

and for any vector $v \in R^{n+1}$ such that $\sum_{i=0}^n v_i = 0$, one has

$v'D_pv = v'Tv + v'(D_p - T)v = -\|v\|^2 + v'(D_p - T)v$,

after Step 1. Then one obtains the equivalence

$v'D_pv \leq 0 \Leftrightarrow v'(D_p - T)v \leq \|v\|^2$,

while a sufficient condition for obtaining the last inequality is that the greatest eigenvalue of the symmetric matrix $(D_p - T)$ not be greater than 1 (after usual properties of Rayleigh's ratio for symmetric matrices). Now, after the well-known theorem of Gerschgorin, one knows that the greatest eigenvalue of any square matrix B cannot exceed $\|B\|_\infty$, while after Lemma 4, there is a real $a > 0$ such that

$0 < p \leq a \Rightarrow \|D_p - T\|_\infty \leq 1$.

Then, clearly, for any p in the above interval, the matrix D_p is almost negative definite, and (i) of Theorem 3 is proved (in account of Theorem 2 and Corollary 2). Note, however, that the condition used for defining the upper bound (a) of the critical interval of p is sufficient but not necessary, and one has in fact $\alpha(S) \geqq a$.
Step 3. After Lemma 4, for any ϵ such that $0 < \epsilon < 1$, there is a real $b > 0$ such that $0 < p \leq b \Rightarrow \|D_p - T\|_\infty \leq \epsilon$. Then, for any p in this interval, for any non-zero vector $v \in R^{n+1}$ such that $\sum_{i=0}^n v_i = 0$, one has $v'D_pv < 0$, which

implies that the matrix $[g^2(\mu_{0j};p)+g^2(\mu_{0j};p)-g^2(\mu_{ij};p)]_{i,j=1,\ldots,n}$ is strictly positive definite (after Corollary 1), and then the dimension of the embedding space is exactly n (after Corollary 2). Note that one has in fact $\beta(S)>b$. Moreover, comparing (i) with (ii), it is clear that $\beta(S)\leq\alpha(S)$, and Theorem 3 is proved. □

Corollary 3. *Let S be a given set of n distinct elements, let Ω be a generalization set including S, while $\mu\in\delta(\Omega)$, and $g\in G$. Then there is a real $\alpha(\Omega|S)>0$ such that for any stricly positive real $p\leq\alpha(\Omega|S)$, and for any $y\in\Omega$, the space $(S\cup\{y\},g(\mu;p))$ is a metric set isometrically embeddable in a n-dimensional Euclidean space.*

Proof. One obviously has $\alpha(\Omega|S)=\inf_{y\in\Omega}\alpha(S\cup\{y\})$, while after Theorem 3:
if $y\in S$ then $\alpha(S\cup\{y\})=\alpha(S)>0$, if $y\notin S$ then $\alpha(S\cup\{y\})>0$,
which completes the proof. □

When the embedding parameter p tends to 0, the space $(S,g(\mu;p))$ tends to (S,t), where t is the trivial distance. Then the points of the mapping in an Euclidean space tend to the vertices of an equilateral polytope (e.g. an equilateral triangle for $n=2$, an equilateral tetraedron for $n=3$, and so on), which account for the equidistance of these points. Then the particular structure of the data set is represented more and more weakly in the embedding space as p tends to 0. However, as long as $p>0$, the structural information remains available, since the g transform remains inversible, by virtue of its strict monotonicity with respect to μ. Corollary 3 is particularly important for neural network applications since it states that one can embed any generalization item (current input), belonging to a given generalization set Ω, together with a finite learning set (or a set of landmarks) S. Note, however, that this does not mean that one can globally embed an infinite set Ω.

5. Relation with the city-block metric

There is a particular relation between city-block metric sets and Euclidean metric sets. This relation allows for a very simple determination of an appropriate g function with an appropriate embedding parameter p, whenever one knows that a data set is a city-block metric set. Moreover, it can be important to know such a relation for the study of psychological spaces, since various observations suggested that objects described on dimensions of the same nature (e.g. width and height) are encoded by humans in an Euclidean space, while objects described on heterogeneous dimensions (e.g. size and colour) are encoded in a city-block space. However, there has been some debate on this question (Ennis, 1988; Nosofsky, 1986; Shepard, 1986).

Lemma 5. *Let S be a finite set of $n+1$ distinct points of R^m, and d be a city-block metric (M_1) on $S\times S$, where d is finite on this set. Then the set $(S,d^{1/2})$ is isometrically embeddable in the Euclidean space (R^n,M_2).*

Proof. After the definition of a city-block metric, one has

$$[d_{ij}]_{i,j=0,\ldots,n}=\sum_{k=1}^{m}[d_{ij}^{(k)}]_{i,j=0,\ldots,n},$$

where the upper index (k) indexes the dimensions, and each matrix $[d_{ij}^{(k)}]_{i,j=0,\ldots,n}$ is the matrix of a Minkowskian metric on a real space of dimension 1. In dimension 1, all Minkowskian metrics are equal, and in particular they are equal to the Euclidean one. Since the power function is in G for finite measures, Theorem 3 guarantees that if one raises an Euclidean metric to the power 1/2, it remains an Euclidean metric (for a different set of points), and after Theorem 2, if one squares this new Euclidean metric one obtains that all matrices $[d_{ij}^{(k)}]_{i,j=0,\ldots,n}$, $k=1,\ldots,m$, are almost negative definite. Since a sum of almost negative definite matrices is an almost negative definite matrix, one concludes that $[d_{ij}]_{i,j=0,\ldots,n}$ is almost negative definite, which implies that $(S,d^{1/2})$ is isometrically embeddable in the Euclidean space (R^n,M_2), and Lemma 5 is proved. □

6. Cholesky factorization of a singular matrix

In this section, one states a result which will be necessary for defining an embedding algorithm in Section 7. One knows that a symmetric strictly positive definite matrix B can be factorized as a product of the form $B=X'X$, where X is a non-singular upper triangular matrix. This is the well-known Cholesky factorization of B. Now, a difficulty arises whenever B is singular, since in this case the Cholesky factorization leads to a division by zero resulting from an indetermination of the form $0\cdot x=0$ in Cholesky's equations. The following result allows for removing this indetermination, and then to define a simple variant of the Cholesky factorization for possibly singular matrices.

Theorem 4. *Let B be a symmetric, possibly singular, positive definite matrix of order $n\times n$. Then there is an upper triangular matrix X such that $X'X=B$, $x_{ii}\geq 0$, $1\leq i\leq n$, and if for an index i one has $x_{ii}=0$, then $x_{ij}=0$, $1\leq j\leq n$. Moreover, the matrix X with these properties is unique.*

Proof. *Existence proof.* Since B is symmetric positive definite, there is a square matrix Y such that $Y'Y=B$ (for example $Y=\Delta^{1/2}Q'$, as in the proof of Corollary 2). One can define a special variant of the usual QR factorization, this variant being of the form $Y=HX$, where H is an

orthogonal matrix and X is upper triangular:

1. $H_0 = 0$ (auxiliary vector),
2. for $j = 1$ to n do (3)–(5)
3. $Z_j = Y_j - \sum_{k=0}^{j-1} (Y_j \cdot H_k) H_k$,
4. if $Z_j = 0$ then replace it by any non-zero vector Z_j such that $Z_j \perp \{H_1, ..., H_{j-1}, Y_j, ..., Y_n\}$,
5. $H_j = Z_j / \|Z_j\|$,
6. $X = H' Y$.

In the above procedure, H_j or Y_j, with $j > 0$, stands for the jth column vector of the corresponding matrix. Steps (1), (2), (3) and (5) correspond to the well-known Gram–Schmidt orthonormalization process, while a variant is introduced at step (4). Then a crucial point is the existence of an appropriate non-zero vector Z_j when step (3) provides a zero vector. All vectors have the dimension n, while if step (3) provides a zero vector, this means that Y_j is a linear combination of the vectors H_k, $k = 1, ..., j-1$. Then the rank of the matrix whose collumn vectors are $\{H_1, ..., H_{j-1}, Y_j, ..., Y_n\}$ is at most $n-1$, which implies that there is a non-zero vector of dimension n which is orthogonal to all these vectors, and step (4) is valid. Now, one verifies that step (6) provides a matrix X which has all the desired properties.

X is upper triangular since $H_j \perp Y_k$, $k = 1, ..., j-1$.

Using the equation of step (3), one obtains that

$$Y_j \cdot Z_j = \|Y_j\|^2 - \sum_{k=1}^{j-1} (Y_j \cdot H_k)^2 \geq 0,$$

since the vectors H_k, $k = 1, ..., j-1$, form an orthonormalized basis (complete or not for Y_j), and $\sum_{k=1}^{j-1} (Y_j \cdot H_k)^2 \leq \|Y_j\|^2$, by virtue of Bessel–Parseval's inequality. If the basis is not complete for Y_j, then the above inequality is strict, which implies that $x_{jj} = H_j \cdot Y_j > 0$, while if the basis is complete for Y_j, then step (3) provides a zero vector, and step (4) guarantees that H_j is orthogonal to all columns of Y, which implies that the whole jth row of X is zero.

Finally, one has $X'X = Y'HH'Y = Y'Y = B$, since H is orthogonal (that is $HH' = I$), which completes the existence proof.

Comment. Assume that a vector H_k has been chosen using step (4). Then this choice does not affect the computation of the next vectors by step (3) since $(Y_j \cdot H_k) = 0$, $j = k+1, ..., n$. In particular, the set of vectors for which step (3) generates a zero result remains the same, whatever be the particular choice of H_k. On the other hand, all vectors generated using step (4) provide an identical effect on the matrix X, that is a zero row. This implies that the particular choice of certain vectors by step (4) does not affect the resulting matrix X, and then this matrix is unique for a given matrix Y, while the orthogonal matrix H is not unique if Y (and hence B) is singular.

Unicity proof. It remains to prove that X does not depend on a particular choice of the matrix Y. Since an appropriate matrix X exists (see the existence proof earlier), one can write the Cholesky's equations of the system $B = X'X$. Using an auxiliary row vector $[x_{0j}]_{j=1,...,n} = 0$, for writing convenience, one obtains:

$j = 1, ..., n, \qquad i = 1, ..., j:$

if $i = j$ then

$$b_{ii} = \sum_{k=0}^{i} x_{ki}^2 \Rightarrow x_{ii} = \sqrt{b_{ii} - \sum_{k=0}^{i-1} x_{ki}^2},$$

else

$$b_{ij} = x_{ii} x_{ij} + \sum_{k=0}^{i-1} x_{ki} x_{kj} \Rightarrow x_{ij}$$

$$= \begin{cases} \left(b_{ij} - \sum_{k=0}^{i-1} x_{ki} x_{kj} \right) / x_{ii}, & \text{if } x_{ii} > 0, \\ 0, & \text{if } x_{ii} = 0 \end{cases}$$

by hypothesis on X properties.

One can note that the solution X is unequivocally determined by the constraints without any reference to a particular matrix Y, which completes the proof of Theorem 4. □

Note that we have just defined a Cholesky's factorization of a possibly singular symmetric positive definite matrix B, which was in fact the main goal of Theorem 4. This factorization is very similar to usual Cholesky's factorizations of strictly positive definite symmetric matrices, except that the singular case ($x_{ii} = 0$) is allowed, with a unique guaranteed solution.

7. Monotonic embedding algorithm

We are now ready to define a monotonic embedding algorithm of data sets in Euclidean spaces with the desired properties for neural network applications. First, one can note that if Y is an isometric embedding mapping of a set S in an Euclidean space, then so is QY, for any orthogonal matrix Q, since the Euclidean distance is invariant by any orthogonal transform of coordinates. In particular $X = H'Y$, where X is upper triangular as in Section 6, is a solution. After Theorem 3 (and Corollary 3), for a given data set S, there is $\alpha(S) > 0$ such that $0 < p \leq \alpha(S)$ implies that

$$\tfrac{1}{2} [g^2(\mu_{0i}; p) + g^2(\mu_{0j}; p) - g^2(\mu_{ij}; p)]_{i,j=1,...,n} = X'X = B,$$

where $\mu \in \delta(S)$, $g \in G$, and the solution X of course depends on p.

After Theorem 4, this system can be solved using an appropriate Cholesky factorization, even if B is singular, which can happen if $p \geq \beta(S)$. It remains to define a way of finding an appropriate p, given that in general, one does not

know $\alpha(S)$ a priori. For doing this, one can exploite the fact that if $p \leq \alpha(S)$, then B is positive definite and the Cholesky factorization has a real solution, while if $p > \alpha(S)$, then B has negative eigenvalues and there are negative arguments for a square root function in the computation of the diagonal coefficients of X. Hence, each time that one detects such an imaginary value case, this means that $p > \alpha(S)$, and that one must lower p. Then a simple bounding procedure allows for approximating $\alpha(S)$ (resp. $\beta(S)$) as closely as one wants.

7.1. Procedure for embedding one element

The following procedure (EMBED) allows for embedding the element number j of a set, while the elements of number 0 to $j-1$ have been previously embedded. This is the fundamental procedure which is called by all particular application algorithms. One can eventually limit the dimension m of the embedding space, while if no limit of dimension is fixed, it suffices to call EMBED with an arbitrary strictly positive parameter $m \geq j$, remembering that $x_{ij} = 0$ if $i > j$. One can arbitrarily limit the dimension m since the Cholesky factorization of the matrix B is not only incremental with respect to the collumns, but also with respect to the rows. One assumes that a dissimilarity function μ is associated to the data set, that a function $g \in G$ has been fixed, and that the embedding parameter p currently has a defined value. The procedure returns an arbitrary negative diagonal value ($x_{jj} = -1$) when the jth element is not embeddable using the current value of p.

```
EMBED (j,m,p)
  if j < m then for i := j + 1 to m do x_ij := 0
  if j > 0 then
    for i := 1 to min(j,m) do
      b_ij := (1/2)(g^2(μ_0i; p) + g^2(μ_0j; p) − g^2(μ_ij; p))
      if i > 1 then s := Σ_{k=1}^{i−1} x_ki x_kj else s := 0
      b_ij := b_ij − s
      if i = j then
        if b_ij is very close to 0 then b_ij := 0
        if b_ij < 0 then x_ij := −1 else x_ij := √b_ij
      if i ≠ j then
        if x_ii = 0 then x_ij := 0 else x_ij := b_ij/x_ii.
```

Note: in the case ($i = j$), if b_{ij} is very close to 0 then it is set to 0 before the test ($b_{ij} < 0$) in order to take into account the rounding errors which occur in computer's floating point arithmetic.

7.2. Multidimensional scaling procedure

The procedure named MDS allows for monotonically embedding a set S of $n+1$ elements, for given μ and g functions, while one requires that any diagonal value of the matrix X is at least equal to a positive value e. If one calls MDS with $e = 0$ then the final value of p is an approximation of $\alpha(S)$ with the specified 'precision'. The parameter pSup (strictly positive) is choosen as the upper bound of the search interval of p. If the final value of p is equal to pSup, this means that pSup $\leq \alpha(S)$. Similarly, one can obtain an approximation of $\beta(S)$ by choosing e just greater than zero. Take care that too large a value of e can be incompatible with the data, which leads p to tend to zero.

```
MDS(e, n, pSup)
  EMBED(0, n, 1) {the first element is the origin}
  pInf := 0
  p := pSup
  repeat
  j := 0
    repeat
    j := j + 1
    EMBED(j, n, p)
    until (j = n) or (x_jj < e)
  if (x_jj < e) then pSup := p else pInf := p
  p := (pInf + pSup)/2
  until (pSup − pInf) < precision.
```

The procedure MDS1 allows for embedding a new element together with $n+1$ previously embedded elements. MDS1 must be called with the current value of the embedding parameter p. This value will be lowered in a call to MDS if necessary. For avoiding this modification, replace the call to MDS by any appropriate instruction (for example, display the message 'the new item is not embeddable').

```
MDS1(e, n, p)
  m := n + 1
  EMBED(m, m, p)
  if x_mm < e then MDS(e, m, p)
```

The procedure MDS1 is useful for approximating $\alpha(\Omega|S)$ of Corollary 3, while S is a learning set of $n+1$ elements, and Ω is a generalization set. Each generalization element must be embeddable together with S, independently of other generalization elements. Then repeatedly sampling Ω and embedding each generalization item together with S by MDS1(0, n, p), one can hope that p converges to $\alpha(\Omega|S)$.

7.3. Fixed dimension spatial maps

The procedure named MAP allows for building a spatial map of fixed dimension m from a set of approximated distances between $n+1$ objects ($n \geq m$). The first $m+1$ objects are taken as landmarks, the first one being the origin of coordinates ($X_0 = 0$). The embedding space is completely defined by the landmarks, while the remaining objects are embedded in this space only, and their mutual distances are not taken into account. In this type of application, μ is in general an approximation of an Euclidean distance (however, this is not necessary), and one chooses for g the power function, that is $g(\mu; p) = \mu^p$.

Table 1
Averaged values of $\alpha(S)$ on four runs of MDS(0, n, 10), and dimension (m') of the embedding space as functions of the size of the set ($n+1$), and the type of data space, with $g = \mu^p$, and $m = n$

Data space:	Non-metric		Metric		Euclidean	
n	$\alpha(S)$	m'	$\alpha(S)$	m'	α(S)	m'
4	0.202	3	0.894	3	1.102	3
8	0.159	7	0.781	7	1.003	7
16	0.084	15	0.596	15	1.003	15
32	0.054	31	0.540	31	>1	31
64	0.035	63	0.528	63	>1	63
128	0.023	127	0.499	127	>1	127

Hence, if μ is actually an Euclidean distance between the landmarks, then one obtains $p = 1$. The bounding method used in the procedure MDS for the search for p is replaced here by a simple decay method, starting from $p = 1$, and applying, if necessary, a reduction coefficient ($0 < \text{reduc} < 1$) close to 1. This strategy is fast whenever μ is close to an Euclidean distance between the landmarks. If this is not the case, then the decay method can of course be replaced by another one, for axample the one used in the MDS procedure. Taking a small $e > 0$ ensures that the embedding space is exactly of dimension m.

```
MAP(e, m, n, reduc)
   EMBED(0, m, 1) {the first element is the origin}
   p := 1
   OK := false
   repeat
   j := 0
      repeat
      j := j + 1
      EMBED(j, m, p)
      until (j = n) or ((j ≤ m) and (x_jj < e))
   if (j ≤ m) and (x_jj < e) then p := reduc × p else
                        OK := true
   until OK
```

Note that in real life navigational applications, the landmarks are frequently fixed objects. In this case, the embedding space must be computed only once, possibly in quite good conditions for the approximation of distances (e.g. prior exploration of the environment). Then the on-line computation reduces to embedding the remaining (possibly moving) objects, that is:

```
MAP1(m, n, p)
   for j := m + 1 to n do EMBED(j, m, p).
```

This is very simple, while the main problem is of course the on-line approximation of the distances.

Table 2
Averaged values of $\alpha(S)$ on four runs of MDS(0, 128, 10), and dimension (m') of the embedding space as functions of the type of metric data space, and of its dimension (m), with $g = \mu^p$, and $n = 128$

Data space:	Euclidean		Metric		City-block	
m	$\alpha(S)$	m'	$\alpha(S)$	m'	$\alpha(S)$	m'
4	1	4	0.146	127	0.504	127
8	1	8	0.114	127	0.511	127
16	1	16	0.160	127	0.524	127
32	1	32	0.219	127	0.547	127
64	1	64	0.330	127	0.592	127

8. Some numerical results

This section summarises numerical results obtained by the application of the procedure MDS (Section 7.2) to data sets of various types and sizes. For 'non-metric' data sets, dissimilarity symmetric matrices with zero diagonal coefficients were generated by a uniform random sampling of non-diagonal coefficients in the interval [0.01,100]. The distance matrices of metric data sets were generated by computing Minkowskian M_∞ distances between random vectors (uniform random coordinates in $[-1,1]$). After Lemma 1 (Fréchet–Schoenberg), this is representative of the whole set of finite metric sets. The distance matrices of 'Euclidean' data sets were generated in the same way, using the Euclidean metric M_2. while distance matrices of 'City-Block' metric sets were obtained using the metric M_1. The size of the sets ($n+1$ elements) was varied from $n = 4$ to 128. The actual dimension (m) of the metric spaces was also varied. Tables present averaged values of the observed embedding parameter bounds ($\alpha(S)$) on four independent problems solved by MDS(0, n, 10), and the obtained dimension of the embedding space (m' = number of non-zero diagonal coefficients of X). Tables 1 and 2 present results obtained with the power function as g function, while Table 3 presents results obtained with two distinct Weibull functions ($r = 1$ and 2, respectively) as g functions. The inspection of Table 1 shows that $\alpha(S)$ is lower for non-metric sets than for metric sets, and that it is greater than one for Euclidean sets only. Moreover, $\alpha(S)$ decreases as the

Table 3
Averaged values of $\alpha(S)$ on four runs of MDS(0, n, 10), and dimension (m') of the embedding space as functions of the size of the set ($n+1$), and the type of data space, with $g = 1 - \exp(-\mu^r/p)$, $m = n$

Data space	Non-metric		Metric		Euclidean	
	$\alpha(S)$	m'	$\alpha(S)$	m'	$\alpha(S)$	m'
$r = 1$, $n = 16$	1.419	15	1.594	15	>10	(16)
$r = 1$, $n = 128$	0.092	127	1.273	127	>10	(128)
$r = 2$, $n = 16$	6.022	15	0.802	15	5.648	15
$r = 2$, $n = 128$	0.009	127	0.831	127	>10	(128)

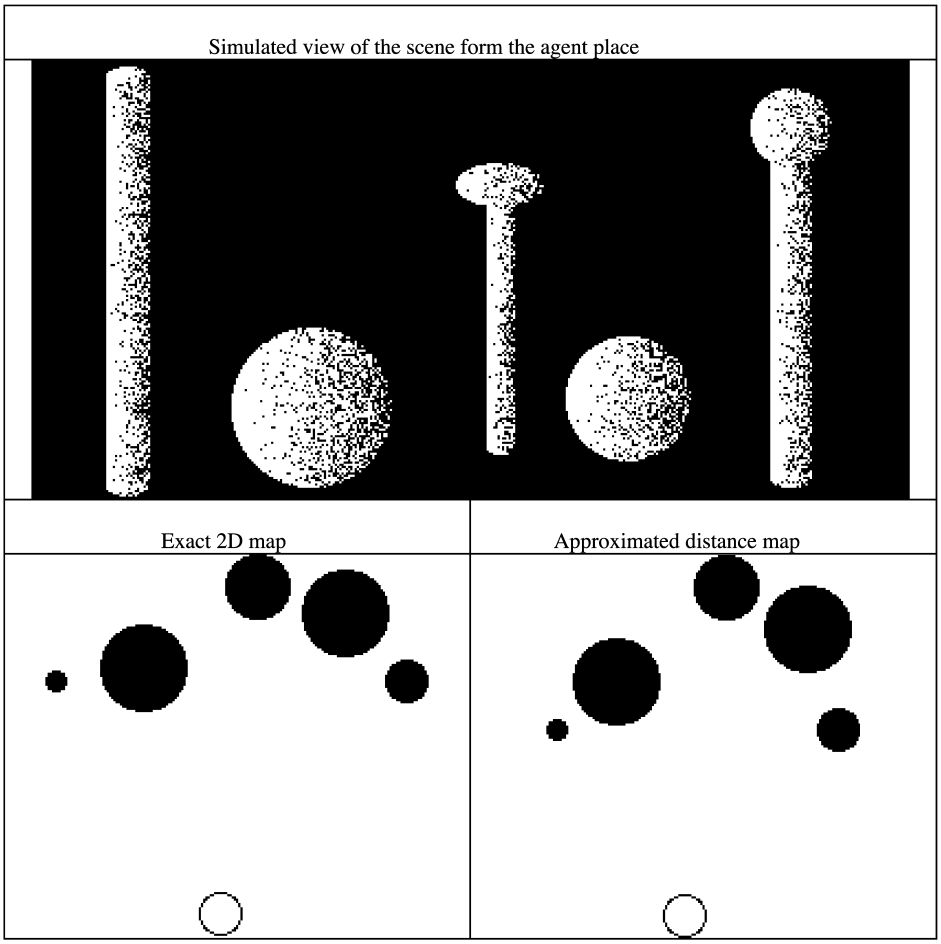


Fig. 1. Simulated view of a scene with three landmarks (pegs) and two possibly moving objects (balls), followed by two maps computed by the embedding algorithm from exact and approximated distance data. In these maps, the white disk stands for the autonomous agent, while the other objects are distinguished by the diameter of their head. The origin of the coordinates is always the thin peg.

size of the set (n) increases, always remaining greater than one for Euclidean sets. The obtained dimension of the embedding space for $p = \alpha(S)$ was always $m' = n - 1$, while the actual dimension of the used metric sets was $m = n$. Additional results were computed for City-Block metric sets. As expected from Lemma 5, $\alpha(S)$ was always greater than 0.5 for these sets and one obtained an averaged $\alpha(S) = 0.673$ with $n = m = 128$.

Table 2 shows the effect of the actual dimension (m) of three types of metric sets (with $n = 128$). The results are clear for Euclidean sets, where the actual dimension was always detected by the algorithm ($m' = m$). For the two other types of metric sets, the actual dimension was not detected ($m' = n - 1$), however, one can observe that $\alpha(S)$ systematically varied as a function of m. Further theoretical investigations are required for understanding this relation.

Table 3 rapidly provides some elements concerning the behavior of the embedding algorithm with Weibull functions. The algorithm works well with these functions which are more appropriate to function approximation contexts than to distance geometry.

Additional results showed, in all cases, that $\beta(S) = \alpha(S)$, that is, taking an embedding parameter p just lower than $\alpha(S)$ always provides an embedding space of dimension n exactly, whatever be m' for $p = \alpha(S)$. Finally, a general observation which can be outlined from the above results is that dissimilarity functions (which were completely random in this study) must be preferably built in a way which provides them with properties close to those of a metric, in order to avoid very low values of $\alpha(S)$ and large transformations of the data space.

9. Example of application to a robot navigation problem

As suggested in Section 1, there are various application fields and various ways of exploiting an embedding algorithm. The example presented in this section has the advantage of providing suggestive visual illustrations. Robotic and compuer vision are very active fields, while robot vision systems can vary considerably in their sophistication. For this illustration, minimal hypothesis concerning the robot technology were retained. We assume

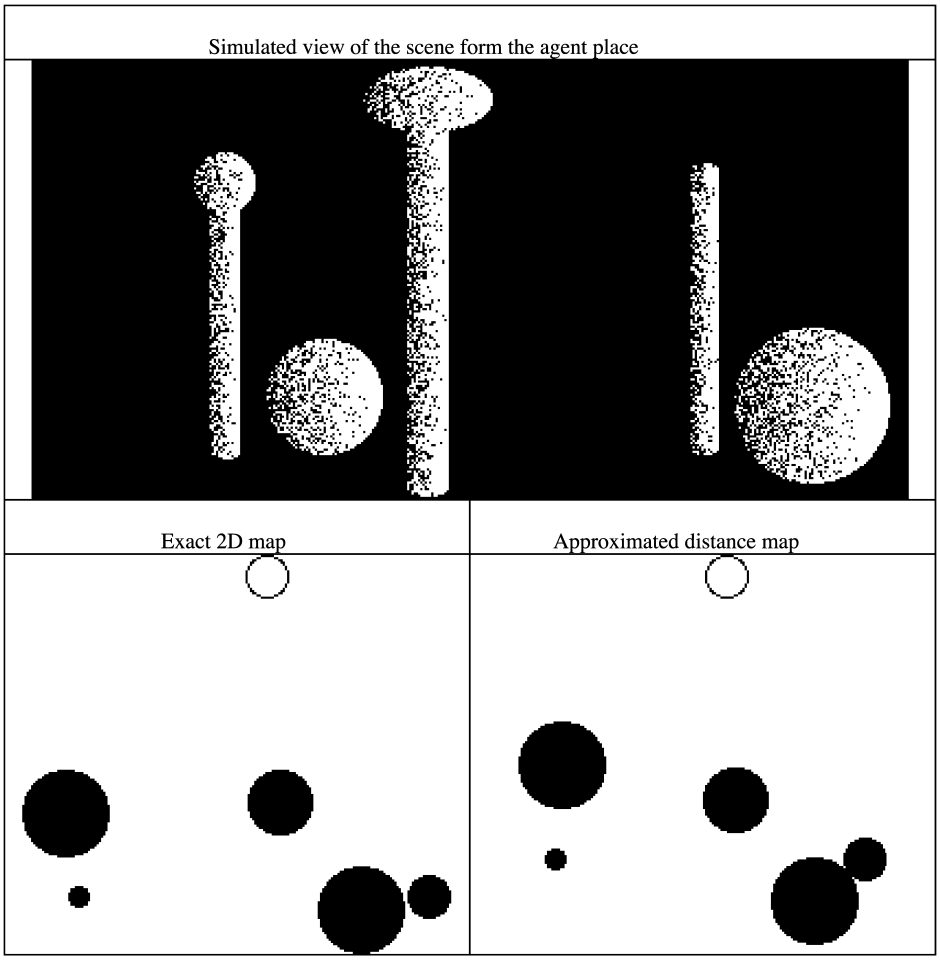


Fig. 2. Similar to Fig. 1, with the same landmarks and a different configuration of the three moving objects.

a robot (autonomous agent) equiped with a simple video camera whose images are numerised in a 200×400 matrix of binary pixels (black/white). We also assume that the robot's computational power is limited, and that the computer vision program is not very sophisticated. However, this program must be able to approximately segment an image into regions corresponding to individual objects, discriminate objects of various shapes, and measure the angular distance between any two points. If the two considered points are not simultaneously present in the visual field (e.g. if the robot lands between two objects), then the angular distance is the absolute rotation angle of the camera required for successively centering the two points. In addition, we assume that the program can use a data base providing the actual size of objects (at least approximately). Such a data base may have been obtained from a prior exploration of the environment. With all this information available, one can approximate the distances between objects (including the agent) by applying usual projection and triangulation formulas (such as those used in astronomy). Then the approximated distance matrix can be used as the input of the monotonic embedding algorithm in order to compute a 2D map of the spatial configuration of the objects (including the agent). Certain fixed objects can be used as landmarks, while at least three objects (fixed or not) are required for computing a 2D map. The main difficulties result from the presence of image numerization noise, shadows, noisy background, partially hidden objects, and poorly discriminable landmarks. This can result in a certain amount of error in the approximation of distances, which leads to distortions in the spatial map. Figs. 1 and 2 show two views of a simulated environment made of three landmarks (pegs) and two moving objects (balls). The choosen order of these objects is: thin peg, round head peg, ellipsoid head peg, and the balls. The two balls have undiscernable shapes, hence their set is a cluster which may eventually be encoded in a special way after a spatial map has been computed (Courrieu, 2001). Below each view, on can see two 2D maps computed by the procedure MAP(e, 2, 5, 0.999), with a small positive e appropriate to the data scale (which is arbitrary). The first map was computed by the embedding algorithm from exact distances (known a priori), while the other map was computed from distances appproximated by a rudimentary (poorly performing) computer vision program. Despite the distance approximation errors generated by this program (up to 20%), one

can see in Figs. 1 and 2 that the obtained maps are quite close to the exact ones.

10. Conclusion

Embedding algorithms and multidimensional scaling potentially have a wide set of applications in various fields (data analysis, function approximation on non-Euclidean topological spaces, autonomous agent navigation problems, psychological science). Mathematical fundations of a fast monotonic embedding algorithm of data sets in Euclidean spaces were presented, and then the algorithm was defined, with variants for various types of applications. The particularity of the algorithm, with respect to usual multidimensional scaling methods, is that it is straight, fast, and incremental, which makes it particularly appropriate to Neural Network applications and on-line computation. Some general numerical results were provided, and an illustrative example of application in robot navigation was presented. Further investigations are needed for solving remaining problems such as the optimal reduction of the embedding space dimension, however, this mainly concerns data analysis applications.

Acknowledgement

This work was partially supported by a grant from Fond National pour la Science – ACI “Cognitique” (2000, #90).

References

Blumenthal, L. M. (1936). New theorems and methods in determinant theory. *Duke Mathematical Journal*, *2*, 396–404.

Courrieu, P. (2001). Two methods for encoding clusters. *Neural Networks*, *14*, 175–183.

Cox, T. F., & Cox, M. A. A. (1994). *Multidimensional scaling*. London: Chapman & Hall.

Cressant, A., Muller, R. U., & Poucet, B. (1997). Failure of centrally placed objects to control the firing fields of hippocampal place cells. *The Journal of Neuroscience*, *17*(7), 2531–2542.

Cressant, A., Muller, R. U., & Poucet, B. (1999). Further study of control of place cell firing by intra-apparatus objects. *Hippocampus*, *9*, 423–431.

Demartines, P., & Hérault, J. (1997). Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. *IEEE Transaction on Neural Networks*, *8*, 148–154.

Donoghue, W. F. (1974). *Monotone matrix functions and analytic continuation*. Berlin: Springer.

Ennis, D. M. (1988). Confusable and discriminable stimuli: comment on Nosofsky (1986) and Shepard (1986). *Journal of Experimental Psychology: General*, *117*(4), 408–411.

Fréchet, M. (1910). Les dimensions d’un ensemble abstrait. *Mathematische Annalen*, *68*, 145–168.

Girosi, F., & Poggio, T. (1990). Networks and the best approximation property. *Biological Cybernetics*, *63*, 169–176.

Kruskal, J. B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, *29*, 1–27.

Kruskal, J. B. (1964b). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, *29*, 115–129.

Lowrance, R. A., & Wagner, R. A. (1975). An extension of the string to string correction problem. *JACM*, *22*(2), 177–183.

Micchelli, C. A. (1986). Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, *2*, 11–22.

Muller, R. U. (1996). A quarter of century of place cells. *Neuron*, *17*, 813–822.

Nosofsky, R. M. (1986). Attention, similarity, and the identification–categorization relationship. *Journal of Experimental Psychology: General*, *115*, 39–57.

Nosofsky, R. M. (1992). Similarity scaling and cognitive process model. *Annual Review of Psychology*, *43*, 25–53.

O’Keefe, J., & Nadel, L. (1978). *Hippocampus as a cognitive map*. Oxford: Clarendon Press.

Okochi, M., & Sakai, T. (1982). Trapezoïdal dynamic programming matching with time reversibility. *Proceedings of the IEEE-ASSP Conference, Paris*, 1239–1242.

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1481–1497.

Poucet, B., Save, E., & Lenck-Santini, P.-P. (2000). Sensory and memory properties of hippocampal place cells. *Reviews in the Neurosciences*, *11*, 95–111.

Schoenberg, I. J. (1937). On certain metric spaces arising from euclidean spaces by a change of metric and their imbedding in Hilbert space. *Annals of Mathematics*, *38*(2), 787–793.

Schoenberg, I. J. (1938). Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, *44*, 522–536.

Shepard, R. N. (1962a). The analysis of proximities: Multidimensional scaling with an unknown distance function, I. *Psychometrika*, *27*, 125–140.

Shepard, R. N. (1962b). The analysis of proximities: Multidimensional scaling with an unknown distance function, II. *Psychometrika*, *27*, 219–246.

Shepard, R. N. (1986). Discrimination and generalization in identification and classification: Comment on Nosofsky. *Journal of Experimental Psychology: General*, *115*, 58–61.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science*, *237*, 1317–1323.

Vinstuk, T. K. (1968). Speech discrimination by dynamic programming. *Kibernetika*, *4*(1), 81–88.

Wagner, R. A., & Fischer, M. J. (1974). The string to string correction problem. *JACM*, *21*(1), 168–173.

Young, F. W., & Torgerson, W. S. (1967). Torsca, a Fortran IV program for Shepard–Kruskal multidimensional scaling analysis. *Behavioral Science*, *12*, 468.

II.C Modèles de codage d'images

Le problème du codage des images est étroitement lié au problème de la segmentation des images en formes identifiables, ainsi qu'au problème des invariants perceptifs, c'est-à-dire des transformations (géométriques ou autres) que peut subir le contenu d'une image tout en restant identifiable, sans donner lieu à une dégradation importante des performances. Il est communément admis que la perception visuelle possède un certain degré d'invariance par translation ou changement d'échelle des objets perçus. En fait, il semble que l'invariance par translation soit limitée à des éléments visuels simples et familiers composant l'image (Nazir & O'Regan, 1990). Ces transformations élémentaires sont faciles à modéliser, mais d'autres transformations affines comme les rotations et les symétries sont plus problématiques. De fait, la réalité perceptive des invariants par rotation ou symétrie n'est pas établie pour le cas général. D'un côté, on a de bonnes raisons de penser que la détection rapide d'éléments relativement simples, par exemple des formes caractérisant localement la présence d'un animal (oeil, bec, aile), est largement invariante par rotation de l'image (Guyonneau, Kirchner, & Thorpe, 2006). D'un autre côté, on sait par exemple que l'identification d'un visage est très fortement perturbée par un retournement vertical de l'image, et l'expérience commune montre qu'il est assez difficile de lire des mots présentés à l'envers. L'invariance perceptive par rotation/symétrie pourrait donc dépendre de la complexité des formes et de celle de la tâche. Le modèle de codage d'images par "Affine Moment Invariants" (AMI: Suk & Flusser, 2003, 2004) fournit une représentation vectorielle d'une image globale invariante par toute transformation affine, mais la vraisemblance perceptive de cette élégante approche n'a, à ma connaissance, jamais été évaluée. Toutefois, une invariance radicale, incluant rotations et symétries, n'est a priori pas compatible avec les observations concernant l'identification perceptive des formes complexes. De plus, il semble que la perception visuelle est en mesure de s'affranchir d'une large gamme de transformations naturelles non affines des images, telles que des flexions et ondulations (ex. effet du vent sur des végétaux, reflets dans l'eau mouvante), et l'on ne connaît pas de moyen de généraliser les AMI à ces sortes de transformations. J'ai donc développé une méthode de codage de données, plus particulièrement adaptée aux images, mais pas uniquement, qui permet de représenter et de comparer des formes globales sur la base d'une distribution de points dont la densité de probabilité reproduit la fonction de densité graphique de chaque image. Cette méthode permet d'obtenir des invariances par une large variété de transformations affines et non affines d'apparence assez naturelle, pourvu que ces transformations aient une matrice jacobienne

triangulaire, ce qui exclut les rotations, mais couvre toutes sortes de flexions et ondulations (Courrieu, 2006, 2007, articles ci-joints). L'article de 2006 présente une version très générale de la méthode, ainsi qu'une version parallèle réservée aux images, sous la forme d'un réseau de neurones feedforward. Cette deuxième version serait très rapide dans une implémentation effectivement parallèle, mais le calcul sur une machine conventionnelle est assez long. Comme la version générale n'est elle-même pas assez rapide pour certaines applications sur de grosses bases de données d'images, j'ai développé, dans l'article de 2007, une variante très rapide de la méthode pour l'encodage des images sur des ordinateurs conventionnels. Cette variante n'est pas aussi rapide que les techniques spécialisées d'indexation d'images (voir par exemple Glotin, Zhao, & Ayache, 2009), mais les temps de calcul sont du même ordre de grandeur et la gamme d'invariants est plus étendue.

References

Courrieu, P. (2006). Density codes and shape spaces. *Neural Networks, 19*, 429-445.

Courrieu, P. (2007). Fast density codes for image data. *Neural Information Processing - Letters and Reviews, 11(12)*, 247-255.

Glotin, H., Zhao, Z., & Ayache, S. (2009). Efficient image concept indexing by harmonic & arithmetic profiles entropy. *16th IEEE International Conference on Image Processing (ICIP)*, pp. 277-280. doi: 10.1109/ICIP.2009.5413350

Guyonneau, R., Kirchner, H., & Thorpe, S.J. (2006). Animals roll around the clock: The rotation invariance of ultrarapid visual processing. *Journal of Vision, 6*, 1008–1017.

Nazir, T.A., & O'Regan, J.K. (1990). Some results on translation invariance in the human visual system. *Spatial Vision, 5(2)*, 81-100.

Suk, T., & Flusser, J. (2003). Combined blur and affine moment invariants and their use in pattern recognition. *Pattern Recognition, 36*, 2895–2907.

Suk, T., & Flusser, J. (2004). Graph method for generating affine moment invariants. *Proceedings of the 17th International Conference on Pattern Recognition, 2*, 192-195.

Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 19 (2006) 429–445
Neural Networks
www.elsevier.com/locate/neunet

Density codes and shape spaces

Pierre Courrieu *

Laboratoire de Psychologie Cognitive, CNRS - UMR 6146, Université de Provence, Centre St Charles, Bat. 9, Case D, 3 Place Victor Hugo, 13331 Marseille Cedex 1, France

Received 30 September 2004; accepted 31 October 2005

Abstract

This paper presents an algorithm that allows for encoding probability density functions associated to samples of points of R^n. The resulting code is a sequence of points of R^n whose density function approximates that of the set of data points. However, contrarily to sampled data points, code points associated to two different density functions can be matched, which allows to efficiently compare such functions. Moreover, the comparison of two codes can be made invariant to a wide variety of geometrical transformations of the support coordinates, provided that the Jacobian matrix of the transformation be everywhere triangular, with a strictly positive diagonal. Such invariances are commonly encountered in visual shape recognition, for example. Thus, using this tool, one can build spaces of shapes that are suitable input spaces for pattern recognition and pattern analysis neural networks. Moreover, a parallel neural implementation of the encoding algorithm is available for 2D image data.

Keywords: Sets of points; Density functions; Pattern encoding; Geometrical invariants; Shape similarity

1. Introduction

There are applications in which the input provided to a neural network is not a point (real vector) but a set of unordered points of R^n. Specific methods for suitably encoding unordered, deterministic, fixed size sets of points have been proposed (Courrieu, 2001). However, one can frequently consider that a set of points results from random sampling governed by a particular density of probability on R^n. In this case, it is usually the density function (not the random sample) that is relevant for the application. For example, the set of black pixels in an image representing a written word, or an object whose detail can be subject to some random variation, can be considered as a random sample of points of R^2 generated by a density function that characterizes a recognizable shape (that of the word or of the object). However, many pattern recognition problems show that recognizable shapes can vary not only in a (limited) random way, but also in a large regular way. For example, depending on the used character font, or on the particular writer, a written word can substantially vary in width, in height, and in skewing, and its position in the image can also vary. Thus, a recognizable shape must, in fact, be characterized by a family of density functions depending on a set of regular transformations of the support. The set of transformations that does not affect the recognizable identity of a shape can conveniently be called 'invariance set'. Note that the invariance set depends on the considered shape: for example, the character 'x' is invariant to a 180° rotation, while the character 'b' is not (since it becomes 'q'). More globally, the set of letters (alphabet) is invariant to transformations such as translations, scaling, stretching, and skewing, but not to reflections (symmetries) or large orthogonal rotations (consider the subsets {b, d, p, q}, {u, n}, {f, t}, {N, Z}). The same seems to be true, in human vision, for more complex shapes such as written words: words are easily recognized with geometrical transformations that do not change their orientation, however, reading words in a mirror is quite uneasy for human readers, and psychologists hypothesized that reading inverted words requires a prior corrective mental rotation of the word image (Tzelgov & Henik, 1983). Various methods have been developed, in pattern recognition area, in order to encode shapes (usually 2D-shapes) invariant to certain affine transformations. Well-known methods are based on moment invariants (Heikkilä, 2004; Hu, 1962; Jin & Tianxu, 2004; Suk & Flusser, 2003), on Fourier descriptors (Arbter, Snyder, Burkhardt, & Hirzinger, 1990; Zhang & Lu, 2002), or on an analysis of characteristic contour points (landmarks) such as extreme points, or curvature maxima (Mokhtarian & Abbasi, 2002; Zhang, Zhang, Krim, & Walter, 2003). Density functions do not seem to be very popular in this context, despite the fact

* Tel.: +33 4 88 57 69 02; fax: +33 4 88 57 68 95.
E-mail address: courrieu@up.univ-mrs.fr

0893-6080/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2005.10.006

that densities can be simply approximated from sampled data (Cacoullos, 1966; Parzen, 1962; Specht, 1990). Although information geometry methods allow for computing similarities between parametrically described densities (Amari & Nagaoka, 2000), density functions are handicapped by the fact that one does not know any way of encoding them that allows for suitably comparing them in the context of shape recognition, and thus for defining spaces of densities as shape spaces (in a sense similar to Kendall, 1984). The purpose of the present work is to define an encoding method for density functions associated to samples of points of R^n, such that the codes associated to different densities can be compared, and a wide family of regular transformations can be simply reduced. As we shall see, there is a simple solution if we forgo reducing certain transformations such as reflections and orthogonal rotations.

2. Approximation of probability functions

Given a set of m sampled data points of R^n, there is a very simple way of approximating the density of probability function that governed the sampling: this is the well-known method of Parzen (1962), that have been extended by Cacoullos (1966), and Specht (1990). In summary, one chooses a n-dimensional kernel function (commonly a Gaussian density), together with a scale parameter (e.g. the standard deviation s for a Gaussian) that determines the degree of smoothing of the approximation, and one centers such a kernel at each of the data points. Then the approximated density at any point X is simply the arithmetic mean of the values of the m kernel functions at this point. One knows that, under very general conditions, this approximation converges in quadratic mean towards the true density as the sample size m tends to infinity. Other probability functions (cumulative, marginal, and conditional) can also be easily approximated if the n-dimensional kernel function is a product of n univariate kernels, and each univariate kernel can easily be integrated because it has a simple primitive function (e.g. logistic kernel, Cauchy kernel), or a suitable expansion (e.g. Gaussian kernel). We rapidly provide hereafter some usual formulas and notations that we must remember in the following sections.

Consider a set K of m data points of R^n

$$K = \{X_1, X_2, \ldots, X_m\}, \qquad X_j = (x_{1j}, x_{2j}, \ldots, x_{nj}), \quad j = 1 \ldots m.$$

An approximation of the density of probability at any point X of R^n is given by

$$f(X) = \frac{1}{m} \sum_{j=1}^{m} g(X; X_j, s),$$

where the kernel functions are simple probability density functions of the form

$$g(X; X_j, s) = \prod_{i=1}^{n} g_i(x_i; x_{ij}, s),$$

with, for example, in the case of a Gaussian kernel:

$$g_i(x_i; x_{ij}, s) = \frac{1}{s\sqrt{2\pi}} \mathrm{e}^{-(x_i - x_{ij})^2 / 2s^2}.$$

We use uppercase letters for integrative functions (or primitive functions if they exist):

$$G_i(a; x_{ij}, s) = \int_{-\infty}^{a} g_i(x; x_{ij}, s) \mathrm{d}x.$$

Given that the kernel functions are densities of probability, one has always:

$$G_i(\infty; x_{ij}, s) = 1.$$

Thus, for example, the $(n-k+1)$-dimensional marginal variable $(x_k, x_{k+1}, \ldots, x_n)$ has the density:

$$f(\bullet, \ldots, \bullet, x_k, x_{k+1}, \ldots, x_n) = \int_{R^{k-1}} f(x_1, \ldots, x_n) \mathrm{d}x_1 \ldots \mathrm{d}x_{k-1}$$

$$= \frac{1}{m} \sum_{j=1}^{m} \left(\prod_{i=1}^{k-1} G_i(\infty; x_{ij}, s) \right) \left(\prod_{i=k}^{n} g_i(x_i; x_{ij}, s) \right)$$

$$= \frac{1}{m} \sum_{j=1}^{m} \prod_{i=k}^{n} g_i(x_i; x_{ij}, s).$$

It is convenient here to use the following rule:

if $B < A$ then $\prod_{i=A}^{B} u_i = 1$, whatever be u_i.

The one-dimensional conditional variable $(x_k | x_{k+1} = a_{k+1}, \ldots, x_n = a_n)$ has the density:

$$f(x_k | x_{k+1} = a_{k+1}, \ldots, x_n = a_n) = \frac{f(\bullet, \ldots, \bullet, x_k, a_{k+1}, \ldots, a_n)}{f(\bullet, \ldots, \bullet, \bullet, a_{k+1}, \ldots, a_n)}$$

$$= \frac{\sum_{j=1}^{m} g_k(x_k; x_{kj}, s) \prod_{i=k+1}^{n} g_i(a_i; x_{ij}, s)}{\sum_{j=1}^{m} \prod_{i=k+1}^{n} g_i(a_i; x_{ij}, s)},$$

and the cumulative probability function of this conditional variable is given by:

$$\mathrm{Prob}(x_k \le b | x_{k+1} = a_{k+1}, \ldots, x_n = a_n)$$

$$= \int_{-\infty}^{b} \frac{f(\bullet, \ldots, \bullet, x_k, a_{k+1}, \ldots, a_n)}{f(\bullet, \ldots, \bullet, \bullet, a_{k+1}, \ldots, a_n)} \mathrm{d}x_k$$

$$= \frac{\sum_{j=1}^{m} G_k(b; x_{kj}, s) \prod_{i=k+1}^{n} g_i(a_i; x_{ij}, s)}{\sum_{j=1}^{m} \prod_{i=k+1}^{n} g_i(a_i; x_{ij}, s)}.$$

We use also a simplified notation for the above function

$$\mathrm{Prob}(x_k \le b | x_{k+1} = a_{k+1}, \ldots, x_n = a_n) = F(b | a_{k+1}, \ldots, a_n),$$

where the uppercase F recalls that the corresponding density function is f, and F could be replaced by H if the density function was named h.

3. Method foundations

The principle of the encoding method is simply a combination of the above probability function approximation techniques with a well-known method for generating pseudo-random data from a given probability law on R^n (see, e.g. Calot, 1967, pp. 185, 436–438). However, the use of these methods for encoding purposes requires some specific foundations.

3.1. Density encoding principle

Definition 1. Let $f(\mathrm{X})$ be a probability density function on R^n, one associates to f the mapping $P[f]$:

$$P[f](X) = (P_1[f](X), P_2[f](X), \ldots, P_n[f](X)),$$

with $P_n[f](X) = F(x_n) = \int_{-\infty}^{x_n} f(\bullet,\ldots,\bullet,t)\mathrm{d}t$, and $P_k[f](X) = \mathrm{F}(x_k|x_{k+1},\ldots,x_n)$, $1 \le k \le n-1$.

Theorem 1. *Assume that f is a continuous probability density function on R^n such that*

$$\|X\|_\infty < \infty \Rightarrow f(X) > 0, \quad X \in R^n,$$

then

(i) $P[f](\mathrm{X})$ *is a continuous bijection from* R^n *to* $(0, 1)^\mathrm{n}$,
(ii) the reciprocal bijection $P^{-1}[f](U)$*, with U uniformly distributed in* $(0, 1)^n$*, is distributed as X, with density f.*

Proof. If the density f is nowhere zero on R^n, then the densities of the conditional variables $(\mathrm{x_k}|\mathrm{x_{k+1}},\ldots,\mathrm{x_n})$, $1 \le \mathrm{k} \le \mathrm{n}$, are nowhere zero on R, which implies that the cumulative probability functions $\mathrm{F}(\mathrm{x_k}|\mathrm{x_{k+1}},\ldots,\mathrm{x_n})$, $1 \le \mathrm{k} \le \mathrm{n}$, are strictly increasing, thus they are everywhere invertible, and (i) is proven since these functions are the n components of $\mathrm{P}[f]$.

In what concerns (ii), with $P[f](X) = U \in (0, 1)^n$, one has:

$$P_n^{-1}[f](U) = F^{-1}(u_n) = x_n,$$

$$P_k^{-1}[f](U) = F^{-1}(u_k|u_{k+1},\ldots,u_n) = (x_k|x_{k+1},\ldots,x_n),$$

$$k = n-1, n-2, \ldots, 1.$$

If the n components u_k, $1 \le k \le n$, are independent variables uniformly distributed in (0, 1), then the conditional variable $(u_k|u_{k+1},\ldots,u_n)$ is simply distributed as u_k, that is uniformly in (0, 1), thus the density of $P_k^{-1}[f](U)$ is equal to $f(x_k|x_{k+1},\ldots,x_n)$, as a (well-known) consequence of applying an inverse cumulative function to a uniform variable in (0, 1). On the other hand, one can easily verify that the conditional variables $(x_k|x_{k+1},\ldots,x_n)$, $1 \le k \le n$, are independent from each other, since:

$$f((x_k|x_{k+1},\ldots,x_n) \text{ AND } (\bullet,\ldots,\bullet,x_{k+1},\ldots,x_n))$$

$$= f(\bullet,\ldots,\bullet,x_k,x_{k+1},\ldots,x_n)$$

$$= f(x_k|x_{k+1},\ldots,x_n) \cdot f(\bullet,\ldots,\bullet,x_{k+1},\ldots,x_n).$$

Thus the density of $P^{-1}[f](U)$ is equal to $\prod_{k=1}^{n} f(x_k|x_{k+1},\ldots,x_n) = f(X)$, which completes the proof. □

In practice, the conditions of Theorem 1 can always be fulfilled given that the densities to be encoded are in fact approximations obtained by a superposition of kernel functions (see Section 2), thus it suffices to use strictly positive continuous kernel functions that tend to zero asymptotically as variables tend to infinity, which is the case of common kernel functions.

Definition 2. Let f be a probability density function on R^n that fulfills the conditions of Theorem 1, and let $S_M = (U_1, U_2, \ldots, U_M)$ be a fixed sequence of points uniformly distributed in the open cube $(0, 1)^n$, then the sequence $(P^{-1}[f](U_j), j = 1\ldots M)$ is called a 'density code' associated to f.

Assume that we generated a fixed sequence of pseudo-random points $(U_1, U_2, \ldots, U_M)$ uniformly distributed in the open cube $(0, 1)^n$, and that we approximated a density function f from an empirical data set as described in Section 2. Then Theorem 1 implies that the sequence of points $(P^{-1}[f](U_j), j = 1\ldots M)$ is approximately distributed as the original data set, with density f. However, contrarily to the original data set, this is not a random sample made of any number of unordered points, but this is in fact a deterministic, fixed length sequence of M points, that is representative of the data distribution.

3.2. Relation between density codes

Now, assume that we approximated two densities, say f and h, from two distinct data sets, then we can obtain two density codes, $(P^{-1}[f](U_j), j = 1\ldots M)$ and $(P^{-1}[h](U_j), j = 1\ldots M)$, that can be compared to each other because the M points of each sequence are paired to the M points of the other one by the intermediate of the common U_j points. We need to know what is the meaning of such a matching, which is the object of Theorem 2.

Theorem 2. *Let $X \in R^n$ be a random variable whose density f(X) fulfills the conditions of Theorem 1. Let $Y \in R^n$ be another random variable functionally related to X by a continuous invertible transformation ψ such that $Y = \psi(X) = (\psi_1(X), \psi_2(X), \ldots, \psi_n(X))$, with:*

$$y_k = \psi_k(X) = \psi_k(x_k,\ldots,x_n), \quad \text{and} \quad \frac{\partial \psi_k}{\partial x_k}(X) > 0, \quad \forall\, X \in R^n.$$

In other words, one assumes that the kth component of Y only depends on the $n-k+1$ last components of X, and that it is a strictly increasing function of the kth component of X. If h denotes the density function of Y, and U is a variable in $(0, 1)^n$, then one has:

(i) $\mathrm{P[f](X) = P[h](Y)}$,
(ii) $\mathrm{P^{-1}[h] \circ P[f] = \psi}$,
(iii) $\mathrm{P^{-1}[h](U) = \psi(P^{-1}[f](U))}$.

Proof. The conditions of Theorem 2 imply that the Jacobian matrix $((\partial\psi_i/\partial x_j)(X))$ of the transformation $\psi(X)$, at any point X, is triangular and has a strictly positive diagonal. Thus, the Jacobian determinant of the transformation is nowhere zero, and it is given by:

$$J(\psi(X)) = \prod_{i=1}^{n} \frac{\partial \psi_i}{\partial x_i}(X) > 0.$$

On the other hand, given the functional relation between the two random variables, their density functions are related by:

$$h(Y) = f(X)J^{-1}(\psi(X)).$$

Considering the n components in decreasing order, one has for the nth one:

$$\begin{aligned} P_n[h](Y) &= \int_{-\infty}^{Y_n} \int_{R^{n-1}} h(\zeta_1,\ldots,\zeta_n)\mathrm{d}\zeta_1\ldots\mathrm{d}\zeta_n \\ &= \int_{-\infty}^{\psi_n^{-1}(Y)} \int_{R^{n-1}} f(\xi_1,\ldots,\xi_n)\cdot J^{-1}(\psi(\xi_1,\ldots,\xi_n))\cdot J(\psi(\xi_1,\ldots,\xi_n))\mathrm{d}\xi_1\ldots\mathrm{d}\xi_n \\ &= \int_{-\infty}^{x_n} \int_{R^{n-1}} f(\xi_1,\ldots,\xi_n)\mathrm{d}\xi_1\ldots\mathrm{d}\xi_n = P_n[f](X). \end{aligned}$$

For the remaining $n-1$ components, one obtains:

$$\begin{aligned} P_k[h](Y) &= \frac{\int_{-\infty}^{y_k} \int_{R^{k-1}} h(\zeta_1,\ldots,\zeta_k,y_{k+1},\ldots,y_n)\mathrm{d}\zeta_1\ldots\mathrm{d}\zeta_k}{\int_{R^k} h(\zeta_1,\ldots,\zeta_k,y_{k+1},\ldots,y_n)\mathrm{d}\zeta_1\ldots\mathrm{d}\zeta_k} \\ &= \frac{\int_{-\infty}^{\psi_k^{-1}(Y)} \int_{R^{k-1}} f(\xi_1,\ldots,\xi_k,x_{k+1},\ldots,x_n)\left(\prod_{i=1}^{k}\frac{\partial\psi_i}{\partial x_i}(\xi_i)\right)^{-1}\left(\prod_{i=k+1}^{n}\frac{\partial\psi_i}{\partial x_i}(x_i)\right)^{-1}\left(\prod_{i=1}^{k}\frac{\partial\psi_i}{\partial x_i}(\xi_i)\right)\mathrm{d}\xi_1\ldots\mathrm{d}\xi_k}{\int_{R^k} f(\xi_1,\ldots,\xi_k,x_{k+1},\ldots,x_n)\left(\prod_{i=1}^{k}\frac{\partial\psi_i}{\partial x_i}(\xi_i)\right)^{-1}\left(\prod_{i=k+1}^{n}\frac{\partial\psi_i}{\partial x_i}(x_i)\right)^{-1}\left(\prod_{i=1}^{k}\frac{\partial\psi_i}{\partial x_i}(\xi_i)\right)\mathrm{d}\xi_1\ldots\mathrm{d}\xi_k} \\ &= \frac{\int_{-\infty}^{x_k} \int_{R^{k-1}} f(\xi_1,\ldots,\xi_k,x_{k+1},\ldots,x_n)\mathrm{d}\xi_1\ldots\mathrm{d}\xi_k}{\int_{R^k} f(\xi_1,\ldots,\xi_k,x_{k+1},\ldots,x_n)\mathrm{d}\xi_1\ldots\mathrm{d}\xi_k} = P_k[f](X). \end{aligned}$$

This completes the proof of (i).

Set $P[f](X) = P[h](Y) = U$, then:

$$P^{-1}[h](P[f](X)) = P^{-1}[h](P[h](Y)) = Y = \psi(X),$$

which proves (ii). Moreover, after (ii) one has:

$$P^{-1}[h](U) = P^{-1}[h](P[f](P^{-1}[f](U))) = \psi(P^{-1}[f](U)),$$

which proves (iii), and then completes the proof of Theorem 2. □

Hence, if two random variables are related by a transformation ψ that fulfills the conditions of Theorem 2, then the corresponding density codes are related by $P^{-1}[h](U_j) = \psi(P^{-1}[f](U_j)),\quad j = 1\ldots M$. Of course, this equality holds for exact probability functions, while the approximations used in practice (see Section 2) can lead to some non-zero error.

3.3. Density code dissimilarity functions

Definition 3. Let Ψ be a set of transformation mappings of the form

$$\tau(X) = (\tau_1(X),\ldots,\tau_n(X)),\quad X \in R^n,$$

$$\tau_k(X) = \sum_{i=0}^{N} t_{ik} b_i(X),\quad t_{ik} \in R,\quad k = 1\ldots n,$$

where $\{b_i(X); 0 \le i \le N, X \in R^n\}$ is a given set of basis functions on R^n, and the matrix $T = (t_{ik}) \in R^{(N+1)\times n}$ depends on the data, as explained below. Consider two density codes $\xi = (X_j = P^{-1}[f] \times (U_j),\quad j = 1\ldots M)$ and $\zeta = (Y_j = P^{-1}[h](U_j),\quad j = 1\ldots M)$. Then one defines the 'dissimilarity' of the first density code from the second one, with invariance to the transformation family Ψ, as:

$$\delta_\Psi(\xi,\zeta) = \min_{\tau\in\Psi}\left(\frac{1}{M}\sum_{j=1}^{M}\|\tau(X_j) - Y_j\|^2\right)^{1/2}.$$

Consider the two above density codes, ξ and ζ, in the form of two $(M\times n)$ real matrices, and let $B_X = (b_i(X_j)), j = 1\ldots M, i = 0\ldots N$, be the $(M\times(N+1))$ matrix of the basis functions for ξ. Then, using a common least square method, one obtains

$$\delta_\Psi(\xi,\zeta) = \min_T\left(\frac{1}{\sqrt{M}}\|B_X T - \zeta\|\right) = \frac{1}{\sqrt{M}}\|B_X B_X^{[-1]}\zeta - \zeta\|,$$

where $B_X^{[-1]}$ is a pseudo-inverse matrix of B_X. If B_X is of rank $N+1$, then one has simply:

$$B_X^{[-1]} = (B_X^t B_X)^{-1} B_X^t,$$

else $B_X^{[-1]}$ is the Moore–Penrose generalized inverse of B_X (Ben Israel & Greville, 2003; Courrieu, 2005b). As a simple but useful example, if Ψ is the set of affine transformations, then the set of basis functions reduces to $\{b_0(X) = 1, b_1(X) = x_1,\ldots,b_n(X) = x_n\}$, $N = n$, and $B_X = [1,\ \xi]$. If two random variables, X and Y, are related by a transformation $Y = \psi(X)$ that fulfills the conditions of Theorem 2, and $\psi \in \Psi$, then $\delta_\Psi(\xi,\ \zeta) = 0$ (plus possibly an approximation error). One can also have $\delta_\Psi(\zeta,\xi) = 0$ iff $\psi^{-1} \in \Psi$,

which is the case for invertible affine transformations, but not for every family of transformations. As an example, the reciprocal function of an invertible algebraic polynomial is usually not an algebraic polynomial. Note that one has in general $\delta_\Psi(\xi, \zeta) \neq \delta_\Psi(\zeta, \xi)$ in the non-zero case, thus δ_Ψ is certainly not a distance, and it cannot be monotonically transformed into a distance (Courrieu, 2002). If one requires that $\delta_\Psi(\xi, \xi) = 0$, then Ψ must include at least the set of linear transformations, in order to make the identity transform available. Let Ω_M be the set of all density codes that can be generated from the same uniform sequence S_M (Definition 2), then the space (Ω_M, δ_Ψ) is a non-metric space whose topology is induced by δ_Ψ. In particular, a closed ball of center $X \in \Omega_M$, and of radius $r \geq 0$, can be defined as the set:

$$B(X,r) = \{Y \in \Omega_M, \delta_\Psi(X,Y) \leq r\}.$$

An open ball can obviously be defined in a similar way, by using strict inequalities ($>$ and $<$, instead of $\geq$ and $\leq$). Thus there is clearly a neighborhood system associated to (Ω_M, δ_Ψ). In the context of shape recognition problems, the space (Ω_M, δ_Ψ) can be called a 'shape space', and it can be used as an input space for networks that do not require metric input data, such as nearest neighbor classifiers, or non-metric data function approximators (Courrieu, 2005a).

4. Implementations

4.1. Computer implementation

The principles stated in Sections 2 and 3 can be used to develop practical algorithms for the computation of density codes associated to empirical data sets, and for their comparison. A Matlab program for the computation of density codes is presented in Appendices A.1–A.3, and a Matlab program for the computation of δ_Ψ, where Ψ is the set of algebraic multivariate polynomials, is presented in Appendix A.4, which allows Matlab users to test the tool and provides an easily readable model for practical implementations. Note, however, that this is an illustrative program for academic use only, since it has not been optimized with respect to reliability, speed and precision performance. The main elements of this program have been discussed above, however, we have not defined a practical way of generating a sequence of points uniformly distributed in $(0,1)^n$, nor a practical way of inversing the mapping $P[f]$. In what concerns this last point, we noted that each component of the mapping $P[f](X)$ is a strictly increasing, one-variable function (Definition 1 and Theorem 1), which means that we can apply simple usual methods (e.g. a bounding method) in order to find the inverse mapping $P^{-1}[f](U)$, at any point U. One must compute the components of the inverse mapping in decreasing order of their number. Since the last component of $P[f](X)$ is simply the cumulative probability function of the marginal variable x_n, it can be computed directly from u_n. The $(n-1)$th component of $P[f](X)$ is the cumulative probability function of the $(n-1)$th marginal variable, conditional to the value of the nth one that has just been computed, thus x_{n-1} can be computed from u_{n-1}, and the value of x_n. The situation is clearly similar for the remaining components in decreasing order, and x_{n-k} can be computed from u_{n-k}, given the previously computed values of $x_{n-k+1},\ldots,x_n$. The main function of the program in Appendix A.1 implements this principle, while the probability functions are approximated using the method of Section 2, and examples of kernels and their cumulative probability functions are given in the Matlab sub-functions of Appendix A2. In what concerns the generation of sequences of points uniformly distributed in $(0,1)^n$ (that is the S_M-sequences of Definition 2), we use quasi-uniform Faure sequences (Faure, 1982, 2001), that are known to provide low discrepancy, even in high dimension spaces. Original Faure (1982) sequences are in $[0, 1)^n$, thus we must exclude points that have at least one zero coordinate, in order to avoid infinite coordinate inverse mappings. This is implemented in the Matlab sub-functions of Appendix A3. In what concerns the choice of the number M of code points, we note that a choice of the form $M = q^k$, where k is a positive integer, and q is the smallest prime number greater or equal to the dimension n, allows to obtain complete Faure sequences. On the other hand, we note that the larger is M, the more precise is the density encoding. However, the densities that we encode are themselves approximated from data point sets. Thus a simple and reasonable strategy consists of choosing k such that $M = q^k$ is close to the mean number of data points per shape of the considered shape space. However, there are many cases where this number can be lowered because data samples are frequently larger than necessary in order to define shapes (see Section 5.2). Finally, note that other methods can be used for generating quasi-uniform sequences, such as, for example, the well-known Halton sequences (Halton, 1960). Halton's sequences work well in low dimension spaces, however, it is known that their uniformity properties are degraded as the dimension increases. Since the present encoding method must work for every finite dimension, we must clearly prefer Faure sequences.

4.2. Parallel implementation for 2D image data

The computer implementation described above is quite general, it allows the computation of density codes in any dimension (n), with various kernel functions, and with any arbitrary precision (via the parameter named 'Tol'). However, in the case of image encoding for perception modeling, one commonly requires that the encoding be fast in a neuron-like architecture, while the dimension of data is typically $n = 2$, and a 'gray level' type value can be associated to each data point. We describe hereafter a parallel implementation suitable for such data. This implementation requires a three-layered feed-forward architecture.

Layer 1. This is a bi-dimensional input layer $(x_j^{(1)}, y_j^{(1)})$, $j = 1\ldots m$, whose m cells correspond to image pixels, and whose activation function $a_j = a(x_j^{(1)}, y_j^{(1)})$ corresponds to a positive gray level function. The upper index in parenthesis indicates the number of the layer in which the coordinates are considered.

Layer 2. This layer computes the mapping $P[f](x,y)$ on a fixed set of points (possibly the same as those of the input layer). A positive gray level function can be considered as a weighting function, and the density approximation at any point $(x,y)\in R^2$ can be reformulated as

$$f(x,y) = \frac{\sum_{j=1}^{m} a_j g((x,y);(x_j^{(1)},y_j^{(1)},s)}{\sum_{j=1}^{m} a_j},$$

which implies that:

$$P_2[f](x,y) = F(y) = \frac{\sum_{j=1}^{m} a_j G_2(y;y_j^{(1)},s)}{\sum_{j=1}^{m} a_j},$$

$$P_1[f](x,y) = F(x|y) = \frac{\sum_{j=1}^{m} a_j G_1(x;x_j^{(1)},s) g_2(y;y_j^{(1)},s)}{\sum_{j=1}^{m} a_j g_2(y;y_j^{(1)},s)}.$$

Note that, if $a(x,y)\in\{0,1\}$, $\forall(x,y)\in R^2$, then the above formulation is equivalent to that of Section 2. The second layer is made of two arrays of cells: a one-dimensional array of $N2$ cells that compute $P_2[f]$ at $N2$ distinct fixed values of the y coordinate $(y_1^{(2)},\ldots,y_{N2}^{(2)})$, and a bi-dimensional $(N2\times N1)$ array of cells that compute $P_1[f]$ at $N1$ distinct fixed values of the x coordinate $(x_1^{(2)},\ldots,x_{N1}^{(2)})$, for the $N2$ fixed values of y. Thus, each cell of the first array is characterized by a particular y coordinate, and each cell of the second array is characterized by particular x and y coordinates. The computation of $P[f]$ as defined above implies that each cell of Layer 2 receives connections from all cells of Layer 1, however, the only quantities that vary from an input to another one are the activation values (a_j).

Layer 3. This is an output layer that approximates the density code $P^{-1}[f](u,v)$ on a fixed set of M points uniformly distributed in $(0,\ 1)^2$. Each cell of the third layer is characterized by a particular point $(u_i,\ v_i)\in(0,\ 1)^2$, $i=1\ldots M$, and an output variable (x or y coordinate). Since two output coordinates are required in all cases, it is convenient to consider that each unit of Layer 3 is made of two cells that share the same $(u_i,\ v_i)$ characteristic point. Each unit of Layer 3 computes a quantity of the form:

$$P^{-1}[f](u_i,v_i) \approx \frac{\sum_{k=1}^{N2}\sum_{j=1}^{N1}\left(x_j^{(2)},y_k^{(2)}\right)\exp\left(-\beta\left\|(u_i,v_i)-\left(P_1[f]\left(x_j^{(2)},y_k^{(2)}\right),P_2[f]\left(\bullet,y_k^{(2)}\right)\right)\right\|\right)}{\sum_{k=1}^{N2}\sum_{j=1}^{N1}\exp\left(-\beta\left\|(u_i,v_i)-\left(P_1[f]\left(x_j^{(2)},y_k^{(2)}\right),P_2[f]\left(\bullet,y_k^{(2)}\right)\right)\right\|\right)},$$

where the values of $P_1[f]$ and $P_2[f]$ are provided, respectively by the bi-dimensional and the one-dimensional array of cells of Layer 2, and the coordinates $(x_j^{(2)},y_k^{(2)})$ are used as synaptic weights. As the parameter β tends to infinity, the above expression tends to the point of coordinates $(x_j^{(2)},y_k^{(2)})$ whose $P[f]$ mapping is the closest to the point (u_i,v_i), which provides the desired approximation of $P^{-1}[f](u_i,v_i)$. Note that this approximation is based on a MIN-like variant of the MAX-like operation of Riesenhuber & Poggio (1999). Appendix A5 presents a Matlab program for computer simulations of the above parallel implementation.

5. Illustrative examples

5.1. Pseudo-handwriting data, affine transformations, and orthographic similarity

In this section, we use, as input data, the six binary images of pseudo-handwritten words presented in Fig. 1. Left column images (we refer to as 'word-1', 'work-1', and 'play-1') are resampled affine transforms (by scaling, stretching and skewing) of right column images (we refer to as 'word-2', 'work-2', and 'play-2'). On the other hand, the first two words ('word' and 'work') are orthographic neighbors, while the third word ('play') has no common letter with the two previous ones. Intuitively, a good dissimilarity measure must provide a minimum (possibly zero) difference between left column images and their corresponding right column images (0-letter

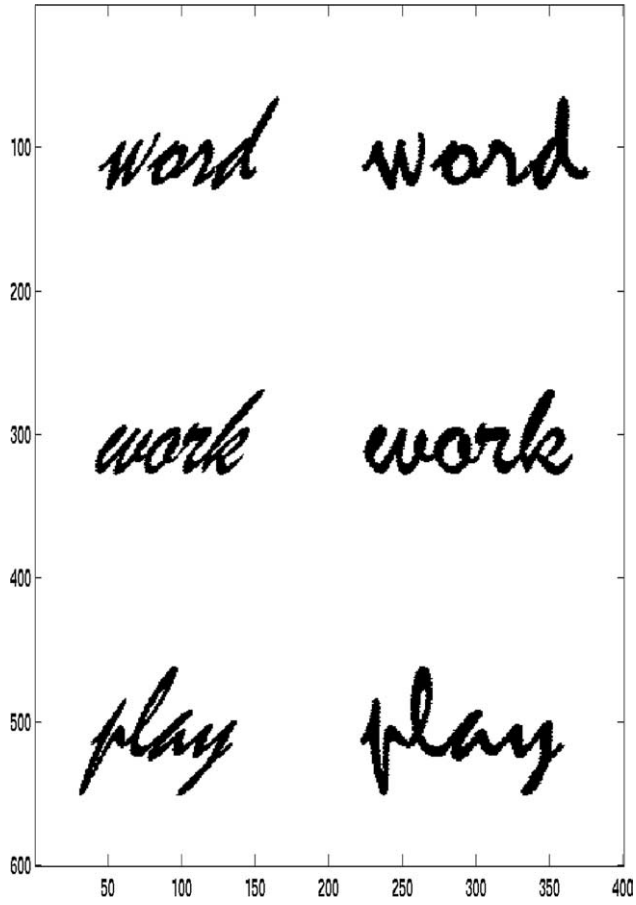


Fig. 1. Images of three words in two graphic versions (pseudo-handwriting). Left column versions (1) differ from right column versions (2) by scaling, stretching and skewing transformations. The first two words differ by only one letter (orthographic neighbors), while the third one has no common letter with them.

difference), and the image dissimilarity must increase as the number of different letters between words increases, independently of affine transformations. In other words, we expect the shape space to have a topology compatible with an abstract orthographic analysis. In order to test these expectations, we must compare various dissimilarity measures, and since we do not know, a priori, how the dissimilarity measures are distributed, we must use non-parametric (rank-based) statistical tests. In the following, we use Mann–Whitney tests for independent data samples, and we use Friedman tests for paired data samples.

5.1.1. Density codes and the effect of S_M-sequences

In this section, we test the above expectations together with the effect of particular S_M-sequences (Faure sequences *vs.* Halton sequences) on the performance of density codes. Since we work in a low dimension space ($n=2$), the two types of S_M-sequences must provide very similar performance, showing that the algorithm is robust with respect to the choice of particular quasi-uniform sequences. Density codes have been computed by the program described in Section 4.1, and by a variant where the Faure sequence was replaced by a Halton sequence. An input data matrix can be obtained from a binary image matrix (say 'image'), with 0 code for black pixels, by the Matlab function $[y, x]=\text{find}(\text{image}==0)$, and $X=[x, y]$. For each type of S_M-sequence and each image, a density code of 4096 points has been generated, using Gaussian kernels with a standard deviation of 0.3 pixels, and a tolerance error of about 0.5 pixels. Then each code has been segmented into two distinct but equivalent density codes of $M=2048$ points, that have been used for computing the δ_Ψ values, and the equivalent δ_Ψ values have been averaged. Fig. 2 shows six distributions of 2048 density code points corresponding to the data of Fig. 1, for Faure S_M-sequences (upper half-figure), and Halton S_M-sequences (lower half-figure). As one can see in Fig. 2, the distributions of density code points are closely similar to the corresponding data distributions, as Theorem 1 implies, whatever be the considered S_M-sequence type.

Table 1 shows the averaged δ_Ψ values, where Ψ is the set of affine transformations (i.e. degree one polynomial transformations), for all pairs of images of Fig. 1, for Faure S_M-sequences (upper sub-table), and Halton S_M-sequences (lower sub-table). At the bottom of each sub-table, we present a comparison (means, standard deviations, and Mann–Whitney tests) between the δ_Ψ functions of pairs of distinct images of words differing by 0, 1, and 4 letters. As one can see, the three δ_Ψ value distributions are ordered and well separated as expected, whatever be the considered S_M-sequence type. Moreover, the linear correlation coefficient between the 30 out-diagonal δ_Ψ values of the two sub-tables is $r=0.9995$, and Friedman's test provides $\chi^2(1)=0.5333$, n.s. Clearly, the two types of S_M-sequences provide very similar results, as expected.

5.1.2. Comparison with Affine Moment Invariants based methods

One of the most attractive ways of encoding shapes with invariance to certain transformations is probably the approach

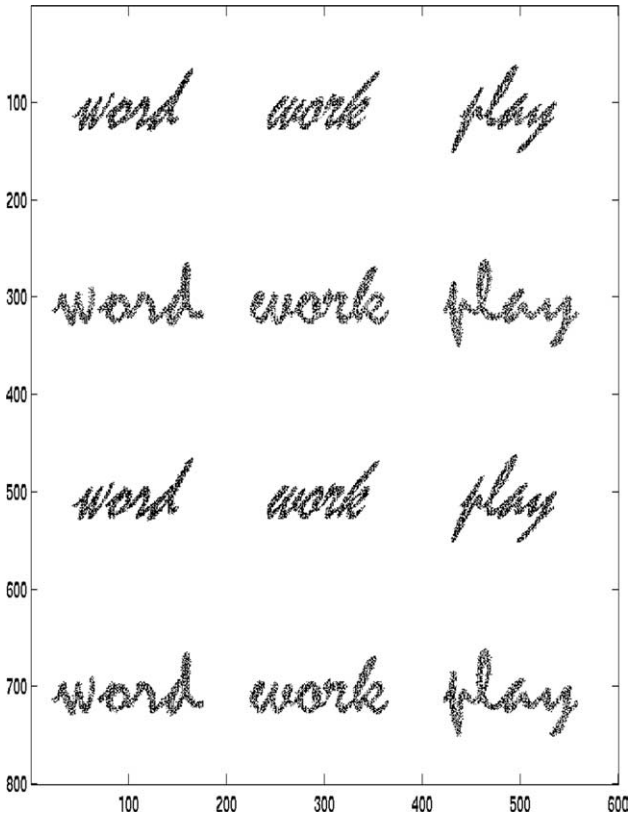


Fig. 2. Spatial repartition of density code points associated to the six images of Fig. 1, using a Faure S_M-sequence (upper half-figure), and a Halton S_M-sequence (lower half-figure). The data density functions were approximated using a Gaussian kernel, with a scale of 0.3 pixels, and each code is a sequence of $M=2048$ points.

based on Affine Moment Invariants, which was initiated by Hu (1962), and that has recently been considerably improved (Heikkilä, 2004; Jin & Tianxu, 2004; Suk & Flusser, 2003). Most of the tools described in this area, as well as Fourier descriptors (Zhang & Lu, 2002), only provide invariance to affine transformations of rigid object images (translation, scale change, and rotation), however, certain algorithms provide invariance to more general affine transformations, including stretching and skewing (Heikkilä, 2004; Suk & Flusser, 2003). To date, affine moment invariants are described for two-dimensional data only (mainly image data). On the other hand, density codes described in this paper can work with data of any dimension, and allow to reduce transformations that are not necessarily affine, but with the restriction that the Jacobian matrix of the transformation must be triangular with strictly positive diagonal (Theorem 2), which excludes rotation and reflection transformations. As one can see, the application fields of the various tools are not the same, although they have a non-empty intersection. In this section, we use the data of Fig. 1 to compare density code performance to that of Combined Blur and Affine moment Invariants (CBAIs) proposed by Suk and Flusser (2003), and to that of affine moments based matching method of Heikkilä (2004). Table 2 shows Euclidean distances between the six images of Fig. 1 in the space of the six available CBAIs (Suk & Flusser, 2003),

Table 1
Affinely minimized δ_Ψ functions between density codes of the six images of Fig. 1

$\delta_\psi(\downarrow,\rightarrow)$	word-1	word-2	work-1	work-2	play-1	play-2
			Faure S_M-sequence			
word-1	0.0000	4.1362	6.3857	8.8726	14.0711	19.2374
word-2	2.9027	0.0000	6.3974	8.8101	13.9922	19.1730
work-1	6.8646	9.5150	0.0000	2.5591	13.4056	18.4914
work-2	6.9050	9.5031	1.8074	0.0000	13.5776	18.8269
play-1	15.8377	21.7298	13.9026	19.3930	0.0000	5.9511
play-2	15.6800	21.5323	13.7298	19.1829	4.1776	0.0000
Differ	0 letters			1 letter		4 letters
mean $\delta\psi$	3.59 ($\pm$1.5) [MW(6,8)=21, $p<0.5$]			7.91 ($\pm$1.4) [MW (8,16)=36, $p<.05$]		16.99 ($\pm$3.0)
			Halton S_M-sequence			
word-1	0.0000	3.7918	6.2827	8.6296	14.1536	19.3849
word-2	2.6674	0.0000	6.3201	8.6807	14.0860	19.3339
work-1	6.7624	9.4031	0.0000	3.0156	13.4681	18.6299
work-2	6.7381	9.3763	2.1326	0.0000	13.6119	18.9303
play-1	15.9097	21.8506	13.9325	19.3818	0.0000	5.3871
play-2	15.8130	21.7443	13.8273	19.2629	3.7959	0.0000
Differ	0 letters			1 letter		4 letters
mean $\delta\psi$	3.47 ($\pm$1.1) [MW(6,8)=21, $p<.05$]			7.77 ($\pm$1.4) [MW (8,16)=36, $p<.05$]		17.08 ($\pm$3.0)

δ_Ψ values are averaged for two distinct Faure S_M-sequences (upper sub-table), and two distinct Halton S_M-sequences (lower sub-table), with $M=2048$. At the bottom of each sub-table, comparisons (means, standard deviations, and Mann–Whitney tests) between the δ_Ψ functions of pairs of distinct images of words differing by 0, 1, and 4 letters. The linear correlation coefficient between the 30 out-diagonal δ_Ψ values of the two sub-tables is $r=0.9995$, and Friedman's test provides $\chi^2(1)=0.5333$, n.s.

where each CBAI has been normalized (i.e. weighted by its inverse empirical standard deviation). At the bottom of the table, we present comparisons between the CBAI distances of pairs of distinct images of words differing by 0, 1, and 4 letters, as we did in Table 1 for density code dissimilarities. Mean CBAI distances are ordered as expected, however, the distributions corresponding to 0-letter and 1-letter differences are not well separated (non-significant Mann–Whitney test), and one can see their overlapping in the distance table, where the CBAI distance between 'word-1' and 'word-2' (0-letter difference) is greater than all 1-letter difference CBAI distances. Table 3 presents the Hausdorff distances provided by Heikkilä (2004)'s matching algorithm between the six images of Fig. 1. At the bottom of the table, one can see that the mean distances for 0, 1, and 4 letter differences are also ordered as expected, however, the distributions corresponding to 1-letter and 4-letter differences are not well separated (non-significant Mann–Whitney test), and one can see their overlapping in the distance table. Comparing these results to those of Table 1, we can conclude that density code dissimilarity functions provide a performance at least as good as those of the tested affine moment based algorithms, in the common part of their respective application domains.

5.2. Effects of code length and kernel scale

The required number of code points (M) can depend on the complexity of the shapes and on the relative scale of relevant distinctive details in the considered shape space. On the other hand, the required kernel scale (s) depends on the spacing of data points, and there are well-known heuristics that allow for estimating this scale. As an example, the 'global first nearest-neighbor heuristic' (Moody & Darken, 1989) is implemented as the default choice for s in the program of Appendix A.1. In this section, we test the robustness of the proposed encoding method across variations of M and s, using the data presented in

Table 2
Euclidean distances between the six images of Fig. 1 in the space of normalized CBAIs (Suk & Flusser, 2003)

$d(\downarrow,\rightarrow)$	word-1	word-2	work-1	work-2	play-1	play-2
word-1	0.0000	3.0444	2.7607	2.6100	3.9205	4.8519
word-2	3.0444	0.0000	2.9648	2.7556	3.3761	4.5974
work-1	2.7607	2.9648	0.0000	0.3308	2.9449	4.8048
work-2	2.6100	2.7556	0.3308	0.0000	2.9169	4.6587
play-1	3.9205	3.3761	2.9449	2.9169	0.0000	2.5279
play-2	4.8519	4.5974	4.8048	4.6587	2.5279	0.0000
Differ	0 letters		1 letter			4 letters
mean d	1.97 ($\pm$1.3) [MW(6,8)=37, n.s.]		2.77 ($\pm$0.1) [MW (8,16)=44, $p<.05$]			4.01 ($\pm$0.8)

At the bottom of the table, comparisons between the distances of pairs of distinct images of words differing by 0, 1, and 4 letters are provided.

Table 3
Dissimilarities between the six images of Fig. 1 in the space generated by Heikkilä (2004)'s matching algorithm

$D(\downarrow,\rightarrow)$	word-1	word-2	work-1	work-2	play-1	play-2
word-1	0	1.5124	19.3769	18.1033	32.4786	27.6237
word-2	1.5968	0	16.3696	17.8982	32.2108	28.1221
work-1	24.8532	17.3045	0	1.5326	22.4767	24.8052
work-2	17.6128	18.4146	1.5732	0	22.5561	16.5242
play-1	27.7808	13.7445	13.9387	22.7215	0	1.9775
play-2	26.6723	15.1146	20.1152	11.4153	1.0259	0
Differ	0 letters		1 letter			4 letters
mean D	1.54 (±0.3) [MW(6,8)=21, $p<.05$]		18.74 (±2.6) [MW (8,16)=80, n.s.]			22.39 (±6.7)

At the bottom of the table, comparisons between the dissimilarities of pairs of distinct images of words differing by 0, 1, and 4 letters are provided.

Fig. 3. The four left column shapes were obtained using a threshold function on one color component of numerical photos of flowers, in order to separate the flower shapes from their background, and to provide suitable data points. The number of data points (black pixels) per shape ranges from 5279 to 7885. The middle column data were obtained by an affine transformation (stretching and skewing) of the left column data. The number of data points per shape ranges from 2911 to 4201. The data in the right column were obtained by a non-affine transformation of left column data. The number of data points per shape ranges from 3774 to 5585. For all shapes, the default kernel scale provided by the 'global first nearest-neighbor heuristic' was very close to 1 (range: 1–1.04). In the experiment, we used four kernel scales ($s=0.5, 1, 2, 4$), with logistic kernel functions, that allow faster computation of $P[f]$ mappings than Gaussian kernels. In what concerns the code length, we varied M from 2^2 to 2^{12}. In order to do this, each of the 12 shapes was encoded with 4096 code points, for each value of s. For each code length, the 4096 point sequence was segmented into 2^{12-k} subsequences of length $M=2^k$, $k=2\ldots12$. Then relevant δ_Ψ functions (where Ψ is the set of affine transformations) were computed using all these subsequences as arguments and averaged. Dissimilarities were computed between Fig. 3 left column shapes and the corresponding middle column shapes (with the two argument orders), which provided δ_Ψ values for the 'affinely similar' condition. Dissimilarities were also computed between Fig. 3 left column shapes and the corresponding right column shapes, which provided δ_Ψ values for the 'non-affinely similar' condition. Finally, dissimilarities were computed for all pairs of distinct shapes in the first column of Fig. 3, which provided δ_Ψ values for the 'unrelated' condition. Since Ψ is the set of affine transformations, we expect small dissimilarities for the 'affinely similar' condition. As in Section 5.2, we expect intermediate dissimilarities for similarities that are not in Ψ, that is for the 'non-affinely similar' condition. Finally, we expect maximum dissimilarities for the 'unrelated' condition. Results are summarized in Fig. 4, where one can see means and standard deviations of δ_Ψ in the three conditions, for all M and s values. In all cases, δ_Ψ distributions are ordered as expected, they reach stable mean values and are well separated with $M\geq2^6$ code points, which is much less than the number of data points per shape. Increasing the kernel scale tends to lower large dissimilarities, however, the proposed default s (global, first nearest-neighbor heuristic) is visibly close to an optimum in this experiment, and it is not necessary to have a very precise estimation of this parameter.

5.3. Three-dimensional shapes and non-affine transformations

In this section, we illustrate density code capabilities with the example of random three-dimensional thread-like shapes as those presented in Fig. 5, with random third degree polynomial transformations whose Jacobian matrices are everywhere triangular with strictly positive diagonals (as required by Theorem 2). As Fig. 5 suggests, such transformations are suitable for modeling soft object spaces. We computed the

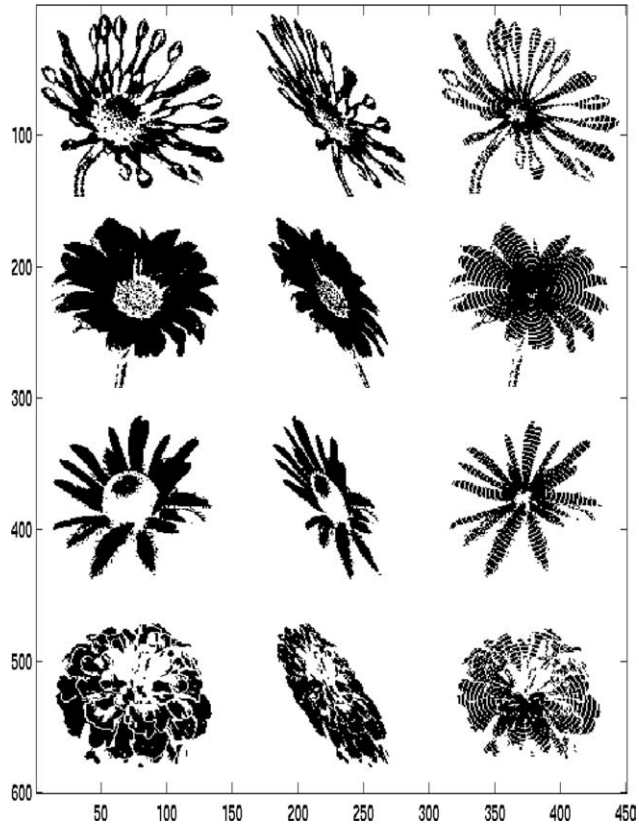


Fig. 3. Four thresholded photos of flowers (left column), affinely similar data (middle column), and non-affinely similar data (right column). The original photos are available at http://www.pdphoto.org/ (public domain).

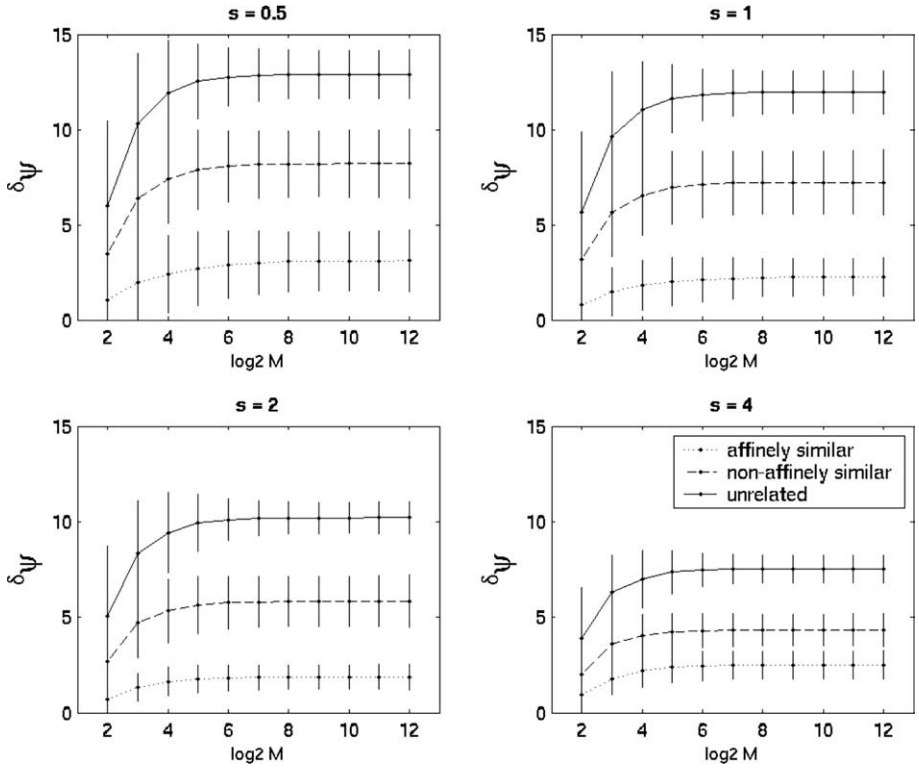


Fig. 4. Averaged affinely minimized dissimilarities (vertical bars mark standard deviations) of affinely similar, non-affinely similar, and unrelated flower data (see Fig. 3), as a function of the code size (for $M = 4$–4096), and of the scale parameter of logistic kernels ($s = 0.5, 1, 2, 4$). The default scale parameters, as provided by the 'global first nearest-neighbour heuristic' (Moody & Darken, 1989), ranged from 1 to 1.04.

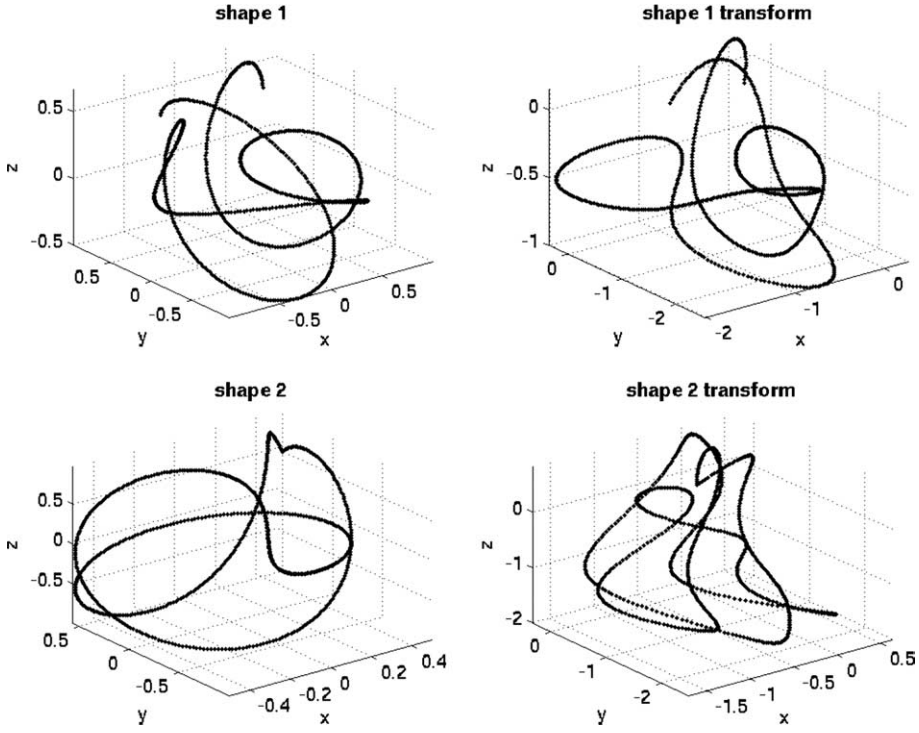


Fig. 5. Two examples of random 3D shapes (left column) and their random third degree polynomial transformations (right column). These soft transformations have triangular Jacobian matrices with positive diagonals everywhere.

density codes of these shapes using $M=3^6=729$ code points (in R^3), with logistic kernel functions. We used the default scale and tolerance parameters provided by the 'DensityCode' program (see Appendix A.1). The dissimilarity values (δ_Ψ) have been computed using the 'DeltaPoly' function (Appendix A4), where the optimal degree parameter for this example is $d=3$. For the examples of Fig. 5, the obtained dissimilarities of shape 1 and shape 2 from their transforms are 0.22 and 0.14, while the dissimilarities of the transforms from the source shapes are 0.25 and 0.16. In contrast, the dissimilarities between distinct shapes ranged from 0.36 to 0.47. Thus, despite the fact that the inverse of a third degree polynomial is not a third degree polynomial, the dissimilarity function correctly separated transformation cases from shape difference cases, even with non-optimally ordered arguments (which leads only to a small penalty).

One can of course ask whether this result generalizes to large sets of shapes and transformations. One can also ask what happens if one does not choose the optimal transformation degree for the computation of δ_Ψ, since in real life problems, one does not necessarily know a priori the appropriate transformation family. In order to answer these questions, we performed the following experiment. First, we generated a set of 80 random source shapes, each one consisting of 730 sampled points of a random 3D trigonometric parametric curve, as the shapes in the left column of Fig. 5. These shapes were randomly paired, resulting in 40 pairs (S_i, S_j) of source shapes. Then for each pair, a random third degree polynomial transformation (that fulfilled the requirements of Theorem 2) was applied to the two shapes, resulting in a pair (T_i, T_j) of transforms, as in the right column of Fig. 5. To each source shape, say S_i, one associated four dissimilarity measures: $\delta_\Psi(S_i, T_i)$, $\delta_\Psi(T_j, T_i)$, $\delta_\Psi(T_i, S_i)$, and $\delta_\Psi(S_j, S_i)$, where the density codes are denoted in the same way as the corresponding data (there is no ambiguity here). Since we must compare dissimilarity measures, we must take care that the scale of a dissimilarity is relative to the scale of its second argument, since the first argument is transformed in order to approximate the second one (see Section 3.3). Given that source shapes and their transforms do not have necessarily equal scales, one must avoid comparisons of dissimilarities that do not have the same second argument, however, one can reliably compare the first two or the last two dissimilarities listed above. Thus, we compare the dissimilarity of a shape from its transform to the dissimilarity of transforms of two different source shapes, and we compare the dissimilarity of a transform from its source shape to the dissimilarity of two different source shapes. In the first type of comparison, we expect that the first dissimilarity is systematically lower than the second one, because transforms of two different source shapes are not related by a tractable transformation. If we obtain a similar inequality for the second type of comparison, this means that one can reasonably approximate an inverse transformation in the same polynomial set as the direct transformation (strictly speaking, this is true only for degree one transformations). Finally, for all shapes, we varied the degree of the polynomial set Ψ (used in the computation of δ_Ψ) from 1 to 5, and we added the pseudo-degree 0, for which Ψ reduces to the identity transform (remember that the actual degree of shape transformations is 3). Results of this experiment are summarized in Table 4, where one can see the mean δ_Ψ (and standard deviation) for each experimental condition, together with Friedman's $\chi^2(1)$ tests for relevant comparisons. All tested differences are highly significant ($p<0.001$) and in the expected direction, except for the pseudo-degree 0 (no invariance). Thus, all Ψ polynomial sets, from degree 1 to degree 5, provide δ_Ψ functions that suitably account for degree 3 similarities, whatever be the order of the arguments provided to the dissimilarity function. These results clearly suggest that the tool is robust with respect to the approximation of Ψ. One can also observe that all dissimilarities decrease as the degree of Ψ increases, which is normal since the approximation capability of a polynomial tends to increase as the number of polynomial's terms increases. In a limit case, if the number of terms equals M, then all dissimilarities equal zero, whatever be the arguments, but this is of course meaningless.

5.4. An example of neural image encoding

The parallel implementation of the density encoding algorithm described in Section 4.2, for image data, implies a fixed discrete sampling of the space of variables and an approximation of code points. This can result in a loss of precision with respect to the standard algorithm, and thus, we must verify that there is no dramatic degradation of performance. In order to do this, we choose the example of

Table 4
Means (with standard deviations) of dissimilarity values of 3D source shapes (S_i, S_j), and their third degree polynomial transforms (T_i, T_j), as a function of the degree of the polynomial set Ψ, and of the type of matching

Ψ degree	$\delta_\Psi(S_i, T_i)$	$\chi^2(1)$	$\delta_\Psi(T_j, T_i)$	$\delta_\Psi(T_i, S_i)$	$\chi^2(1)$	$\delta_\Psi(S_j, S_i)$
0	1.32 (0.31)	8.45	1.19 (0.31)	1.32 (0.31)	57.80	0.88 (0.13)
1	0.39 (0.13)	76.05	0.69 (0.20)	0.30 (0.09)	76.05	0.53 (0.11)
2	0.29 (0.12)	80.00	0.59 (0.19)	0.23 (0.08)	80.00	0.47 (0.10)
3	0.23 (0.12)	76.05	0.51 (0.17)	0.21 (0.08)	80.00	0.41 (0.09)
4	0.22 (0.12)	76.05	0.46 (0.15)	0.19 (0.07)	80.00	0.37 (0.09)
5	0.21 (0.11)	76.05	0.41 (0.14)	0.18 (0.07)	80.00	0.34 (0.08)

Friedman's $\chi^2(1)$ tests are all significant.

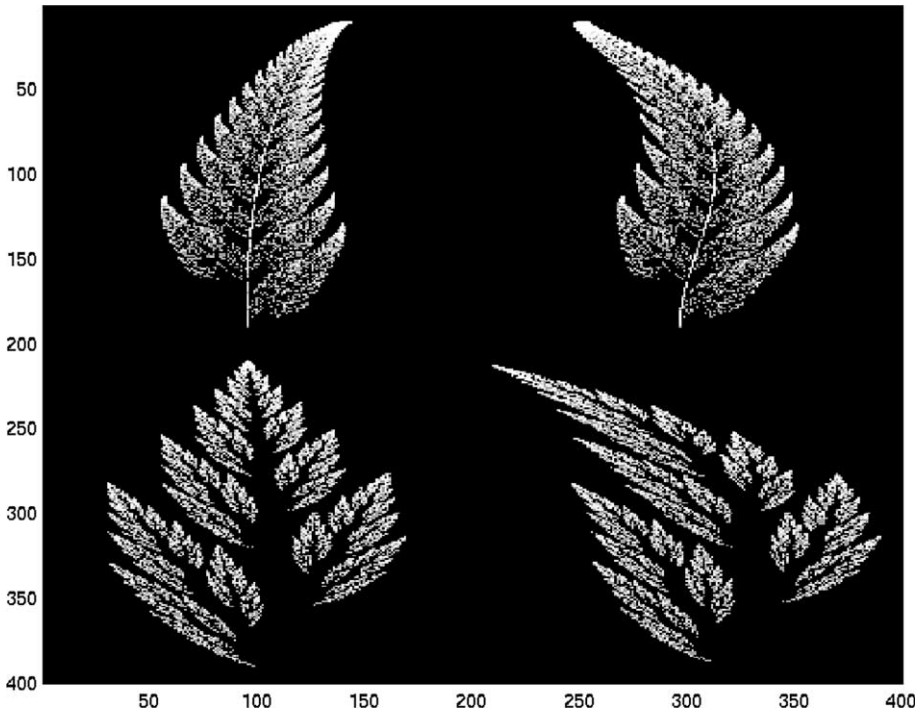


Fig. 6. Two blurred 'vegetable-like' fractals (left-top = 'fern', left-bottom = 'bush'), and their 'wind-like' transformations (right column).

the four images shown in Fig. 6. These are 200×200 gray level pixel images, that are blurred 'vegetable-like' fractals (left column), subject to a 'wind-like' transformation (right column), that is in fact a third degree polynomial transformation that fulfills Theorem 2 requirements. For convenience, the fractal at the top of the figure will be called 'fern', and the fractal at the bottom of the figure will be called 'bush'. The parallel encoding algorithm (Appendix A5 program) has been applied to the four images, with $M = 2048$ code points, logistic kernels with scale parameter $s = 0.3$ pixels, and $\beta = 5000$. For comparison, the standard encoding algorithm (Appendix A.1 program) has been applied to the set of coordinates of all non-black pixels from each image (in Matlab: [y,x] = find(image > 0), $X = [x,y]$), with the same parameters as for the parallel version, except β that was replaced by the default tolerance. Resulting δ_Ψ values, where Ψ is the set of third degree polynomial transformations, are reported in Table 5. As one can see, the loss of precision in the parallel implementation results in a global 10% increase of non-zero dissimilarities, however, all important relations between dissimilarity values are preserved, that is, each of the vegetables can be recognized and discriminated from the other one, whatever be the wind conditions.

Finally, take care that the simulation of a parallel process on a sequential processor can lead to some uncomfortable computation time. As an indication, the encoding of a 200×200 image by the program of Appendix A5, on a common personal computer, requires about 67 min, while the encoding by the standard implementation requires about 6 min and 45 s for about 4300 data points. However, the important fact, for modeling natural perceptual processes, or for specialized hardware development, is that the image encoding can be

Table 5
Comparison of δ_Ψ values on density codes provided by the parallel version and the standard version of the encoding algorithm, for the four images of Fig. 6

$\delta_\Psi(\downarrow,\rightarrow)$	Fern	Fern/wind	Bush	Bush/wind
Simulated parallel implementation				
Fern	0.0000	2.9899	10.1709	9.8703
Fern/wind	2.9319	0.0000	10.0721	9.9167
Bush	6.2984	6.3261	0.0000	5.4999
Bush/wind	6.2787	6.3934	5.5801	0.0000
Standard implementation				
Fern	0.0000	1.5961	9.7007	9.2874
Fern/wind	1.6042	0.0000	9.2206	9.0129
Bush	5.9770	5.8693	0.0000	4.9764
Bush/wind	6.0729	6.0519	5.1855	0.0000

computed by a three-layered feed-forward neural network, thus very fast. Note also that there are simple recurrent neural networks that can solve least square systems (in order to compute δ_Ψ functions) in a short time compatible with known performance of biological perceptual systems (Courrieu, 2004).

6. Conclusion

We have defined a tool that allows for encoding any set of data points of R^n in the form of a fixed length, deterministic sequence of points of R^n, whose density function approximates the (unknown) data probability density function. Such sequences are called 'density codes', and they can be used as elements of a 'shape space' whose topology is induced by a suitable dissimilarity function. Contrarily to data points, code points associated to different data sets can be suitably matched, which allows to define dissimilarity functions that can be made invariant to any set of coordinate transformations whose Jacobian matrices are triangular and have strictly positive diagonals. This is a restriction with respect to other methods such as affine moment invariants (Suk & Flusser, 2003), or affine moment based matching (Heikkilä, 2004), for comparisons of two-dimensional data subject to affine transformations. However, the present method is clearly not limited to affine transformations, and it works for any data space dimension. Moreover, intractable transformations such as reflections and orthogonal rotations are known to be problematic for visual shape recognition by human as well. In the perspective of modeling perceptual processes, a parallel neural implementation for encoding image data is proposed. More generally, shape spaces, as they are defined here, can be used as input spaces for pattern analysis and pattern recognition neural networks that do not require a metric input space (Courrieu, 2005a), since dissimilarity measures are not distances.

Appendix A. Matlab code

Appendix A.1

Main function of a Matlab program that computes the density code ($(M\times n)$ matrix) from an input data ($m\times n$) matrix X. One can choose the kernel function (Gaussian, Logistic, or Cauchy density), and its scale parameter (default options are also provided).

```
function C = DensityCode(X, M, krnl, s, Tol)
% Input: X = (m x n) matrix of data, M = number of code points.
% Option: krnl = Gauss(1), Logistic(2), or Cauchy(3) kernel.
% Option: s = kernel scale parameter (> 0).
% Option: Tol = tolerance coefficient (0 < Tol < 1).
% Output: C = (M x n) density code matrix.
        % Initializations:
if nargin < 5, Tol = 1e-6; end  % default tolerance = 1e-6.
[NbrData, n] = size(X); mData = mean(X); sData = std(X);
TolC = Tol*sData;
C = Faure(M,n);  % Faure (1982) sequence of M points in (0,1)^n.
if nargin < 4    % default scale ≈ mean nearest neighbour distance.
    NProbe = 30; s = 0;
    for i = 1:NProbe
        p = fix(rand(1,1)*NbrData)+1; NN = inf;
        for j = 1:NbrData
            d = norm(X(p,:)-X(j,:));
            if (d ~= 0) && (d < NN), NN = d; end
        end
        s = s + NN;
    end
    s = s/NProbe;
end
if nargin < 3, krnl = 1; end  % default kernel = Gaussian.
        % Code computation by a bounding method:
for k = 1:M
    ProdKer = ones(NbrData,1); sumProdKer = NbrData; u = C(k,:);
    for ii = 1:n
        i = n-ii+1; ci = mData(i);
        p = sum(prob(ci,X(:,i),s,krnl).*ProdKer)/sumProdKer;
        if p < u(i)
```

```
            while p < u(i)
                cinf = ci; ci = ci + sData(i);
                p = sum(prob(ci,X(:,i),s,krnl).*ProdKer)/sumProdKer;
            end
            csup = ci;
        else
            while p >= u(i)
                csup = ci; ci = ci - sData(i);
                p = sum(prob(ci,X(:,i),s,krnl).*ProdKer)/sumProdKer;
            end
            cinf = ci;
        end
        while (csup-cinf) > TolC(i)
            ci = (csup+cinf)/2;
            p = sum(prob(ci,X(:,i),s,krnl).*ProdKer)/sumProdKer;
            if p < u(i), cinf = ci; else csup = ci; end
        end
        C(k,i) = ci;
        ProdKer = ProdKer.*kernel(ci,X(:,i),s,krnl);
        sumProdKer = sum(ProdKer);
    end

end  % end of the main function DensityCode.
```

Appendix A.2

Probability functions that are called by the main function DensityCode.

```
function f = kernel(x,m,s, krnl)
% Three kernel density functions
switch krnl
    case 1      % Gauss density
        f = exp(-(x-m).^2./(2*s.^2))./(s*sqrt(2*pi));
    case 2      % Logistic density
        vv = exp(-(x-m)./s);
        f = (vv./s)./(1+vv).^2;
    otherwise  % Cauchy density
        f = (1./(pi*s))./(1+((x-m)./s).^2);
end  % end of kernel.


function f = prob(x,m,s, krnl)
% Three cumulative probability functions
switch krnl
    case 1      % Gauss probability
        f = 0.5*erfc(-(x-m)./(s*sqrt(2)));
    case 2      % Logistic probability


        f = 1./(1+exp(-(x-m)./s));
    otherwise  % Cauchy probability
        f = atan((x-m)./s)./pi + 0.5;

end  % end of prob.
```

Appendix A.3

Functions for generating Faure sequences (Faure, 1982) of length M and dimension n. Points that have at least one zero coordinate are excluded.

```
function S = Faure(M,n)
% Generates a sequence of M points, quasi-uniform in (0,1)^n
q = n;
while ~isprime(q), q = q+1; end
S = zeros(M,n);
t1 = 1;
for t = 1:M
    S(t,:) = FaurePoint(n,t1,q);
    while min(S(t,:))==0  % exclude bound points
        t1 = t1+1;
        S(t,:) = FaurePoint(n,t1,q);
    end
    t1 = t1+1;
end  % end of Faure.


function p = FaurePoint(n,t,q)
% Generates the tth point of a Faure (1982) sequence
t = t-1;
kk = 1;
b(kk) = mod(t,q);
t = fix(t/q);
while t > 0
    kk = kk+1;
    b(kk) = mod(t,q);
    t = fix(t/q);
end
p = zeros(1,n);
qj = 1;
i = 1:n;
for j = 0:(kk-1)
    c = ones(1,n);
    cj = b(j+1)*c;
    if j < (kk-1)
        for jj = 1:(kk-1-j)
            c = (j+jj)*(i-1).*(c./jj);
            cj = cj+b(j+jj+1)*c;
        end
    end
    cj = mod(round(cj),q);
    qj = qj*q;
    p = p+cj./qj;

end  % end of FaurePoint.
```

Appendix A.4

The function DeltaPoly($c1$, $c2$, d) computes $\delta_{\Psi}(c1, c2)$, where $c1$ and $c2$ are density codes, and Ψ is the set of n-variable polynomials of degree lower or equal to d.

```
function delta = DeltaPoly(c1, c2, d)
% delta(c1,c2) with d-degree polynomial invariants
[m,n] = size(c1);
if d == 0    % direct comparison of density codes
    delta = sqrt(sum(sum((c1-c2).^2))/m);
else         % comparison of codes using invariants
    pw = AllPowers(n,d);
    [n,NbrTerms] = size(pw);
    x = ones(m,NbrTerms);
    for t = 1:NbrTerms
        for i = 1:n
            x(:,t) = x(:,t).*c1(:,i).^pw(i,t);
```

```
            end
        end
        T = pinv(x)*c2; % Moore-Penrose inverse method
        delta = sqrt(sum(sum((x*T - c2).^2))/m);
    end  % end of DeltaPoly.


    function pwrs = AllPowers(n,d)
    % All vectors of n positive integers of sum ≤ d
    global PW;
    PW = [];
    for k = 0:d
        kPowers(n,[],k);
    end
    pwrs = PW;  % end of AllPowers.


    function kPowers(n,v,k)
    % Recursively builds vectors of sum k
    global PW;
    if length(v) == (n-1)
        v = [v;k];
        PW = [PW,v];
    else
        for p = 0:k
            kPowers(n,[v;p],k-p);
        end

    end  % end of kPowers.
```

Appendix A.5

Main function of a Matlab program for the simulation of a parallel neural implementation of the density encoding algorithm usable on gray level images.

```
function layer3 = ImageDensityCode(layer1a,M,krnl,s,beta)
% Simulation of a parallel implementation of image density encoding

% The first input argument is the Layer 1 activation matrix
[row,col] = size(layer1a);
layer2P2 = zeros(row,1);
layer2P1 = zeros(row,col); % similar grids for layers 1 and 2
[X1,Y1] = meshgrid(1:col,1:row);
suma = sum(sum(layer1a));
for y2 = 1:row  % computation of P2[f] in Layer 2
    sumaG2 = sum(sum(layer1a.*prob(y2,Y1,s,krnl)));
    layer2P2(y2,1) = sumaG2/suma;
end
for y2 = 1:row  % computation of P1[f] in Layer 2
    sumag2 = sum(sum(layer1a.*kernel(y2,Y1,s,krnl)));
    for x2 = 1:col
        sumaG1g2 = ...
        sum(sum(layer1a.*prob(x2,X1,s,krnl).*kernel(y2,Y1,s,krnl)));
        layer2P1(y2,x2) = sumaG1g2/sumag2;
    end
end
layer3 = Faure(M,2); % computation of the density code in Layer 3
for i = 1:M
    ui = layer3(i,1); vi = layer3(i,2);
    W = (ui - layer2P1).^2;
    V = (vi - layer2P2).^2;
    for x2 = 1:col
        W(:,x2) = exp(-beta*sqrt(W(:,x2) + V));
    end
    sumexp = sum(sum(W));
    layer3(i,1) = sum(sum(X1.*W))/sumexp;
    layer3(i,2) = sum(sum(Y1.*W))/sumexp;
end  % end of ImageDensityCode.
```

References

Amari, S., & Nagaoka, H, (2000). *Methods of information geometry. AMS translations of mathematical monographs* (Vol. 191), Oxford: Oxford University Press.

Arbter, K., Snyder, W. E., Burkhardt, H., & Hirzinger, G. (1990). Applications of affine-invariant Fourier descriptors to recognition of 3-D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(7), 640–646.

Ben Israel, A., & Greville, T. N. E. (2003). *Generalized inverses: Theory and applications* (2nd ed.). New York: Springer.

Cacoullos, T. (1966). Estimation of a multivariate density. *Annals of the Institute of Statistical Mathematics (Tokyo)*, *18*(2), 179–189.

Calot, G. (1967). *Cours de Calcul des Probabilités* (2nd ed.). Paris: Dunod.

Courrieu, P. (2001). Two methods for encoding clusters. *Neural Networks*, *14*, 175–183.

Courrieu, P. (2002). Straight monotonic embedding of data sets in Euclidean spaces. *Neural Networks*, *15*, 1182–1193.

Courrieu, P. (2004). Solving time of least square systems in sigma–pi unit networks. *Neural Information Processing: Letters and Reviews*, *4(3)*, 39–45 (http://www.nip-lr.info).

Courrieu, P. (2005a). Function approximation on non-Euclidean spaces. *Neural Networks*, *18*, 91–102.

Courrieu, P. (2005b). Fast computation of Moore–Penrose inverse matrices. *Neural Information Processing: Letters and Reviews*, *8(2)*, 25–29 (http://www.nip-lr.info).

Faure, H. (1982). Discrépance de suites associées à un système de numération (en dimension s). *Acta Arithmetica*, *XLI*, 337–351.

Faure, H. (2001). Variations on (0, s)-sequences. *Journal of Complexity*, *17*, 741–753.

Halton, J. H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, *2*, 84–90.

Heikkilä, J. (2004). Pattern matching with affine moment descriptors. *Pattern Recognition*, *37*, 1825–1834.

Hu, M. K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, *8*, 179–187.

Jin, L., & Tianxu, Z. (2004). Fast algorithm for generation of moment invariants. *Pattern Recognition*, *37*, 1745–1756.

Kendall, D. G. (1984). Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, *16*, 81–121.

Mokhtarian, F., & Abbasi, S. (2002). Shape similarity retrieval under affine transforms. *Pattern Recognition*, *35*, 31–41.

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*, 281–294.

Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, *33*, 1065–1076.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*(11), 1019–1025.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks*, *3*, 109–118.

Suk, T., & Flusser, J. (2003). Combined blur and affine moment invariants and their use in pattern recognition. *Pattern Recognition*, *36*, 2895–2907.

Tzelgov, J., & Henik, A. (1983). On the recognition of words with inverted letters. *Canadian Journal of Psychology*, *37*, 233–242.

Zhang, D., & Lu, G. (2002). Shape-based image retrieval using generic Fourier descriptor. *Signal Processing: Image Communication*, *17*, 825–848.

Zhang, J., Zhang, X., Krim, H., & Walter, G. G. (2003). Object representation and recognition in shape spaces. *Pattern Recognition*, *36*, 1143–1154.

Neural Information Processing – Letters and Reviews Vol. 11, No. 12, December 2007

LETTER

Fast Density Codes for Image Data

Pierre Courrieu

Laboratoire de Psychologie Cognitive, UMR CNRS 6146, Université de Provence – Centre St Charles
Bat. 9, Case D, 3 place Victor Hugo, 13331 Marseille cedex 3, France
E-mail: Pierre.Courrieu@univ-provence.fr

(Submitted on October 22, 2007)

Abstract—Recently, a new method for encoding data sets in the form of "Density Codes" was proposed in the literature (Courrieu, 2006). This method allows to compare sets of points belonging to every multidimensional space, and to build shape spaces invariant to a wide variety of affine and non-affine transformations. However, this general method does not take advantage of the special properties of image data, resulting in a quite slow encoding process that makes this tool practically unusable for processing large image databases with conventional computers. This paper proposes a very simple variant of the density code method that directly works on the image function, which is thousands times faster than the original Parzen window based method, without loss of its useful properties.

Keywords—Image encoding, shape recognition, invariants, fast computation, neural processing simulation.

1. Introduction

Recently, a new method for encoding sets of points belonging to multidimensional spaces has been proposed by [1]. Given a set of data points, this method builds a deterministic sequence of points, called "density code", whose spatial distribution approximates that of the data. However, contrarily to data points, code points are strictly ordered in a non-arbitrary way, which makes any pair of density codes comparable. This allows building powerful code dissimilarity functions that can be simply made invariant to a wide variety of affine and non-affine natural transformations. As demonstrated in [1], this is clearly a promising approach to pattern recognition problems. However, it turned out that the encoding process is quite slow (several minutes for a 200×200 pixels image), mainly due to the use of a Parzen window scheme to smoothly approximate the data density [2], and to the requirement of integrating kernel functions. It was also proposed, in [1], a parallel neuron-like implementation of the method for image data, which could be fast when actually running on a parallel architecture. Unfortunately, the simulation of this parallel implementation on a conventional sequential computer is much slower than the basic implementation, that is itself too slow to be used on-line, or to process large image databases and to perform realistic simulations of neural processing models. Given that most simulations are performed on conventional computers, there is clearly a need to find a rapid way of building density codes for common data types such as images or image sequences, without loosing the useful properties of these codes. Fortunately, this is easy if one takes into account the "array of pixels" structure of image data, as we shall see.

2. Theory and Formulation

Background—First, we rapidly summarize the density code method foundations as stated in [1]. Let $f(X)$ be a probability density function on R^n, then the density of the $(n-k+1)$-dimensional marginal variable $(x_k, x_{k+1}, ..., x_n)$ is given by:

$$f(\bullet, ..., \bullet, x_k, x_{k+1}, ..., x_n) = \int_{R^{k-1}} f(x_1, ..., x_n)\, dx_1 ... dx_{k-1}.$$

The density of the one-dimensional conditional variable $(x_k \mid x_{k+1} = a_{k+1}, ..., x_n = a_n)$ is given by:

$$f(x_k \mid x_{k+1} = a_{k+1},...,x_n = a_n) = \frac{f(\bullet,...,\bullet,x_k,a_{k+1},...,a_n)}{f(\bullet,...,\bullet,\bullet,a_{k+1},...,a_n)}\ .$$

The cumulative probability function of this variable is given by:

$$\Pr(x_k \le b \mid x_{k+1} = a_{k+1},...,x_n = a_n) = \int_{-\infty}^{b} \frac{f(\bullet,...,\bullet,x_k,a_{k+1},...,a_n)}{f(\bullet,...,\bullet,\bullet,a_{k+1},...,a_n)}\, dx_k.$$

In compliance with [1], one simplifies the above notation by:

$$\Pr(x_k \le b \mid x_{k+1} = a_{k+1},...,x_n = a_n) = F(b \mid a_{k+1},...,a_n),$$

where the uppercase F recalls that de corresponding density function is f.

For a random variable $X \in R^n$ with density f, one defines a mapping $P[f]$ from R^n to $(0,1)^n$ as:

$$P[f](X) = (P_1[f](X), P_2[f](X),...,P_n[f](X)),$$

where

$$P_n[f](X) = F(x_n) = \int_{-\infty}^{x_n} f(\bullet,...,\bullet,t)\, dt,$$

and

$$P_k[f](X) = F(x_k \mid x_{k+1},...,x_n), \quad 1 \le k \le n-1.$$

Let U be a random variable uniformly distributed in $(0,1)^n$, and assume that $P[f]$ is a bijection, then, according to Theorem 1 from [1], the reciprocal bijection $P^{-1}[f]$ has the following property:

$P^{-1}[f](U)$ is distributed as X, with a probability density equal to f.

The above result is the foundation of the density coding method, since a density code is simply a realization of the mapping $P^{-1}[f](U)$ with a fixed sequence of m distinct values of U. Theorem 1 from [1] also states that a sufficient condition for $P[f]$ to be a bijection is that f be continuous and nowhere zero. This fact motivated the use in [1] of a superposition of continuous kernel functions centered on data points and asymptotically decreasing to zero, in order to approximate f. This solution works, however it is computationally slow.

Let $Y \in R^n$ be a random variable functionally related to X by a continuous invertible transformation ψ, then:

$$Y = \psi(X), \text{ and } g(Y) = f(X)J^{-1}(\psi(X)),$$

where g is the probability density of Y, and $J(\psi(X))$ is the Jacobian determinant of the transformation.

Theorem 2 from [1] states that if the Jacobian matrix $((\partial\psi_i/\partial x_j)(X))$ of the transformation is everywhere triangular ($j < i \Rightarrow \partial\psi_i/\partial x_j = 0$), and has a strictly positive diagonal ($\partial\psi_i/\partial x_i > 0, \quad 1 \le i \le n$), then:

$$P^{-1}[g](U) = \psi(P^{-1}[f](U)).$$

The above result is the foundation of the density code comparison method. The set of transformations that have the required properties includes a wide variety of affine and non-affine natural transformations. However, certain common affine transformations such as rotations and reflections are excluded.

Encoding Algorithm—Now, how can we translate the above model for finite array data types such as images or sequences of images? First, we assume that the (unknown) original image function h is a positive function whose continuous support is a hyper-rectangle $R_h = [0, S_1] \times ... \times [0, S_n]$, where the S_i's are expressed in pixel side units, and $h = 0$ outside this hyper-rectangle. The image discretization results in a (given) multidimensional array $\tilde{h}$ where each (hyper-) pixel, with integer coordinates $(x_1,...,x_n)$, $1 \le x_i \le S_i$, has the mean value of h on the unit volume hypercube $[x_1 - 1, x_1] \times ... \times [x_n - 1, x_n]$, that is:

$$\tilde{h}(x_1,...,x_n) = \int_{x_1-1}^{x_1} ... \int_{x_n-1}^{x_n} h(t_1,...,t_n)\, dt_1...dt_n.$$

This simple and quite reasonable approximation of the discretization process allows us to reduce all subsequent integrals to finite discrete sums of pixel values. However, image data are subject to variations of foreground and background lighting that are irrelevant for shape recognition. An image affine transformation

allows us to partly solve this problem. Let $\min(\tilde{h})$ and $\max(\tilde{h})$ denote respectively the minimum and maximum pixel values in $\tilde{h}$, then one transforms the array $\tilde{h}$ into an array g by:

$g = (\tilde{h} - \min(\tilde{h}))/(\max(\tilde{h}) - \min(\tilde{h}))$, for a light figure on a dark background,

$g = (\max(\tilde{h}) - \tilde{h})/(\max(\tilde{h}) - \min(\tilde{h}))$, for a dark figure on a light background.

Let $\Sigma(A)$ denote a real number that is the sum of all cell values of an array A. Then one can choose the code length about $m \approx \alpha \Sigma(g)$, for a fixed $\alpha > 0$, in order to make the number of code points proportional to the image "foreground mass". One can note that the array $g/\Sigma(g)$ has the properties of a discrete probability function, however, certain cells have a zero value, with the consequence that certain cumulative probability functions are not strictly increasing, and thus they cannot be inversed. A simple solution to this problem consists of adding to each cell of the array g a small positive quantity c such as:

$$c = \lambda \, \Sigma(g) / \prod_{i=1}^{n} S_i ,$$

where λ is a small positive constant (e.g. $\lambda = 0.0001$). This limited "lighting of the background" has the same role as the strictly positive kernel functions used in the original method [1]. Finally, the discrete probability function from which we are going to build the density code is given by the array f defined by:

$$f = (g + c)/\Sigma(g + c).$$

The remaining difficulty results from the fact that f contains only a finite set of values, and thus the same is true for any cumulative function computed from f. As a consequence, one can find an infinite number of values of $U \in (0,1)^n$ for which there is no corresponding cell in f. This problem can be solved using a discrete dichotomic bounding search completed with a local linear interpolation. Given a value of $U \in (0,1)^n$, one can compute its corresponding code point $X = P^{-1}[f](U) \in R_h$ as follows:

function $P^{-1}[f](u_1,...,u_n)$ returns $(x_1,...,x_n)$
$f_n \leftarrow f$
for $k \leftarrow n$ downto 1 do
  $P_k[f](0) \leftarrow 0$ % computation of the vector $P_k[f](0:S_k)$
  for $i \leftarrow 1$ to S_k do $P_k[f](i) \leftarrow P_k[f](i-1) + \sum_{i_1,...,i_{k-1}} f_k(i_1,...,i_{k-1},i)$ endfor$i$
  $P_k[f](1:S_k) \leftarrow P_k[f](1:S_k)/P_k[f](S_k)$
  $inf \leftarrow 0$, $sup \leftarrow S_k$ % dichotomic search
  while $(sup - inf) > 1$ do
    $mid \leftarrow (inf + sup)$ div 2
    if $u_k \geq P_k[f](mid)$ then $inf \leftarrow mid$ else $sup \leftarrow mid$ endif
  endwhile
  $w \leftarrow (u_k - P_k[f](inf))/(P_k[f](sup) - P_k[f](inf))$ % linear interpolation
  $x_k \leftarrow inf + w\ sup$
  if $k > 1$ then
    if $inf > 0$ then $f_{k-1} \leftarrow f_k(1:S_1,...,1:S_{k-1},inf) + w\ f_k(1:S_1,...,1:S_{k-1},sup)$
      else $f_{k-1} \leftarrow w\ f_k(1:S_1,...,1:S_{k-1},sup)$ endif
  endif
endfor$k$

In the above pseudo-code, the notation "$a:b$" is that of Matlab, and it refers to the index range $[a,a+1,...,b-1,b]$. Any text at right of "%" is a comment. Note that, for computational effectiveness, it is preferable to implement a specific version for each dimension n, as illustrated by the Matlab function "ImageCode", listed in the Appendix, for $n = 2$. One must also take care that the order of variables determines the set of coordinate transformations to which the comparison of codes can be made invariant. Typically, for image data, the order (x,y) of the geometrical plane coordinates corresponds to a more probable variety of

natural transformations than the reverse order (y,x). However, the first dimension of an array usually corresponds to the y coordinate, and the second dimension corresponds to the x coordinate. So, the implementation must consider the dimensions in the most appropriate order, which is not necessarily the order of the array dimensions.

It remains to choose a sequence of points $(U_1,...,U_m)$ uniformly distributed in $(0,1)^n$, and to compute the code point corresponding to each of these points by the mapping $P^{-1}[f](U_j), 1 \le j \le m$, in the same order, which provides the desired density code. The length m of the sequence can vary, if necessary, depending on the image size and complexity, however, for every given index j, the point U_j must always be the same in order to make different density codes comparable. A good choice is to use a quasi-uniform sequence such as a Halton sequence [3] or a Faure sequence [4,5], as in [1]. It is well known that Faure sequences must be preferred for high dimension spaces, however, the data arrays considered in this paper have rarely more than three dimensions, thus one can as well use simple Halton sequences, and the Matlab function named "Halton" in the Appendix generates such sequences.

Dissimilarity Function—Given two density codes V and W, that are $m \times n$ real matrices, both computed using the same quasi-uniform sequence, and given a chosen family Ψ of transformation mappings, one can attempt to find a transformation $\tau \in \Psi$ that minimizes the quadratic matching error:

$$E^2_\Psi(V,W) = \min_{\tau \in \Psi} \left\| \tau(V) - W \right\|^2.$$

This minimization problem is very easy to solve if the transformation family Ψ is linear in its parameters, which is the case, for example, of the family of multivariate polynomials of a given degree d, that has the special advantage of naturally including the family of affine transformations (first degree polynomials). In this case, one computes the polynomial basis functions for each point in V, resulting in a real matrix B_V of order $m \times q$, where the number of monomials q depends on n and d. Then the $q \times n$ real matrix T of the optimal polynomial coefficients is simply given by $T = B_V^\dagger W$, where $B_V^\dagger$ is the pseudo-inverse of B_V [6,7]. In [1], it was proposed to use a dissimilarity function defined by:

$$\delta_\Psi(V,W) = \left\| B_V T - W \right\| / \sqrt{m}\,.$$

This dissimilarity function works well for data that conform to the basic probabilistic model, however, image functions do not behave exactly as probability functions, due to the presence of lighting variations, shadows, non-uniform background, and other sources of noise. As a result, when one compares two similar shapes, there is frequently a small proportion of code points that do not match and that provide very large errors. These points are outliers, and it is desirable to limit their effect on the dissimilarity measure. A simple solution to this problem is to replace the square root of the mean quadratic matching error, which is sensitive to outliers, by the median matching error, which is much less sensitive to outliers. Another difficulty, pointed out by [1], is that the dissimilarity measure is asymmetric and has the scale of the target code (W). This makes dissimilarity measures hard to compare when one works on a set of images that have different sizes. The solution is to make the dissimilarity measure relative to some evaluation of the scale of the target code. For example, one can divide the median matching error by the median distance of the target code points to their center of gravity (and multiply the result by 100 in order to obtain more readable numbers). Finally, in [1], only density codes having the same length were considered comparable, however, if two codes have different length, say m_1 and m_2, then the first $m = \min(m_1, m_2)$ code points are in fact comparable since they have been computed using the same quasi-uniform sub-sequence. Thus, we can compare density codes of different length, and we are no longer constrained to use a unique code length for all items in a database. The minor, yet useful, improvements of δ_Ψ suggested above are implemented in the Matlab function "DeltaMedian" listed in the Appendix.

3. Results

Codes—In order to test the performance of the above described method, we generated 6 pairs of images, each pair including a "plant-like" blurred fractal (A), and a "wind-like" transformation of it (B). Figure 1 shows the 12 test images, together with their first 1025 density code points generated using function calls of the form "*CodeName* = ImageCode (*DataArray*, U, 0)", with "U = Halton (1025, 2)" (see Appendix).

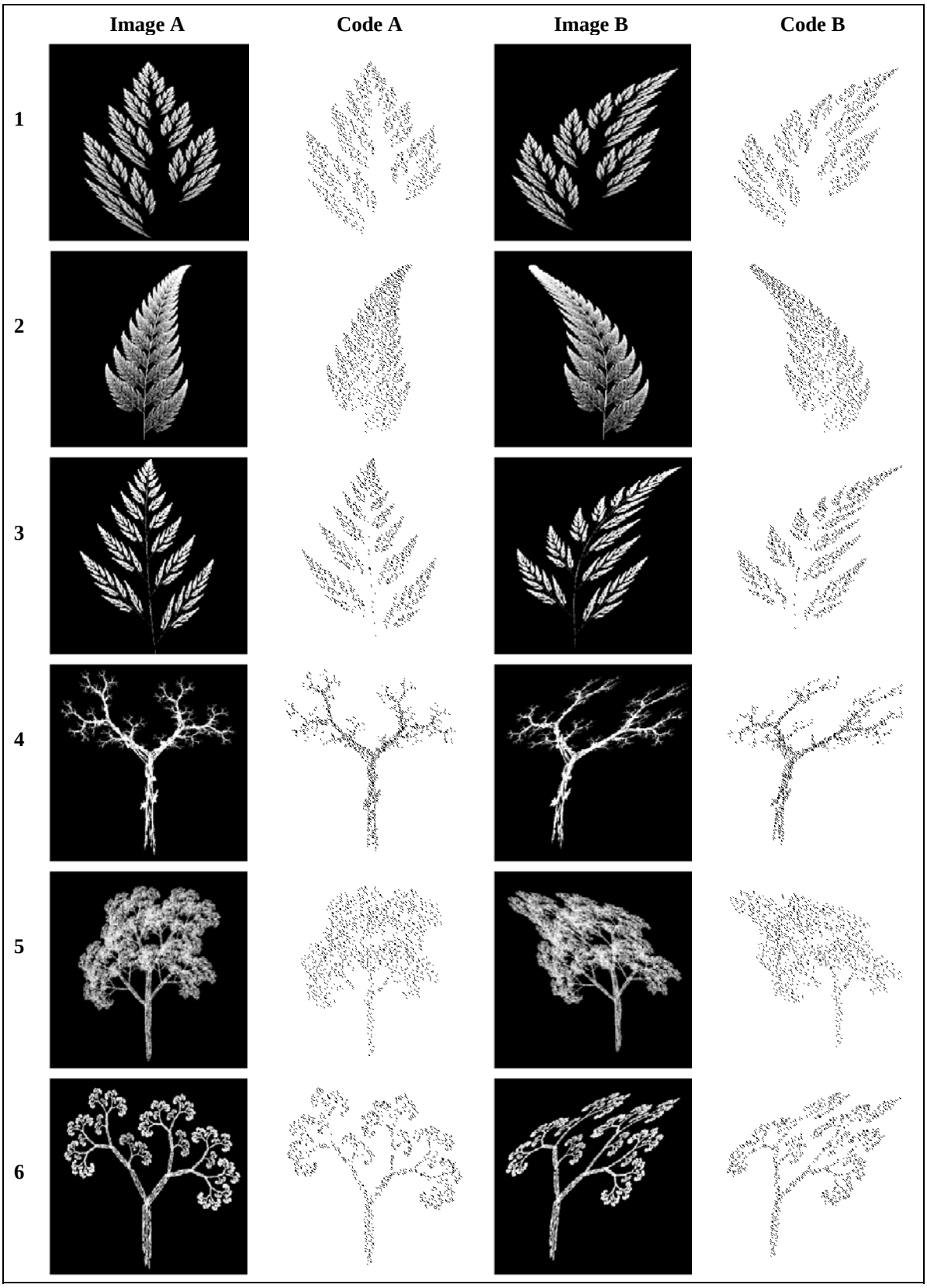


Figure 1. The 12 test images (256×256 pixels each) and their first 1025 density code points.

As one can see in Figure 1, the spatial distributions of code points suitably approximate the corresponding image foregrounds. Each code (1025 points for an image size of 256×256 pixels) was computed in about 19 milliseconds, in Matlab 7.4 running on a MacBook computer (Mac OS X, version 10.4.10), with a 2 GHz Intel Core 2 Duo processor.

Dissimilarity Measure—The "wind-like" transformations between A and B images of Figure 1 can be approximated by bivariate third-degree polynomial transformations, whose set was chosen as Ψ. The number of code points was made proportional (coefficient α) to the image foreground mass, using function calls of the form "*CodeName* = ImageCode (*DataArray*, U, 0, α)" (see Appendix), while α was experimentally varied from 0.01 to 0.5 (step 0.01). The foreground masses of test images ranged 3729-8923, and a long enough Halton sequence was available in all cases. Dissimilarity measures (δ_Ψ) were computed for all pairs of distinct images, for the two possible argument orders (since δ_Ψ is asymmetric), and for all α values. The function calls were of the form "*Delta* = DeltaMedian (*Code1*, *Code2*, 3)" (see Appendix). For each α value, one selected the minimum and maximum obtained δ_Ψ values, for pairs of unrelated shapes (distinct "plants"), and for pairs of related shapes (A-B "wind-like" transforms). The result is plotted in Figure 2.

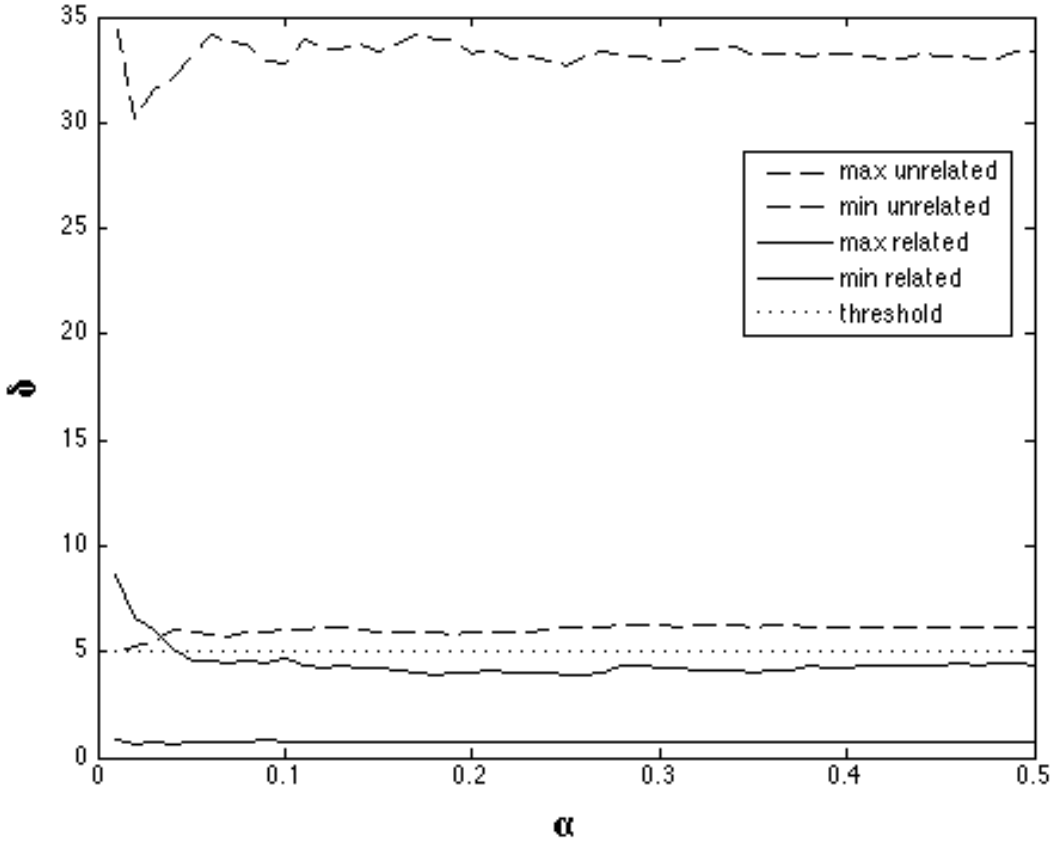


Figure 2. Observed δ_Ψ boundaries for related and unrelated image pairs, as functions of α.

As one can see in Figure 2, for $\alpha > 0.04$, the related and unrelated δ_Ψ distributions do not overlap, the δ_Ψ values become quite stable, and one can reliably decide whether or not two images are related using a simple threshold of about 5. One can also note that the maximum separation of δ_Ψ distributions is reached for α = 0.25, however, this *a priori* depends on the considered shape space, and in particular on the relative scale of relevant distinctive features.

Code Computation Time—An examination of the operations involved in the image coding process shows that the complexity depends on both the image height (H), the image width (W), and the number of code

points (m). The heaviest operations concern the image preprocessing ($O(HW)$), the m-times iterated dichotomic searches ($O(m(\log_2 H + \log_2 W - 2))$), and the m-times iterated computation of interpolated row vectors ($O(mW)$). In order to estimate the weights of these operations (for our platform), we measured the encoding time of random images whose height and width where independently varied from 16 to 1024 pixels (in powers of 2), while the code length was also independently varied from 16 to 1024, with repeated measures using 20 independent random images per condition. Then the regression equation was solved using a least square method, and we obtained the following approximation of the code computation time (in milliseconds):

$$t(H,W,m) \approx 10^{-4}(0.6853\,HW + 3.8459\,m(\log_2(HW) - 2) + 0.3943\,mW), \quad (\pm 2.66).$$

The standard approximation error is small enough for practical use, and the correlation between the observed and approximated computation times is $r = 0.99$. As an example, the approximated computation time for the images of Figure 1, with 1025 code points, is 20.4 ± 2.66 milliseconds, whereas the observed computation time was about 19 milliseconds. We observed that adding more terms to the regression equation does not significantly improve the approximation accuracy, whereas the formula is obscured by the presence of negative coefficients.

Finally, we note that the obtained computation times make the density code approach perfectly usable for on-line computation and for the processing of large image databases. Using the original encoding algorithm, the code computation time, as reported in [1], was of 6 minutes and 45 seconds for 2048 code points on an image of 200×200 pixels. Even though the used computers are not the same, there is no doubt that the present algorithm considerably improves the situation, performing a similar encoding in only 29.4 milliseconds.

Appendix

The following implementation code, in Matlab 7.4, is provided for example, and for academic use only. The code is not optimized and exception cases are not managed.

```
function code = ImageCode(f,u,DarkOnLight,alpha)
% Density code of an image f for a quasi-uniform sequence u
% For a fixed length code, do not provide the alpha argument
% Set DarkOnLight=1 for a dark figure on a light background (else 0)
[ymax,xmax]=size(f); minf=min(min(f)); maxf=max(max(f));
if DarkOnLight>0, f=(maxf-f)/(maxf-minf); else f=(f-minf)/(maxf-minf); end
Sf=sum(sum(f));
if nargin<4, m=length(u); else m=min(length(u),round(alpha*Sf)); end
lambda=0.0001; f=f+lambda*Sf/(ymax*xmax); Pyf=sum(f,2);
Pyf=cumsum(Pyf); Pyf=Pyf/Pyf(ymax);          % Pn[f] is computed only once
for p=1:m
    v=u(p,2); lob=1; upb=ymax;
    if v<=Pyf(1), w=v/Pyf(1); Pxf=f(1,:)*w;
    else while (upb-lob)>1
            y=round((lob+upb)/2);
            if Pyf(y)>v, upb=y; else lob=y; end
         end
        w=(v-Pyf(lob))/(Pyf(upb)-Pyf(lob));
        u(p,2)=lob+(upb-lob)*w;
        Pxf=f(lob,:)+(f(upb,:)-f(lob,:))*w;
    end
    Pxf=cumsum(Pxf); Pxf=Pxf/Pxf(xmax);   % Pk[f],k<n, is computed m times
    v=u(p,1); lob=1; upb=xmax;
    if v>Pxf(1)
       while (upb-lob)>1
           x=round((lob+upb)/2);
           if Pxf(x)>v, upb=x; else lob=x; end
       end
       u(p,1)=lob+(upb-lob)*(v-Pxf(lob))/(Pxf(upb)-Pxf(lob));
    end
end
code=u(1:m,:);

function u = Halton(m,n)
% Halton quasi-uniform sequence of m points in (0,1)^n
p=zeros(n,1);
```

```
p(1,1)=2;
for k=2:n
    p(k,1)=p(k-1,1)+1;
    while ~isprime(p(k,1)), p(k,1)=p(k,1)+1; end
end
u=zeros(m,n);
for t=1:m
    u(t,:)=point(n,t,p);
end

function pt=point(n,t,p)
pt=zeros(1,n);
for k=1:n
    pk=p(k,1); i=t; h=0; ib=1/pk;
    while (i>0)
        d=mod(i,pk);
        h=h+d*ib;
        i=round((i-d)/pk);
        ib=ib/pk;
    end
    pt(1,k)=h;
end

function delta = DeltaMedian(c1, c2, d)
% Delta function with d-degree polynomial invariants
% Modified from the DeltaPoly function listed in Courrieu (2006)
[m1,n]=size(c1); [m2,n]=size(c2); m=min(m1,m2);
c1=c1(1:m,:); c2=c2(1:m,:);     % reduce to comparable sub-sequences
if d == 0                       % direct comparison of density codes
    err = sqrt(sum((c1-c2).^2,2));
else                            % comparison of codes using invariants
    pw = AllPowers(n,d);
    [n,NbrTerms] = size(pw);
    x = ones(m,NbrTerms);
    for t = 1:NbrTerms
        for i = 1:n
            x(:,t) = x(:,t).*c1(:,i).^pw(i,t);
        end
    end
    T = pinv(x)*c2;             % optimal transformation coefficients
    err = sqrt(sum((x*T - c2).^2,2));
end
delta=median(err);              % median mismatch based dissimilarity
TargetCentre=mean(c2);
TargetScale= median(sqrt(sum((c2-kron(ones(m,1),TargetCentre)).^2,2)));
delta=100*delta/TargetScale;

function pwrs = AllPowers(n,d)
% All vectors of n positive integers of sum <= d
global PW;
PW = [];
for k = 0:d
    kPowers(n,[],k);
end
pwrs = PW;

function kPowers(n,v,k)
% Recursively builds vectors of sum k
global PW;
if length(v) == (n-1)
    v = [v;k];
    PW = [PW,v];
else
    for p = 0:k
        kPowers(n,[v;p],k-p);
```

```
    end
end
```

References

[1] P. Courrieu, "Density Codes and Shape Spaces", *Neural Networks, Vol. 19*, pp. 429-445, 2006.
[2] D.F. Specht, "Probabilistic Neural Networks", *Neural Networks, Vol. 3*, pp. 109-118, 1990.
[3] J.H. Halton, "On the Efficiency of Certain Quasi-random Sequences of Points in Evaluating Multi-dimensional Integrals", *Numerische Mathematik, Vol. 2*, pp. 84-90, 1960.
[4] H. Faure, "Discrépance de Suites Associées à un Système de Numération (en Dimension s)", *Acta Arithmetica, XLI*, pp. 337-351, 1982.
[5] H. Faure, "Variations on (0, s)-sequences", *Journal of Complexity, Vol. 17*, pp. 741-753, 2001.
[6] A. Ben Israel, & T.N.E. Greville, *Generalized Inverse: Theory and Applications* (2nd ed.), New York, Springer, 2003.
[7] P. Courrieu, "Fast Computation of Moore-Penrose Inverse Matrices", *Neural Information Processing-Letters and Reviews, Vol. 8, No. 2*, pp. 25-29, 2005.

Pierre Courrieu received his PhD degree from University of Provence in 1983, and he is a CNRS researcher currently working with psychologists and neuroscientists in Marseille (France). He is a member of the European Neural Network Society, and his research interests include visual shape recognition, neural computation, data encoding, function approximation, supervised learning, and global optimization methods.

II.D Réseaux de neurones et apprentissage supervisé

Lorsqu'on a construit un espace d'entrée et un espace de sortie, il reste à construire la machinerie établissant un lien fonctionnel des entrées vers les sorties. La Théorie de l'Approximation des Fonctions fournit depuis longtemps des bases solides pour construire des approximations aussi précises que l'on voudra de toute fonction continue (ou même seulement continue par morceaux) d'un espace métrique vers un (autre) espace métrique. Ces bases théoriques indispensables ne fournissent cependant pas directement les méthodes pratiques nécessaires pour résoudre les problèmes concrets d'approximation de fonctions que l'on rencontre dans les applications et dans la modélisation cognitive. C'est un des grands mérites de la théorie des réseaux de neurones artificiels que d'avoir développé des méthodes d'apprentissage qui, associées aux théorèmes fondamentaux d'approximation des fonctions, ont donné des algorithmes très efficaces pour résoudre des problèmes "naturels". Pour ma part, je me suis essentiellement intéressé aux réseaux de neurones dits "feedforward", c'est-à-dire aux réseaux où l'activation chemine des entrées vers les sorties sans rétroactions. Le cerveau humain étant en réalité un système dynamique non-linéaire complexe (Pezard & Nandrino, 2001), il est clair que les modèles feedforward sont des abstractions qui permettent, au mieux, d'approcher certaines fonctions cognitives, mais sans doute pas les fonctions cérébrales sous-jacentes. Ce que l'on appelle "apprentissage supervisé" consiste à calculer les paramètres, et éventuellement l'architecture, d'une machinerie neuronale implémentant une fonction complète (en général continue) à partir d'un échantillon fini de points de la fonction, appelés "exemples", dont on connaît les valeurs d'entrée et de sortie. Les points qui ne sont pas des exemples sont des points de "généralisation" où, étant donnée une valeur d'entrée, la machine neuronale doit estimer la valeur de sortie correspondante (par interpolation ou extrapolation), après un éventuel "lissage" visant à éliminer le bruit que peuvent contenir les exemples. Certains algorithmes d'apprentissage utilisent les exemples pour calculer les paramètres d'un réseau de neurones dont l'architecture est prédéterminée par le modélisateur. C'est le cas du très célèbre algorithme de rétropropagation du gradient d'erreur (Rumelhart, Hinton, & Williams, 1986), et des algorithmes plus simples, de type "moindres carrés", que l'on utilise avec les réseaux à fonctions bases radiales (Poggio & Girosi, 1990; Yoon, 2001). D'autres algorithmes d'apprentissage, comme la "Cascade-Correlation" (Fahlman, & Lebiere, 1990), déterminent à la fois l'architecture et les paramètres du réseau rendant compte des exemples. Il existe également des variantes incrémentales des méthodes de moindres carrés qui permettent de construire progressivement un réseau à fonctions bases radiales ou

similaires (Sin & DeFigueiredo, 1993). J'ai proposé, indépendamment du travail de Fahlman et Lebiere (1990) que je ne connaissais pas à l'époque, un algorithme d'apprentissage qui s'est avéré être une variante de l'algorithme de Cascade-Correlation (Courrieu, 1993a). Le travail de Fahlman et Lebiere n'était pas encore très connu, et personne (pas même les referees de "Neural Networks" où j'ai publié mon travail) n'a relevé la similitude des algorithmes à ce moment là. D'autres variantes de l'algorithme de Cascade-Correlation ont été publiées depuis, et c'est une famille d'algorithmes très utilisée dans les applications pratiques.

Cependant, une difficulté est apparue avec les réseaux de neurones et autres méthodes classiques d'approximation des fonctions. Ces méthodes sont conçues pour approcher des fonctions sur des espaces métriques euclidiens. Or il est de nombreux problèmes, notamment en reconnaissance des formes, où l'espace d'entrée n'est pas euclidien. C'est par exemple souvent le cas lorsque, désirant approcher une fonction sur un espace de chaînes de caractères, on structure cet espace par une pseudo-distance entre chaînes, obtenue par une méthode de Programmation Dynamique. C'est également le cas lorsque, désirant approcher une fonction sur un espace de formes en réduisant les transformations régulières, on structure l'espace par une mesure de "dissimilarité" non métrique entre formes (Courrieu, 2006, 2007, voir Section II.C). Il existe à ce jour deux façons de résoudre ce problème. La première fait appel à une extension simple de l'approximation "au plus proche voisin" usuelle, mais cette approche est très peu régulière et donne des généralisations assez grossières. J'ai proposé une autre approche, beaucoup plus "régularisée", qui donne de meilleurs résultats (Courrieu, 2005a, article ci-joint). Le modèle neurocomputationnel développé garantit la possibilité d'approcher aussi précisément que l'on voudra toute fonction continue, non seulement sur tout espace métrique (euclidien ou non), mais également sur une très large variété d'espaces topologiques non métriques, couvrant sans doute l'ensemble des besoins pratiques courants. Ainsi qu'il est dit dans l'article, ce résultat représente, me semble-t-il, une extension significative de la notion usuelle de "capacité d'approximation universelle". Il me faut cependant mettre un bémol à cet autosatisfecit, car la publication de ce travail semble n'avoir eu qu'un succès très modeste. Parmi les rares auteurs qui ont cité l'article, certains n'ont même pas relevé que l'outil proposé permet d'approcher des fonctions sur des espaces non métriques, de sorte que je me demande pourquoi ils ont cité ce travail. La faute est sans doute à ma maladresse chronique en matière de communication. Je profite donc de la présente circonstance académique pour signaler que, si l'on a à approcher une fonction sur un espace aux propriétés topologiques extravagantes, il peut être de quelque utilité de consulter la référence Courrieu (2005a), ainsi d'ailleurs que la référence Courrieu (2002) présentée dans la section II.B.

References

Courrieu, P. (1993a). A convergent generator of neural networks. *Neural Networks, 6*, 835-844.

Courrieu, P. (2005a). Function approximation on non-Euclidean spaces. *Neural Networks, 18*, 91-102.

Fahlman, S.E., & Lebiere, C. (1990). The Cascade-Correlation learning algorithm. *In* D.S. Touretsky (Ed.): *Advances in Neural Information Processing Systems, 2*. San Mateo, CA: Morgan Kauffman Publishers, pp. 525-532.

Pezard, L., & Nandrino, J.-L. (2001). Paradigme dynamique en psychopathologie: la «Théorie du chaos», de la physique à la psychiatrie. *L'Encéphale, XXVII,* 260-8.

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE, 78(9)*, 1481-1497.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. *In* D.E. Rumelhart & J.L. McClelland (Eds.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press, pp. 318-362.

Sin, S.-K., & DeFigueiredo, R.J.P. (1993). Efficient learning procedures for optimal interpolative nets. *Neural Networks, 6*, 99-113.

Yoon, J. (2001). Interpolation by Radial Basis Functions on Sobolev space. *Journal of Approximation Theory, 112*, 1-15.

Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 18 (2005) 91–102
Neural Networks
www.elsevier.com/locate/neunet

Function approximation on non-Euclidean spaces

Pierre Courrieu*

Laboratoire de Psychologie Cognitive, CNRS-UMR 6146, Université de Provence, 29 avenue Robert Schuman, 13621 Aix-en-Provence cedex 1, France

Received 30 April 2003; accepted 24 September 2004

Abstract

This paper presents a family of layered feed-forward networks that is able to uniformly approximate functions on any metric space, and also on a wide variety of non-metric spaces. Non-Euclidean input spaces are frequently encountered in practice, while usual approximation schemes are guaranteed to work only on Euclidean metric spaces. Theoretical foundations are provided, as well as practical algorithms and illustrative examples. This tool potentially constitutes a significant extension of the common notion of 'universal approximation capability'.

Keywords: Function approximation; Non-metric spaces; Feed-forward neural networks; Regularization; Invariants

1. Introduction

Neural computation research, together with related areas in approximation theory, have developed powerful methods for approximating continuous mappings on compact subsets of R^n as a Euclidean space. Most approximation schemes use three layered feed-forward neural architectures with scalar product based neurons (Cybenko, 1989; Funahashi, 1989; Hornik, 1993; Leshno, Lin, Pinkus, & Schocken, 1993; Rumelhart, Hinton, & Williams, 1986a,b), or Euclidean distance based neurons (Girosi & Poggio, 1990; Micchelli, 1986; Poggio & Girosi, 1990; Yoon, 2001), while more general feed-forward architectures have also sometimes been studied (Courrieu, 1993; Fahlman & Lebiere, 1990; Kreinovich, 1991). In such schemes, function approximation capabilities critically depend on the Euclidean metric nature of the input space. However, it is frequent in practical applications that one must approximate functions on data spaces that are not Euclidean, not metric, or even not numerical (symbol strings, graphs). In such cases, one usually attempts to empirically encode the input data in the form of numerical vectors, with the hope, but without any guarantee, that the performed encoding is relevant. Then one applies conventional neural methods on the space of codes, assuming that it is Euclidean and that it suitably preserves the topology of the original data space. Such an empirical encoding can result in a failure in approximating the objective function because the complexity of the approximation to be found not only depends on that of the objective function, but also on the suitability of the encoding. The data-encoding problem has been previously treated for special data types such as clusters, that have been shown to be suitably encodable in Euclidean spaces (Courrieu, 2001). It has also been shown that certain data sets can be monotonically embedded in a Euclidean space, provided that their topology is induced by a function that has at least some properties of a metric, except possibly the triangle inequality (Courrieu, 2002). In these cases, one can apply a conventional neural method for function approximation on the Euclidean embedding space. However, there are also data spaces that cannot be embedded in a Euclidean space without strongly modifying their topology. This is the case, for example, of data spaces where neighborhood relations are not symmetric, or whose topology is induced by a semi-metric (that can take a zero value for pairs of distinct elements). Such topological data spaces are not rare in practice, and it can even happen that their basically non-metric properties are in fact relevant to the function approximation problem to be solved. Consider, for example, objective functions invariant to certain regular transformations of the input, which are common in pattern recognition (see Section 8 of this paper for a meaningful

* Tel.: +33 4 42 95 37 28; fax: +33 4 42 20 59 05.
E-mail address: courrieu@up.univ-mrs.fr.

0893-6080/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2004.09.003

example). Non-Euclidean and non-metric spaces are also commonly generated by Dynamic Programming methods applied to time series or symbol strings (see Section 9). Thus, there is clearly a need for building function approximation schemes on data spaces with minimal requirements concerning their topology, which is the purpose of this paper. The model proposed hereafter has the form of a layered feed-forward neural network, with special neural basis functions, and it can approximate functions on a wide variety of non-metric data spaces, as well as on any metric space. Data spaces are usually probabilized by a (possibly unknown) sampling probability function on the input space. This property will be explicitly used hereafter.

2. Problem statement

2.1. Input space topology

Let Ω be any set, and δ be a real valued function on $\Omega \times \Omega$ such that, for any $X,Y \in \Omega$:

$$\delta(X,X) = 0,$$

$$\delta(X,Y) \geq 0,$$

$$\delta(X,Y) < \infty.$$

Then (Ω,δ) is a topological space whose topology is induced by δ. In particular, a closed ball of center $X \in \Omega$, and of radius $r \geq 0$, is the set defined by:

$$B(X,r) = \{Y \in \Omega,\ \delta(X,Y) \leq r\}.$$

2.2. Sampling probability

Let μ be a sampling probability on Ω such that:

$$\mu(\Omega) = 1, \quad \text{for any } X \in \Omega,\ r > 0 \Rightarrow \mu(B(X,r)) > 0.$$

Note that this last requirement implies some restriction concerning δ. In particular, if Ω is a continuum and μ is a continuous probability function, then δ cannot be a trivial distance because any ball of radius less than 1 would have only one element (its center), and then would be a subset of zero measure of Ω.

2.3. Objective function

Let f be a real valued mapping from Ω to R^K such that, for any $X,Y \in \Omega$:

$$|f_i(X) - f_i(Y)| \leq a_i \delta(X,Y), \quad 0 < a_i < \infty, \quad 1 \leq i \leq K.$$

If (Ω,δ) is a metric space, then the above requirement is simply a Lipschitz condition on f. The above property, together with the finiteness of δ (see Section 2.1), implies that f is bounded on Ω.

2.4. Approximation problem

Find an approximation of f, given a learning set of M data points

$$\Xi = \{(X_i, f(X_i)),\ 1 \leq i \leq M\},$$

where points are sampled on Ω with the probability μ.

One requires that the approximation be uniformly convergent as M tends to infinity.

3. Approximation method

3.1. Prototypical examples

First, one must select a subset of m data points from Ξ, with $m \leq M$, in such a way that

$$\min_{1 \leq i \neq j \leq m} \delta(X_i, X_j) = s > 0.$$

Each of these points is a 'prototype' that will be associated to a particular basis function. The strictly positive real number s is the minimum 'spacing' of prototypes in (Ω,δ), while non-prototypical examples are not subject to any spacing constraint. The set of input values of prototypes is denoted χ.

3.2. Low level layer(s)

Given an input $X \in \Omega$, one uses one or more low level layer(s) for computing δ values between the current input X and the m prototypical input points X_i, $1 \leq i \leq m$. The output of this processing is a set of m real values $\delta(X,X_i)$, $1 \leq i \leq m$. Note that the order of the arguments of δ is relevant since we do not assume that δ is symmetric. The exact specification of this low level processing of course depends on each particular input space (Ω,δ).

3.3. Basis functions

The output of the low level processing (Section 3.2) is used as the input of a layer of m neural basis functions, that are defined as follows:

$$g_i(X;\beta) = \frac{e^{-\beta \cdot \delta(X,X_i)}}{\sum_{j=1}^{m} e^{-\beta \cdot \delta(X,X_j)}}, \quad 1 \leq i \leq m.$$

The determination of the real parameter $\beta > 0$ will be studied in the following sections. We note that the basis functions are all positive and that their sum is equal to 1, for any $X \in \Omega$.

3.4. Output layer

The output of the basis function layer (Section 3.3) is used as the input of an output layer of K (linear) neurons.

The synaptic weight of the connection from the ith basis function neuron to the jth output neuron is denoted w_{ij}, with $1 \leq i \leq m$, and $1 \leq j \leq K$. The output functions are given by:

$$\varphi_j(X) = \sum_{i=1}^{m} w_{ij} g_i(X;\beta), \quad 1 \leq j \leq K.$$

In the following, we will refer to such approximators as 'φ approximators'.

3.5. Computation of synaptic weights

Synaptic weights are simply computed by a usual Least Square method, which provides an exact interpolation of data points if $m=M$, or which allows for filtering possible data noise if $m<M$.

Let G be the $M \times m$ real matrix of basis function values for the M learning examples:

$$G = (g_i(X_p;\beta)), \quad 1 \leq p \leq M, \quad 1 \leq i \leq m.$$

Let F be the $M \times K$ real matrix of the objective function values for the learning examples.

Let W be the $m \times K$ real matrix of synaptic weights to be computed.

The Least Square solution can be obtained, for example, by using the pseudo-inverse method

$$W = (G'G)^{-1}G'F,$$

where the $'$ denotes the transposition operator.

This assumes that the symmetric matrix $(G'G)$ is invertible, which can be guaranteed by a suitable choice of the parameter β, as we shall see in Section 5. In practice, very large systems can be solved using another approach, such as a Conjugate Gradient method. However, the existence of a solution is always guaranteed by the non-singularity of $G'G$.

4. Uniform approximation capability

The uniform convergence proof of the above approximator does not require that $s>0$ (see Section 3.1), thus in this section we can simply consider the case $m=M$. Before proving the uniform convergence of the approximator, we must generalize a result that is well-know for continuous function approximation on compact subsets of R^n. In order to simplify the writing, we state the results for $K=1$, while the generalization to any K is immediate since all output components have similar properties.

Definition 1. The 'nearest known neighborhood' of any point $X \in \Omega$ is the set defined by

$$N(X) = \{X_i \in \chi; \delta(X, X_i) = \min_{1 \leq j \leq m} \delta(X, X_j)\}.$$

The number of points in $N(X)$ is denoted $|N(X)|$, and one has necessarily $|N(X)| \geq 1$.

Definition 2. The 'stepwise approximation' of a function f on Ω is defined by

$$Q_m(X) = \sum_{j=1}^{m} q_i(X) f(X_i),$$

with

$$q_i(X) = \begin{cases} \dfrac{1}{|N(X)|} & \text{if } X_i \in N(X) \\ 0 & \text{otherwise} \end{cases}.$$

Lemma 1. *Under the general conditions stated in Section 2, one has, for any* $\varepsilon > 0$:

$$\lim_{m \to \infty} \text{Prob}(\sup_{X \in \Omega} |f(X) - Q_m(X)| > \varepsilon) = 0.$$

Proof.

(i) Using Section 2.2, for any $X \in \Omega$, $r>0 \Rightarrow \mu(B(X,r))>0$, and thus: for any $r>0$, $\lim_{m\to\infty} \text{Prob}(\min_{1\leq i\leq m} \delta(X, X_i) > r) = \lim_{m\to\infty}(1-\mu(B(X,r)))^m = 0$.

(ii) Using Section 2.3, $\delta(X,X_i) \leq r \Rightarrow |f(X)-f(X_i)| \leq$ a.r, with $a < \infty$, and thus:

$$\begin{aligned} |f(X) - Q_m(X)| &= \left| f(X) - \frac{1}{|N(X)|} \sum_{X_i \in N(X)} f(X_i) \right| \\ &= \frac{1}{|N(X)|} \left| \sum_{X_i \in N(X)} f(X) - f(X_i) \right| \\ &\leq \frac{1}{|N(X)|} \sum_{X_i \in N(X)} |f(X) - f(X_i)| \\ &\leq \frac{1}{|N(X)|} |N(X)| \text{a.r} = \text{a.r}. \end{aligned}$$

(iii) One obtains Lemma 1 from (i) and (ii), with $r = \varepsilon/a$. □

We are now ready to prove the following theorem.

Theorem 1. *Let* $\beta = \beta(m)$ *be a positive increasing function of m such that* $\lim_{m\to\infty} \beta(m) = \infty$. *Then, under the general conditions stated in Section 2, there are synaptic weights* w_i, $1 \leq i \leq m$, *for the approximator* $\varphi(X)$ *defined in* Section 3.4, *such that, for any* $\varepsilon > 0$:

$$\lim_{m \to \infty} \text{Prob}\left(\sup_{X \in \Omega} |f(X) - \varphi(X)| > \varepsilon \right) = 0.$$

Proof. Given Lemma 1, it suffices to take $w_i = f(X_i)$, $1 \leq i \leq m$, and to show that:

$$\lim_{m \to \infty} g_i(X; \beta(m)) = q_i(X).$$

We note that the ith neural basis function can also be written as:

$$g_i(X;\beta) = \frac{1}{\sum_{j=1}^{m} e^{\beta(\delta(X,X_i)-\delta(X,X_j))}}.$$

If $X_i \in N(X)$ then

$$g_i(X;\beta) = \frac{1}{|N(X)| + \sum_{X_j \notin N(X)} e^{\beta(\delta(X,X_i)-\delta(X,X_j))}},$$

where the arguments of the exponentials are all strictly negative, and thus the sum of these exponentials tends to zero as β tends to infinity, which leads to the expected result:

$$g_i(X;\beta) \to \frac{1}{|N(X)|}, \quad \text{as} \quad \beta \to \infty.$$

On the other hand, if $X_i \notin N(X)$ then

$$\begin{aligned} g_i(X;\beta) &\le \sum_{X_k \notin N(X)} g_k(X;\beta) \\ &= 1 - \sum_{X_j \in N(X)} g_j(X;\beta) \to 1 - |N(X)| \frac{1}{|N(X)|} \\ &= 0, \quad \text{as} \quad \beta \to \infty, \end{aligned}$$

which completes the proof. □

5. Interpolation and Least Square approximation

5.1. Interpolation problem

This is the case $m=M$, with a minimum spacing $s>0$ (see Sections 3.1 and 3.5). The computation of synaptic weights reduces to $W=G^{-1}F$, which requires that the $m \times m$ matrix G be invertible.

Theorem 2. *With $m>1$ and $s>0$, there is a real number β_0 such that $0 \le \beta_0 \le \ln(m-1)/s$, and if $\beta > \beta_0$, then the matrix $G=(g_{ij})=(g_j(X_i;\beta))$, $1 \le i,j \le m$, is invertible.*

Proof. After the well-known theorem of Gerschgorin–Hadamard, one knows that a sufficient (but not necessary) condition for a square matrix to be invertible is that the absolute value of each of its diagonal coefficients be greater than the sum of the absolute values of all non-diagonal coefficients in the same row. Given that all coefficients of G are positive and that each row has a sum equal to 1, the theorem of Gerschgorin–Hadamard applies if $g_{ii} > 1/2$, $1 \le i \le m$.

One has:

$$g_{ii} = \frac{1}{1 + \sum_{\substack{j=1 \\ j \neq i}}^{m} e^{-\beta \cdot \delta(X_i,X_j)}} > \frac{1}{2} \Leftrightarrow \sum_{\substack{j=1 \\ j \neq i}}^{m} e^{-\beta \cdot \delta(X_i,X_j)} < 1.$$

On the other hand, one has:

$$\sum_{\substack{j=1 \\ j \neq 1}}^{m} e^{-\beta \cdot \delta(X_i,X_j)} \le (m-1)e^{-\beta s},$$

and finally:

$$(m-1)e^{-\beta s} < 1 \Leftrightarrow \beta > \frac{\ln(m-1)}{s}.$$

Thus the lower bound β_0 is at most equal to $\ln(m-1)/s$, which completes the proof. □

5.2. Least square approximation

This is the case $M>m$, which allows for approximating the objective function while filtering possible data noise. As stated in Section 3.5, one must inverse the symmetric matrix $G'G$, while G is a rectangular $M \times m$ matrix. The following evidence solves the problem.

Lemma 2. *If the square $m \times m$ submatrix of G corresponding to the prototypical examples is invertible, then $G'G$ is invertible.*

Proof. If the submatrix of G corresponding to the prototypical examples is invertible, then its m column vectors are linearly independent, which implies that the m column vectors of G are also linearly independent since there is no non-zero vector u such that $Gu=0$. As a consequence, there is no non-zero vector u such that $u'G'Gu=0$, which means that none of the eigenvalues of $G'G$ is zero, and thus $G'G$ is invertible. □

As one can see, it suffices to apply Theorem 2 to the set of prototypes to be sure that the Least Square approximation problem has a solution ($W=(G'G)^{-1}G'F$). Note however that the above proof assumes that all prototypes actually belong to the learning set.

6. Behavior of the approximator as a function of β

In order to visualize the behavior of the approximator $\varphi(X)$ as a function of β, we interpolated a fixed set of 20 distinct data points on the unit square of R^2, with $\delta(X,Y)=\|X-Y\|^2$ (thus the support space is Euclidean for this visual example), using various values for β. In this case, each coefficient of the matrix G is equal to a Gaussian divided by a sum of Gaussians, and the matrix G is equal to the product of a non-singular diagonal matrix (inverse sums) by a symmetric matrix of Gaussians. This implies that any $\beta>0$ can be used (that is $\beta_0=0$), since such a matrix G is always invertible (Micchelli, 1986). We have $m=20$, $s=0.02$, and thus $\ln(m-1)/s=147.22$. One can see, in Fig. 1, interpolation surfaces obtained for $\beta=150$ (upper panel), and for $\beta=50$ (lower panel). The interpolation surface obtained

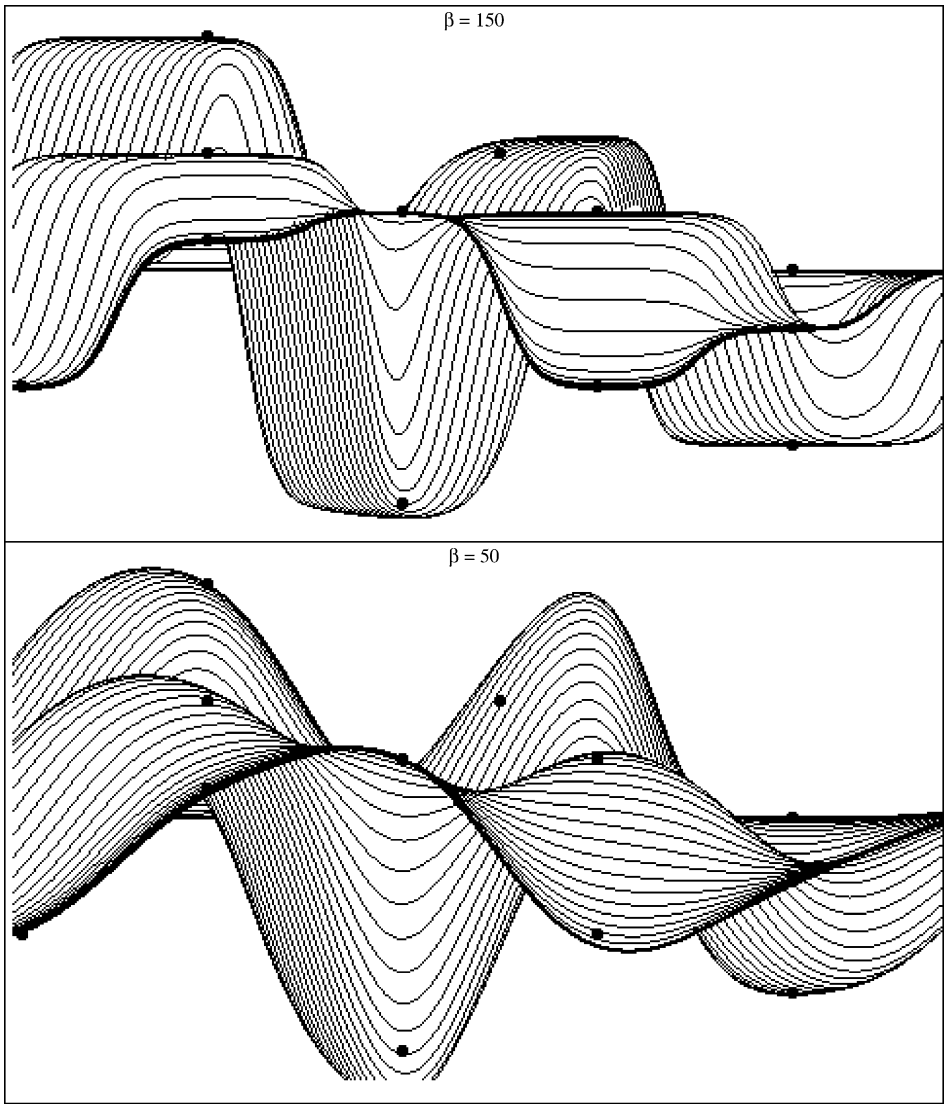


Fig. 1. Two interpolation surfaces of a set of 20 data points on the real unit square, with two different values of β.

with $\beta = 150$ looks like a 'smoothed stepwise approximation', with typical sigmoid profiles between data points. The corresponding minimum diagonal coefficient of G is equal to 0.95, which is unnecessarily large. The interpolation surface obtained with $\beta = 50$ is more regular, although its variation range is wider. The corresponding minimum diagonal coefficient of G is equal to 0.675. Gradually lowering β, one obtains larger and larger oscillations of the interpolator between data points, and clear symptoms of ill-conditioning (of G) for $\beta < 15$, while the minimum diagonal coefficient of G is lower than 0.30.

7. Regularization

One knows that regularizing an approximator is important in order to obtain a good generalization capability from finite samples of data points. Regularizations of approximators on R^n are commonly obtained by minimizing norms of differential operators (Girosi & Poggio, 1990; Poggio & Girosi, 1990). Unfortunately, such operators are not defined for functions on non-metric spaces. So, we must first define some suitable stabilizer usable on such spaces, which requires some reasonably restrictive additional conditions concerning the space (Ω,δ).

7.1. Foundations

Definition 3. With $C(X,r) = \{Y \in \Omega,\ 0 < \delta(X,Y) \leq r\}$, the absolute local variation ratio of an approximator φ at point $X \in \Omega$ is defined as:

$$\Delta\varphi(X) = \lim_{r \to 0} \sup_{Y \in C(X,r)} (|\varphi(X) - \varphi(Y)|/\delta(X,Y)).$$

In the following, we will determine a positive function $V(\beta)$ such that:

$$\sup_{X \in \Omega} \Delta\varphi(X) \leq V(\beta).$$

Then there is a particular β, denoted $\beta*$, such that

$$V(\beta*) = \min_{\beta>\beta''} V(\beta),$$

where β'' is the β value for which $\min_{1\le j\le m} g_{jj} = 0.5$, and thus $\beta > \beta''$ guarantees that the matrix G is invertible since $\beta'' \ge \beta_0$ (see Theorem 2).

In other words, our regularization approach consists of minimizing an upper bound of the absolute local variation ratio of φ on (Ω,δ).

In order to do this, we consider first the case of an exact interpolation ($m=M$, $s>0$), and we require the following two additional conditions:

- For any $X\in\Omega$, for any $r>0$, $C(X,r)$ is not empty (thus $\Delta\varphi(X)$ is defined).
- There is a positive number $\gamma<\infty$ such that, for any $X,Y,Z\in\Omega$

$$|\delta(X,Z) - \delta(Y,Z)| \le \gamma\cdot\delta(X,Y).$$

As we shall see, there is no need for knowing the value of γ, provided that one can assume that it is finite. Note also that in the special case where δ is a metric, one has $\gamma=1$, and the condition is equivalent to the triangle inequality (which, of course, is not required here).

Theorem 3. *Under the above specified conditions, for any $\beta>\beta''$, one has*:

$$\sup_{X\in\Omega} \Delta\varphi(X) \le V(\beta) = \max_{1\le i\le m} |f(X_i)| \cdot \gamma \cdot \frac{\beta}{\min_{1\le j\le m} g_{jj} - 0.5}.$$

Proof. *Step 1*

$$\begin{aligned}|\varphi(X) - \varphi(Y)| &= \left|\sum_{i=1}^m w_i g_i(X;\beta) - \sum_{i=1}^m w_i g_i(Y;\beta)\right| \\ &= \left|\sum_{i=1}^m w_i (g_i(X;\beta) - g_i(Y;\beta))\right| \\ &\le \max_{1\le i\le m} |w_i| \sum_{j=1}^m |g_j(X;\beta) - g_j(Y;\beta)|,\end{aligned}$$

and thus, for any $X\in\Omega$:

$$\Delta\varphi(X) \le \left(\max_{1\le i\le m} |w_i|\right) \cdot \left(\lim_{r\to 0} \sup_{Y\in C(X,r)} \sum_{j=1}^m |g_j(X;\beta) - g_j(Y;\beta)|/\delta(X,Y)\right).$$

Step 2

Given that all prototypes have non-zero spacing, the $m\times m$ matrix G tends to the identity matrix I as β tends to infinity, and thus $W = G^{-1}F$ tends to F. Now, for $\beta<\infty$, one can use a well-known theorem on linear system conditioning (Ciarlet, 1982, pp. 30–31), which gives

$$||F - W||_\infty \le \frac{||I-G||_\infty}{1 - ||I-G||_\infty} ||F||_\infty,$$

that is

$$\max_{1\le i\le m} |f(X_i) - w_i| \le \frac{2(1 - \min_{1\le j\le m} g_{jj})}{1 - 2(1 - \min_{1\le j\le m} g_{jj})} \cdot \max_{1\le i\le m} |f(X_i)|,$$

which implies that

$$\begin{aligned}\max_{1\le i\le m} |w_i| &\le \left(1 + \frac{2(1 - \min_{1\le j\le m} g_{jj})}{1 - 2(1 - \min_{1\le j\le m} g_{jj})}\right) \\ &\quad \cdot \max_{1\le i\le m} |f(X_i)| = \frac{\max_{1\le i\le m} |f(X_i)|}{2\cdot \min_{1\le j\le m} g_{jj} - 1}.\end{aligned}$$

Step 3

First, we note that

$$\begin{aligned}&|\delta(X,Z) - \delta(Y,Z)| \le \gamma\cdot\delta(X,Y) \Rightarrow e^{-\beta\cdot\delta(X,Z)} \\ &\in [e^{-\beta(\delta(Y,Z)+\gamma\cdot\delta(X,Y))}, e^{-\beta(\delta(Y,Z)-\gamma\cdot\delta(X,Y))}] \\ &= [e^{-\beta\gamma\cdot\delta(X,Y)}, e^{\beta\gamma\cdot(X,Y)}]\cdot e^{-\beta\cdot\delta(Y,Z)}.\end{aligned}$$

Here, we use some interval calculation rules (numbered I1–I12, see Appendix):

$$\begin{aligned}&\sum_{i=1}^m |g_i(X;\beta) - g_i(Y;\beta)| \\ &= \sum_{i=1}^m \left|\frac{e^{-\beta\cdot\delta(X,X_i)}}{\sum_{j=1}^m e^{-\beta\cdot\delta(X,X_j)}} - \frac{e^{-\beta\cdot\delta(Y,X_i)}}{\sum_{j=1}^m e^{-\beta\cdot\delta(Y,X_j)}}\right| \\ &\in \sum_{i=1}^m \left|\frac{[e^{-\beta\gamma\cdot\delta(X,Y)}, e^{\beta\gamma\cdot\delta(X,Y)}]e^{-\beta\cdot\delta(Y,X_i)}}{[e^{-\beta\gamma\cdot\delta(X,Y)}, e^{\beta\gamma\cdot\delta(X,Y)}]\cdot\sum_{j=1}^m e^{-\beta\cdot\delta(Y,X_j)}}\right. \\ &\quad \left. - \frac{[1,1]\cdot e^{-\beta\cdot\delta(Y,X_i)}}{[1,1]\cdot\sum_{j=1}^m e^{-\beta\cdot\delta(Y,X_j)}}\right| \quad \text{(I1, I12)} \\ &= \sum_{i=1}^m \left|[e^{-2\beta\gamma\cdot\delta(X,Y)}, e^{2\beta\gamma\cdot\delta(X,Y)}]\frac{e^{-\beta\cdot\delta(Y,X_i)}}{\sum_{j=1}^m e^{-\beta\cdot\delta(Y,X_j)}}\right. \\ &\quad \left. - [1,1]\cdot\frac{e^{-\beta\cdot\delta(Y,X_i)}}{\sum_{j=1}^m e^{-\beta\cdot\delta(Y,X_j)}}\right| \quad \text{(I5, I10)} \\ &= \sum_{i=1}^m \left|[e^{-2\beta\gamma\cdot\delta(X,Y)} - 1, e^{2\beta\gamma\cdot\delta(X,Y)} - 1]\right. \\ &\quad \left.\times\frac{e^{-\beta\cdot\delta(Y,X_i)}}{\sum_{j=1}^m e^{-\beta\cdot\delta(Y,X_j)}}\right| \quad \text{(I11)} \\ &= [0, e^{2\beta\gamma\cdot\delta(X,Y)} - 1]\sum_{i=1}^m g_i(Y;\beta) \quad \text{(I8, I9, I12)}\end{aligned}$$

Since $\sum_{i=1}^{m} g_i(Y;\beta) = 1$, one obtains:

$$\sum_{i=1}^{m} |g_i(X;\beta) - g_i(Y;\beta)| \leq e^{2\beta\gamma\cdot\delta(X,Y)} - 1.$$

Remembering that $(e^u - 1)$ is equivalent to u on the neighborhood of 0, we obtain:

$$\lim_{r\to 0} \sup_{Y\in C(X,r)} \sum_{j=1}^{m} |g_j(X;\beta) - g_j(Y;\beta)|/\delta(X,Y)$$

$$\leq \lim_{r\to 0} \sup_{Y\in C(X,r)} \frac{e^{\beta\cdot 2\gamma\cdot\delta(X,Y)} - 1}{\delta(X,Y)} \approx 2\gamma\cdot\beta.$$

Following Step 1, this last result time the result of Step 2 provides an upper bound of $\Delta\varphi(X)$ on Ω, which completes the proof of Theorem 3. □

We note that the location of a minimizer β^* of $V(\beta)$ depends only on the ratio $\beta/(\min_{1\leq j\leq m} g_{jj} - 0.5)$ since the remaining factors of $V(\beta)$ are constant with respect to β.

7.2. Computation of β^*

In order to examine the behavior of the function $V(\beta)$ defined in Theorem 3, we generated a large set of square matrices with zero diagonal coefficients and strictly positive random out-diagonal coefficients. These coefficients were used as random δ values, and we plotted the corresponding $V(\beta)$ functions for $\beta \geq 0$. It turned out that $V(\beta)$ was always uniminimal on its positive part (that is for $\beta > \beta''$). A typical profile of $V(\beta)$ can be seen in Fig. 2. Unfortunately, we failed to state a formal proof that $V(\beta)$ is necessarily uniminimal on its positive part, so there is a small doubt that could justify the use of a global optimization method in order to minimize $V(\beta)$. As verification, we applied well-known random walk type global optimization algorithms, whose convergence is guaranteed and that usually provide accurate results (Courrieu, 1997; Ingber & Rosen, 1992). This always provided the same result as the following simple local search procedure, where the matrix G is explicitly expressed as a function of β (i.e. $G = G(\beta)$), and the output β^* is a minimizer of V(β) on its positive part.

Procedure 1

function $V(b) = b/(\min_{1\leq j\leq m} g_{jj}(b) - 0.5)$.
$b1 := \ln(m)/s$; $b2 := 1.1*b1$; $b3 := 1.2*b1$;
while $V(b1) < V(b2)$ **do**
  $b3 := b2$; $b2 := b1$; $b1 := 2*b2 - b3$;
  while $\min_{1\leq i\leq m} g_{ii}(b1) \leq 0.5$ **do** $b1 := (b1 + b2)/2$;
end
while $V(b3) < V(b2)$ **do**
  $b1 := b2$; $b2 := b3$; $b3 := 2*b2 - b1$;
end
while $(b3 - b1) >$ precision$*b2$ **do**
  $c1 := (b1 + b2)/2$; $c2 := (b2 + b3)/2$;
  if $V(c1) \leq V(b2)$ **then**
    $b3 := b2$; $b2 := c1$;
  else if $V(b2) \leq V(c2)$ **then**
    $b1 := c1$; $b3 := c2$;
  else
    $b1 := b2$; $b2 := c2$;
  end
end
$\beta^* := b2$.

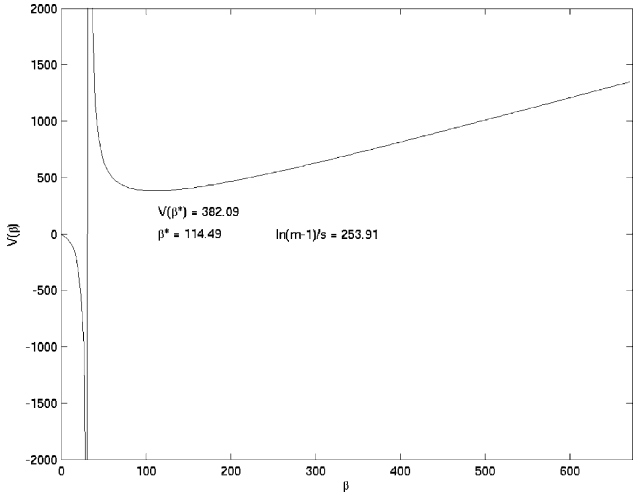


Fig. 2. A typical shape of the function $V(\beta)$.

Note that the above procedure is written to be easily readable, however, in practical implementations, one must of course avoid repeated calls to the $V(\beta)$ function with the same argument, and reusable values must be stored. This procedure has been applied to the illustrative interpolation problem of Section 6. The obtained value was $\beta^* \approx 70$, while the corresponding minimum diagonal coefficient of G was 0.78. The obtained interpolation surface is shown in Fig. 3, where the regularization provided by β^* seems effective. We note that the interpolation surface is close to that obtained with $\beta = 50$ (Fig. 1, lower panel), but that surface oscillations between data points are a bit smaller.

Finally, we note that β^* can be computed in all cases, provided that $s > 0$. In the case of a Least Square approximation ($m < M$), β^* must be computed for the square $m \times m$ submatrix of G corresponding to the prototypical examples. β^* can also be computed whether we know that γ is finite or not, since γ is not actually used in practical computation. Similarly, the condition that $C(X,r)$ is never empty, which excludes discrete spaces, is not necessary for β^* computation. Thus we conclude that β^* can always be used at least as a reasonable default parameter, while its full theoretical justification of course requires the conditions stated in (Section 7.1).

8. The example of affinely invariant pattern functions

There are many kinds of data spaces on which the above function approximation scheme can be used because the

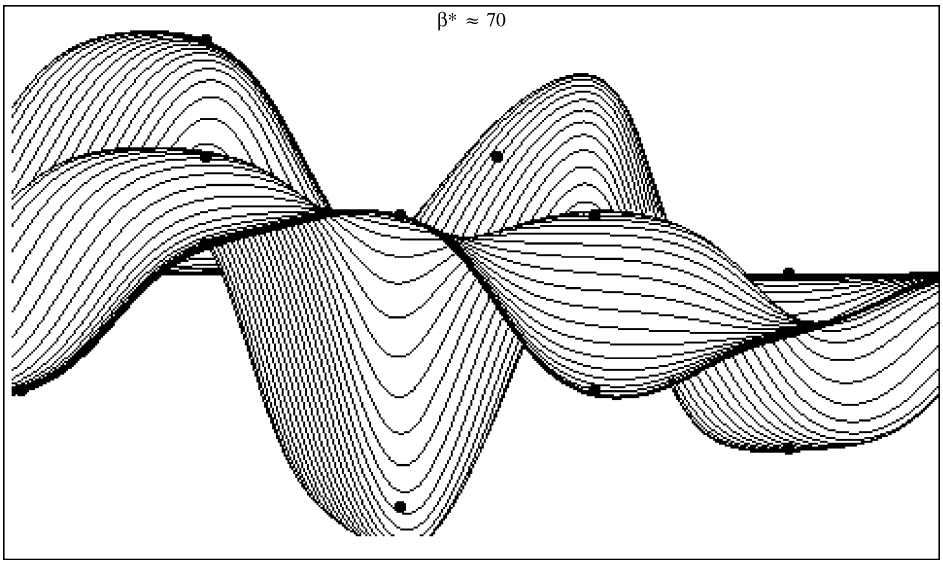


Fig. 3. Interpolation surface of the same data points as those of Fig. 1, with $\beta=\beta^*$.

requirements concerning (Ω,δ) are very weak. As an example, we chose a problem that, at first glance, can seem to be of metric nature, but that in fact is not. This is the problem of approximating a function on a space of patterns, while the function is invariant to affine transformations of the input. Of course, one can ignore this last property and build a metric input space in which each possible pattern is considered as independent of other ones. Another approach consists of taking into account the invariance property in order to improve the generalization and to reduce the required amount of learning, but this leads to build some non-metric input space, as we shall see hereafter.

8.1. Input patterns

We consider here Ω as the set of sequences of L points of a bounded subset of R^n, with $L>n$ Any sequence is represented in the form of a $L\times(n+1)$ real matrix X whose first column coefficients are equal to 1, and the remaining n columns correspond to the coordinates of points. A sequence X belongs to Ω if $\det(X'X)>0$, which means that X is of rank $n+1$, or equivalently that the set of L points is actually of dimension n.

8.2. The δ function

We must define a δ function such that for any $X,Y\in\Omega$, $\delta(X,Y)=0$ if there is an affine transformation T of the n coordinates such that $T(X)=Y$. Given that we added a constant unit coordinate to each point in order to compute translations, it is equivalent to say that $\delta(X,Y)=0$ if there is a square $(n+1)\times(n+1)$ matrix T such that $XT=Y$. The following lemma will be useful.

Lemma 3. *If X and Y are both of rank n+1 and there is T such that XT=Y*, then T is invertible.

Proof. If T is not invertible, then there is a non-zero vector u such that $Tu=0$. Then $XT=Y$ implies that $XTu=Yu=0$, and thus Y is not of rank $n+1$, which contradicts the hypothesis.

Now, one can define δ(X, Y), for example, as the least square error function:

$$\delta(X,Y)=\inf_T ||XT-Y||^2=||(X(X'X)^{-1}X'-I)Y||^2.$$

This δ function obviously satisfies the requirements of Section 2.1, however one can have $\delta(X,Y)=0$ while $X\neq Y$, and if $\delta(X,Y)\neq 0$, then one has in general $\delta(X,Y)\neq\delta(Y,X)$. Thus $\delta(X,Y)$ is certainly not a metric and it cannot by monotonically transformed into a metric (Courrieu, 2002). The first additional requirement of Section 7.1 is satisfied since (Ω,δ) is in fact a continuum. Now, we must verify that δ satisfies the second additional requirement of (Section 7.1) that γ is finite, which allows for applying Theorem 3, and thus theoretically justifies the use of β^*.

Theorem 4. *With Ω and δ defined as above, there is a positive number* $\gamma<\infty$ *such that, for any* $X,Y,Z\in\Omega$, $|\delta(X,Z)-\delta(Y,Z)|\leq\gamma\cdot\delta(X,Y)$.

Proof. Given that all data points belong to a bounded subset of R^n, and that for any $X\in\Omega$, the matrix $X'X$ is invertible, we have that for any $X,Y\in\Omega$, $\delta(X,Y)<\infty$, and thus for any $X,Y,Z\in\Omega$, $|\delta(X,Z)-\delta(Y,Z)|<\infty$. This implies that, if $\delta(X,Y)>0$, then $|\delta(X,Z)-\delta(Y,Z)|/\delta(X,Y)<\infty$. On the other hand, if $\delta(X,Y)=0$, then there is a transformation matrix T such that $XT=Y$, and after Lemma 3, this matrix is invertible, thus $YT^{-1}=X$. Assume that $\delta(X,Z)=||XU-Z||^2$, and $\delta(Y,Z)=||YV-Z||^2$. If $\delta(X,Z)<\delta(Y,Z)$, then $\delta(Y,Z)$ is not the least square solution since $||YT^{-1}U-Z||^2<\delta(Y,Z)$. Similarly, if $\delta(X,Z)>\delta(Y,Z)$, then $\delta(X,Z)$ is not the least square solution since $||XTV-Z||^2<\delta(X,Z)$. Since δ is in all cases the least square error function, we can conclude that if $\delta(X,Y)=0$ then $|\delta(X,Z)-\delta(Y,Z)|=0$, for any $Z\in\Omega$ □.

8.3. Computational test

For this test, we used as input patterns sequences of 4 points of $[0,1]^2$, and sequences of 12 points of $[0,1]^4$, while patterns were encoded as described in Section 8.1. Learning example patterns were randomly generated, while generalization test patterns were generated in the following way. For each generalization input Z, a learning example X was selected, a random affine transformation matrix T and a random pattern R were generated, and finally:

$$Z = (1-\eta)XT + \eta R, \quad 0 \le \eta \le 1.$$

Whenever $\eta = 0$, Z is a random affine transform of a learning example. Whenever $\eta = 1$, Z is completely independent of the learning set. Five η values were used for the test: 0, 0.25, 0.5, 0.75, and 1. The size m of the learning set was varied from 15 to 240, and for each (m,η) combination, 60 generalization patterns were generated. An artificial objective function invariant to affine transformations of the input was built in the following way

$$f(X) = 100/(1 + c\cdot\delta(X, X_0)), \quad c = 28/(\mathrm{nL}),$$

where the δ function is defined as in Section 8.2, and X_0 is a fixed reference pattern that does not belong to the learning set.

For comparison, we tested three types of interpolators. The first one, referred to as 'NN-invar', is the φ approximator with the δ function invariant to affine transformations defined in Section 8.2. The second one, referred to as 'NN-metric', is the φ approximator with a δ function, say δ', that is a squared Euclidean metric on R^{nL}, namely $\delta'(X,Y) = \|X-Y\|^2$. In all cases, φ approximators were tested with $\beta = \beta^*$, where β^* was computed by Procedure 1. Now, δ' can also be used with usual radial basis function approximators, since it is a squared Euclidean metric. We chose Radial Splines as the third type of interpolator. Radial Spline interpolators are of common use, they have well-known uniform approximation and regularization capabilities on Euclidean spaces (Girosi & Poggio, 1990; Poggio & Girosi, 1990), and they do not require any free parameter tuning. Given that nL is always even here, we used Radial Spline basis functions of the form:

$$S(X,Y) = \ln(r)\cdot r^2, \quad \text{with} \quad r^2 = \delta'(X,Y) = ||X-Y||^2.$$

Comparing the generalization performance of NN-invar to that of NN-metric allows for evaluating the interest of using non-metric input spaces in this type of problem. Comparing the generalization performance of NN-metric to that of Spline interpolators (necessarily on a Euclidean space) provides an evaluation of general capabilities of φ approximators with respect to a well-known reference.

Results of the test are reported in Table 1, for $L=4$ and $n=2$, and in Table 2, for $L=12$ and $n=4$. Tables show the mean absolute generalization error (for 60 test items), and the corresponding standard deviation in parenthesis, in the various (m,η) conditions for the three types of interpolators. In addition, Student t-tests were performed in order to test the difference of performance between NN-invar and NN-metric interpolators, and between NN-metric and Spline interpolators. The notation 'n.s' means that the difference between the mean just above and the mean just below 'n.s' is statistically non-significant. The notation $\wedge$ (or $\vee$) means that the difference between means in marginally significant ($p<0.10$). The notation $\wedge\wedge$ (or $\vee\vee$) means that the difference is significant ($p<0.05$), according to usual decision criterions, while the notation $\wedge\wedge\wedge$ (or $\vee\vee\vee$) means that the difference is highly significant ($p<0.01$).

Although there are some visible differences between the processing of the smallest patterns (Table 1) and that of the largest patterns (Table 2), we can make the following general observations. First, the NN-invar interpolator always provides zero generalization error for $\eta = 0$, which simply confirms that this type of network 'recognizes' known patterns independently of affine transformations. For $\eta > 0$, the advantage of NN-invar over NN-metric is not systematic with small learning sets, however, NN-invar

Table 1
Mean absolute generalization error (and standard deviation) of three types of approximations of a function invariant to affine transformations of input sequences of 4 points of $[0,1]^2$

$L=4, n=2$	$\eta=0$	$\eta=0.25$	$\eta=0.50$	$\eta=0.75$	$\eta=1$
$m=15$					
NN-invar	0 (0)	22 (17)	26 (21)	19 (14)	28 (22)
	$\wedge\wedge\wedge$	n.s	n.s	n.s	n.s
NN-metric	15 (11)	24 (15)	23 (15)	19 (13)	25 (13)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
Spline	777 (519)	676 (394)	755 (359)	712 (371)	880 (750)
$m=30$					
NN-invar	0 (0)	12 (14)	11 (11)	12 (10)	12 (12)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
NN-metric	18 (13)	22 (11)	22 (13)	19 (13)	20 (14)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
Spline	85 (54)	64 (40)	62 (41)	60 (48)	102 (81)
$m=60$					
NN-invar	0 (0)	12 (15)	15 (17)	11 (14)	13 (14)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
NN-metric	18 (13)	24 (13)	20 (13)	21 (12)	21 (13)
	n.s	Ú	n.s	n.s	$\vee$
Spline	19 (14)	22 (13)	21 (12)	22 (14)	18 (15)
$m=120$					
NN-invar	0 (0)	6.9 (8.8)	7.8 (9.0)	7.6 (8.7)	8.2 (11)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
NN-metric	18 (15)	20 (14)	20 (15)	17 (13)	15 (12)
	$\wedge\wedge$	n.s	n.s	n.s	$\wedge$
Spline	21 (13)	21 (13)	18 (13)	19 (13)	18 (14)
$m=240$					
NN-invar	0 (0)	5.3 (6.3)	5.1 (6.4)	4.6 (7.2)	3.3 (4.5)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
NN-metric	21 (16)	20 (13)	18 (17)	17 (14)	17 (13)
	n.s	n.s	$\wedge\wedge$	$\vee$	n.s
Spline	22 (14)	19 (13)	22 (15)	15 (12)	16 (11)

The φ approximator 'NN-invar' uses a non-metric input space with affine invariance properties.

Table 2
Similar to Table 1, with input sequences of 12 points of $[0,1]^4$

$L=12, n=4$	$\eta=0$	$\eta=0.25$	$\eta=0.50$	$\eta=0.75$	$\eta=1$
$m=15$					
NN-invar	0 (0)	5.7 (4.6)	8.1 (5.2)	8.4 (4.6)	8.4 (4.7)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	n.s	$\vee\vee$	n.s
NN-metric	9.3 (6.1)	8.5 (6.1)	7.2 (3.7)	7.2 (5.0)	8.0 (4.7)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
Spline	51 (45)	38 (32)	31 (22)	29 (22)	40 (28)
$m=30$					
NN-invar	0 (0)	4.4 (3.7)	6.5 (5.6)	7.5 (6.5)	6.0 (4.9)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	n.s	n.s	n.s
NN-metric	5.6 (3.7)	6.1 (5.0)	6.9 (6.2)	7.8 (7.5)	6.5 (5.8)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
Spline	48 (41)	38 (25)	33 (23)	29 (27)	34 (24)
$m=60$					
NN-invar	0 (0)	4.7 (4.5)	5.9 (5.4)	4.9 (4.2)	5.6 (4.7)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge$
NN-metric	7.2 (7.1)	6.9 (7.2)	7.0 (6.7)	6.0 (4.4)	6.4 (4.7)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
Spline	89 (61)	59 (37)	57 (40)	58 (42)	55 (38)
$m=120$					
NN-invar	0 (0)	5.3 (5.1)	5.2 (4.0)	6.2 (4.5)	6.3 (5.0)
	$\wedge\wedge\wedge$	$\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
NN-metric	6.5 (5.6)	6.9 (7.6)	6.1 (4.4)	7.2 (5.8)	7.1 (5.2)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
Spline	39 (32)	22 (19)	22 (16)	20 (15)	21 (17)
$m=240$					
NN-invar	0 (0)	4.7 (4.5)	5.8 (5.9)	5.5 (3.5)	5.0 (4.5)
	$\wedge\wedge\wedge$	$\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$
NN-metric	7.3 (6.4)	6.3 (5.2)	6.8 (6.8)	6.5 (4.0)	5.8 (4.7)
	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge\wedge\wedge$	$\wedge$	$\wedge$
Spline	10 (10)	8.0 (5.4)	8.7 (7.1)	7.3 (4.0)	6.7 (5.6)

performs systematically better than NN-metric with large enough learning sets ($m \geq 30$ for the smallest patterns, $m \geq 60$ for the largest patterns). In fact, if η is small enough, then NN-invar performs better than NN-metric whatever be m, as one can see in Table 2 for $\eta=0.25$. The explanation is obvious: the invariance of the interpolator to affine transformations is efficient only if the current input is quite close to at least one of the learning examples, modulo an affine transformation. This can be obtained by using a small η, or by using a large learning set as well. Spline interpolators seem to have special difficulties to generalize from small learning sets for this problem, however, their learning is convergent as m increases. This appears to result from the fact that, contrarily to φ approximators, Spline interpolators are not bounded on the extrapolation area, that is, outside the interpolation polytope of the learning set (Courrieu, 1994; Pelillo, 1996). So, a great Spline generalization error can occur whenever a generalization input falls outside the interpolation polytope, which is more probable with small (random) learning sets than with large ones. We note that the φ approximator NN-metric performs at least as well as Spline interpolators, even with the largest learning sets. Thus, we conclude from this test that the approximator presented in this paper is suitable for practical applications, and its particular capability of working on non-metric spaces makes it of special interest.

9. Non-metric δ functions from dynamic programming

Dynamic Programming methods provide another source of non-metric δ functions. These methods allow for comparing numerical sequences or symbol strings of different lengths, and they have been extensively applied in speech recognition (Sakoe & Chiba, 1978). Although the resulting δ functions can have some properties of distances under reasonably restrictive conditions, most of these functions are not metrics, since they usually break the triangle inequality (Okochi & Sakai, 1982). Whenever the triangle inequality is the only distance property that is broken, one can monotonically embed the data set into a Euclidean space (Courrieu, 2002), and then apply a usual neural algorithm on the embedding space. However, it can be more convenient to directly use a φ approximator on the original data space, which avoids prior input data embedding, or possibly undesirable restrictions concerning δ, and does not require prior knowledge of all properties of δ, except those listed in Section 2.1.

As an example, we briefly consider hereafter the well-known algorithm of Sakoe and Chiba (1978). Let Ω be a set of sequences of points of R^k, and let d be some metric associated to R^k. Let $X=(x_1,x_2,\ldots x_m)$, and $Y=(y_1,y_2,\ldots y_n)$ be two sequences, of length $m>1$ and $n>1$, respectively, belonging to Ω. The algorithm statement is:

$D(1,1)=2d(x_1,y_1)$;
for $j=2\ldots n$ $D(1,j)=D(1,j-1)+d(x_1,y_j)$;
for $i=2\ldots m$ $D(i,1)=D(i-1,1)+d(x_i,y_1)$;
for $i=2\ldots m$
  for $j=2\ldots$n $D(i,j)=\min[D(i,j-1)+d(x_i,y_j)$,
  $D(i-1,j-1)+2d(x_i,y_j)$, $D(i-1,j)+d(x_i,y_j)]$;
$\delta(X,Y)=D(m,n)/(m+n)$.

The above δ function is insensitive to repetition (e.g. $\delta((1,1,1,2,3,3),(1,2,2,3))=0$), which makes it suitable for comparing sequences of regularly sampled acoustical parameters in speech recognition, given that speech speed is naturally variable. As an example of triangle inequality breaking, consider $X=(1,2,3)$, $Y=(4,5,6)$, $Z=(2,5)$. Then one obtains $\delta(X, Y)=2.333\ldots$, $\delta(X,Z)=1$, $\delta(Z,Y)=1$, thus $\delta(X,Y)>\delta(X,Z)+\delta(Z,Y)$. The above algorithm can as well be used to compare symbol strings provided that one can define some natural distance d between the elements of the used alphabet (Courrieu, Farioli, & Grainger, in press). Its behavior is quite different from that of well-known 'edition distances' for character strings (Lowrance & Wagner, 1975; Wagner & Fischer, 1974).

10. Conclusion

We have defined a function approximation scheme that can be expressed as a layered feed-forward neural network, and that is able to uniformly approximate functions on a wide

variety of non-metric spaces as well as on any metric space, while usual approximators are guaranteed to work only on real Euclidean spaces. Non-Euclidean metric or non-metric data spaces are commonly encountered in practical applications. For example, time series can be compared using Dynamic Programming methods (elastic matching), but the resulting space is not metric, in general. The same is true for spaces of symbol strings, or for spaces of graphs. Another example is that of spaces of real patterns invariant to some class of transformations, while the particular case of affine invariance has been detailed and illustrated above. Hence, the proposed tool clearly responds to practical needs that have been poorly investigated previously. Theoretical foundations are provided concerning the uniform approximation capability of the approximator, the solution of interpolation and least square approximation problems, and an approach of regularization suitable to the considered data spaces. All required practical algorithms are simple, and computational examples are provided that clearly show the suitability of the tool. As a final remark, we note that φ approximators have conventional architectures, and that their specificity simply resides in their special basis functions, although these basis functions themselves do not have an extremely 'exotic' form, except that they accept very weakly constrained δ functions as arguments, where Radial Basis Functions, for example, require Euclidean metrics. This leads us to suspect that other approximators with similar capabilities could exist, however, their theoretical investigation remains to do. This is of interest because the result is, in fact, a significant extension of the concept of 'universal approximation capability'.

Appendix. Interval calculation

Interval calculation has been developed by Moore (1966) and Ratschek and Rokra (1984). Interval arithmetic is also reported in Zhigljavsky (1991). We list hereafter some useful rules (I1–I12) whose consistency is easy to verify. We consider here closed real intervals of the form $Z_i = [a_i,b_i]$, with $a_i \le b_i$.

I1. For $x \in R$, $x = [x,x] = [1,1]\cdot x$.

Interval arithmetic

I2. $Z_1 + Z_2 = [a_1 + a_2, b_1 + b_2]$
I3. $Z_1 - Z_2 = [a_1 - b_2, b_1 - a_2]$
I4. $Z_1 \cdot Z_2 = [\min(a_1a_2, a_1b_2, b_1a_2, b_1b_2), \max(a_1a_2, a_1b_2, b_1a_2, b_1b_2)]$
I5. $Z_1/Z_2 = Z_1 \cdot [1/b_2, 1/a_2]$, if $0 \notin Z_2$.

Interval functions

I6. If h is a monotonic increasing function, then $h(Z) = [h(a),h(b)]$.

I7. If h is a monotonic decreasing function, then $h(Z) = [h(b),h(a)]$.

I8. The absolute value function of an interval is given by:

$$|[a,b]| = \begin{cases} [0, \max(|a|,|b|)] & \text{if } ab \le 0 \\ [\min(|a|,|b|), \max(|a|,|b|)] & \text{otherwise} \end{cases},$$

Intervals and numbers

I9. $|[a,b]\cdot x| = |[a,b]|\cdot|x|$
I10. $\dfrac{[a_1,b_1]\cdot x_1}{[a_2,b_2]\cdot x_2} = \dfrac{[a_1,b_1]}{[a_2,b_2]} \cdot \dfrac{x_1}{x_2}$, if $a_1, b_1 \ge 0$, and $a_2, b_2, x_2 > 0$

I11. $[a,b]\cdot x - [1,1]\cdot x = [a-1, b-1]\cdot x$
I12. $\sum_{i=1}^{m} [a,b]x_i = [a,b]\cdot \sum_{i=1}^{m} x_i$, if $x_i \ge 0$, $l \le i \le m$

References

Ciarlet, P. G. (1982). *Introduction à l'Analyse Numérique Matricielle et à l'Optimisation*. Paris: Masson.

Courrieu, P. (1993). A convergent generator of neural networks. *Neural Networks*, *6*, 835–844.

Courrieu, P. (1994). Three algorithms for estimating the domain of validity of feedforward neural networks. *Neural Networks*, *7*, 169–174.

Courrieu, P. (1997). The Hyperbell algorithm for global optimization: a random walk using Cauchy densities. *Journal of Global Optimization*, *10*, 37–55.

Courrieu, P. (2001). Two methods for encoding clusters. *Neural Networks*, *14*, 175–183.

Courrieu, P. (2002). Straight monotonic embedding of data sets in Euclidean spaces. *Neural Networks*, *15*, 1185–1196.

Courrieu, P., Farioli, F., & Grainger, J. Inverse discrimination time as a perceptual distance for alphabetic characters. *Visual Cognition*, 11(7), 601–919.

Cybenko, G. (1989). Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, *2*, 303–314.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning algorithm. *Advances in neural information processing systems* (Vol. 2) (pp. 525–532). San Mateo, CA: Morgan Kauffman, 525–532.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2*, 183–192.

Girosi, F., & Poggio, T. (1990). Networks and the best approximation property. *Biological Cybernetics*, *63*, 169–176.

Hornik, K. (1993). Some new results on neural network approximation. *Neural Networks*, *6*, 1069–1072.

Ingber, L., & Rosen, B. (1992). Genetic algorithms and very fast simulated reannealing: a comparison. *Mathematical and Computer Modelling*, *16*, 87–100.

Kreinovich, V. Y. (1991). Arbitrary nonlinearity is sufficient to represent all functions by neural networks: a theorem. *Neural Networks*, *4*, 381–383.

Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, *6*, 861–867.

Lowrance, R., & Wagner, R. A. (1975). An extension of the string to string correction problem. *JACM*, *22*(2), 177–183.

Micchelli, C. A. (1986). Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, *2*, 11–22.

Moore, R. E. (1966). *Interval analysis*. Englewood Cliffs, NJ: Prentice-Hall.

Okochi, M., & Sakai, T. (1982). Trapezoidal D.P. matching with time reversibility. *Proceedings of the IEEE-ASSP Conference (Paris)* , 1239–1242.

Pelillo, M. (1996). A relaxation algorithm for estimating the domain of validity of feedforward neural networks. *Neural Processing Letters*, *3*, 113–121.

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1481–1497.

Ratschek, H., & Rokra, J. (1984). *Computer methods for the range of functions*. New York: Ellis Harwood/Wiley.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 318–362). Cambridge, MA: MIT Press, 318–362.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on ASSP*, *26*(1), 194–200.

Wagner, R. A., & Fischer, M. J. (1974). The string to string correction problem. *JACM*, *21*(1), 168–173.

Yoon, J. (2001). Interpolation by radial basis functions on Sobolev space. *Journal of Approximation Theory*, *112*, 1–15.

Zhigljavsky, A. A. (1991). *Theory of global random search*. Dordrecht: Kluwer.

II.E Méthodes de calcul des paramètres de modèles

La modélisation numérique fait appel à des techniques de calcul sophistiquées, soit pour calculer les paramètres internes des modèles, soit pour déterminer leur domaine d'utilisation. C'est là, on s'en doute, un terrain de jeu de choix pour les mathématiques appliquées. La validation expérimentale des modèles fait, pour sa part, appel à des techniques statistiques spécialisées qui seront évoquées dans la section II.F.

Polytopes convexes et extériorité d'un point

J'ai eu l'occasion de montrer que les réseaux de neurones généralisent efficacement en des points d'interpolation, c'est-à-dire, des points situés à l'intérieur du polytope enveloppe convexe de l'ensemble des exemples d'apprentissage dans l'espace d'entrée. Par ailleurs, les performances de généralisation se dégradent lorsque le point de généralisation est extérieur à ce polytope (extrapolation), et ce d'autant plus que le point de généralisation est plus éloigné du point intérieur au polytope le plus proche. Ceci m'a conduit à définir "l'extériorité" d'un point de généralisation comme la distance euclidienne de ce point à son plus proche voisin intérieur au polytope d'interpolation. J'ai proposé un algorithme efficace pour calculer le polytope d'interpolation et l'extériorité de tout point de généralisation à ce polytope (Courrieu, 1994a). Cette technique s'applique aussi à d'autres problèmes, et elle peut notamment permettre de calculer la probabilité optimale, en regard des données, de chaque système dans les modèles multi-systèmes (Ashby, Alfonso-Reese, Turken, & Waldron, 1998).

Formellement, le problème se présente ainsi. Etant donné un ensemble $D = \{X_1, X_2, ..., X_m\}$ de m points $X_i \in R^n$, $1 \le i \le m$, et un point $Y \in R^n$ quelconque, trouver un vecteur réel $P(Y,D) = (p_1, p_2, ..., p_m)$, tel que:

$$p_i \ge 0,\ 1 \le i \le m,$$

$$\sum_{i=1}^{m} p_i = 1,$$

et tel que l'extériorité du point Y au nuage D, définie par

$$E(Y,D) = \min_P \left\| Y - \sum_{i=1}^{m} p_i X_i \right\|,$$

soit minimale.

La solution très simple que j'ai proposée (Courrieu, 1994a, Algorithme 1) consiste à remplacer le problème avec contraintes ci-dessus par le problème sans contraintes suivant. Trouver un vecteur réel $W(Y,D) = (w_1, w_2, ..., w_m)$ tel que:

$$E^2(Y,D) = \min_W \left\| Y - \sum_{i=1}^{m} w_i^2 X_i \Big/ \sum_{j=1}^{m} w_j^2 \right\|^2,$$

soit minimal. Ce problème peut être résolu par une simple minimisation locale (ex. descente de gradient partant d'un W initial sans composante nulle), et l'on obtient ensuite aisément:

$$p_i = w_i^2 \Big/ \sum_{j=1}^{m} w_j^2,\ 1 \le i \le m.$$

Cette solution est une alternative pratique à des approches plus complexes que l'on trouve en Programmation Mathematique (Clarkson, 2008; Pelillo, 1996). Il existe également une variante technique assez facile à calculer de l'extériorité, appelée "distance de polytope" (Gärtner, & Jaggi, 2009), mais cette approche ne fournit pas les poids (p_i). A noter que les sommets du polytope enveloppe convexe du nuage D sont les points de D dont l'extériorité au nuage des autres points est non nulle, ce qui fournit une méthode pratique pour déterminer le polytope enveloppe convexe.

A titre d'exemple, supposons que Y est un vecteur d'observations (moyennes ou fréquences), et que D est un ensemble de vecteurs de prédictions générés par m systèmes distincts, supposés indépendants. Alors on peut calculer $P(Y,D)$ qui est, dans ce cas, un vecteur de probabilités associées aux différents systèmes, au sein d'un modèle multi-systèmes, de façon à minimiser l'erreur de prédiction globale évaluée par l'extériorité $E(Y,D)$.

Optimisation globale

Dans la modélisation numérique, il est courant que l'on doive déterminer les valeurs de paramètres d'un modèle de façon à minimiser (ou parfois maximiser) une certaine fonction, par exemple une fonction d'erreur des prédictions du modèle relativement à des données, fonction éventuellement compliquée par un certain nombre des contraintes de domaine et/ou de régularisation. Quand on est très astucieux, on arrive parfois à construire le modèle et la fonction à optimiser de telle manière que ladite fonction ne possède qu'un seul optimum global, et aucun extremum "local". Les extrema locaux correspondent à des points de l'espace des paramètres pour lesquels la fonction prend une valeur localement optimale, relativement au voisinage proche de ces points, mais la valeur obtenue n'est pas globalement optimale pour l'ensemble du domaine de recherche des paramètres. Les extrema locaux sont des pièges pour

les méthodes de recherche "locales" des valeurs de paramètres (ex. descente de gradient), méthodes qui sont par ailleurs assez simples et efficaces lorsqu'il n'y a pas d'extrema locaux. Malheureusement, l'astuce du modélisateur a ses limites, et il est des problèmes où l'on ne sait pas éviter la présence d'extrema locaux. C'est par exemple le cas lorsqu'on calcule les poids synaptiques des unités cachées dans un réseau de neurones multicouche ou de type "Cascade-Correlation". On est alors obligé de recourir à des techniques d'optimisation globale capables d'échapper aux pièges des extrema locaux pour converger vers un optimum global. Il n'existe aucune méthode générale permettant d'obtenir à coup sûr le bon résultat en un temps fini, mais il existe des méthodes permettant d'approcher ce résultat, et dont on peut garantir la convergence "à la limite" grâce, notamment, à un théorème dû à Solis et Wets (1981). J'ai proposé un premier algorithme d'optimisation globale, appelé "Recherche Distribuée", qui était assez performant mais vorace en espace mémoire, ce qui le rendait peu pratique pour des problèmes comportant un grand nombre de variables comme le calcul des réseaux de neurones (Courrieu, 1993b). J'ai par la suite développé un algorithme beaucoup plus économe et non moins efficace, dénommé "Hypercloche", que j'utilise aujourd'hui encore, notamment pour calculer les unités cachées des réseaux dans les procédures d'apprentissage de type "Cascade-Correlation" (Courrieu, 1997, article ci-joint).

Méthodes de moindres carrés

Les techniques de moindres carrés sont d'un usage très courant lorsqu'on désire minimiser une fonction d'erreur quadratique. Les récentes sophistications de ces techniques ("iteratively weighted least squares") sont également utilisées au sein de méthodes de calcul de "statistiques robustes" dépassant le cadre des méthodes de moindres carrés proprement dites (Maronna, Martin, & Yohai, 2006). En ce qui concerne la modélisation neurocomputationnelle, les techniques de moindres carrés sont habituellement utilisées pour calculer les poids synaptiques des neurones de sortie des réseaux feedforward.

Dans le cadre d'applications en reconnaissance de formes, diverses variantes des techniques de moindres carrés sont utiles pour calculer des mesures de similitude entre formes. Je me suis donc assuré, dans un premier temps, qu'il serait théoriquement possible, pour des réseaux de neurones biologiques (récurrents dans ce cas), de résoudre des systèmes moindres carrés conséquents dans des temps compatibles avec les performances perceptives connues, disons en moins de 250 millisecondes (Courrieu, 2004). Il ne serait donc pas a priori

complètement extravagant de modéliser des processus de reconnaissance visuelle de formes à l'aide de systèmes moindres carrés.

La résolution d'un système moindres carrés est une opération assez simple lorsque la matrice du système est "de plein rang", mais il n'est pas rare en pratique de rencontrer des systèmes "de rang déficient", surtout lorsqu'il s'agit de systèmes pondérés. Dans tous les cas, la solution consiste à calculer une matrice "inverse généralisée", également appelée "pseudo-inverse", la plus connue de ces inverses généralisées étant l'inverse de Moore-Penrose (Ben-Israel & Greville, 2003). J'ai proposé un algorithme qui est à ce jour, et à ma connaissance, l'algorithme numérique le plus rapide pour calculer l'inverse de Moore-Penrose (Courrieu, 2005b). Je ne joins pas cet article au dossier car l'algorithme est également décrit dans l'article dont il sera question au paragraphe suivant.

L'inverse de Moore-Penrose possède des propriétés tout à fait remarquables, mais elle n'est pas la seule inverse généralisée qui permette de résoudre des systèmes moindres carrés. D'autres inverses généralisées, désignées comme "{1, 3}-inverses", font tout aussi bien l'affaire. J'ai pu définir un algorithme de calcul de l'une ces inverses, qui est à la fois plus rapide et numériquement plus stable que le calcul de l'inverse de Moore-Penrose (Courrieu, 2009, article ci-joint). J'ai utilisé cet outil pour développer une méthode de résolution de "systèmes moindres carrés à appariement pondéré". Ce sont des systèmes que l'on rencontre, par exemple, lorsqu'on veut mesurer la similitude de deux ensembles de points (tels que deux codes de densité représentant des formes), en utilisant une méthode de régression, mais qu'on ne connaît pas précisément les correspondances entre les points des deux ensembles. On associe alors à chaque paire de points possible une certaine mesure de vraisemblance (poids), ce qui donne un système moindres carrés pondéré dont la matrice de poids est pleine et rectangulaire (au lieu d'être diagonale comme dans les systèmes pondérés classiques). Il n'y a alors plus qu'à résoudre le système pour obtenir la mesure désirée.

Références

Ashby, F. G., Alfonso-Reese, L. A., Turken A. U., & Waldron, E. M. (1998). A neuropsychological theory of multiple systems in category learning. *Psychological Review*, *105*, 442-481.

Ben-Israel, A., & Greville, T.N.E. (2003). *Generalized Inverses: Theory and Applications (2nd ed.)*. New York, Springer-Verlag.

Clarkson, K.L. (2008). Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. SODA'08: Proceedings of the nineteenth annual ACM-SIAM Symposium on Discrete Algorithms.

Courrieu, P. (1993b). A distributed search algorithm for global optimization on numerical spaces. *RAIRO: Recherche Opérationnelle / Operations Research, 27*, 281-292.

Courrieu, P. (1994a). Three algorithms for estimating the domain of validity of feedforward neural networks. *Neural Networks, 7*, 169-174.

Courrieu, P. (1997). The Hyperbell algorithm for global optimization: a random walk using Cauchy densities. *Journal of Global Optimization, 10*, 37-55.

Courrieu, P. (2004). Solving time of least square systems in Sigma-Pi unit networks. *Neural Information Processing - Letters and Reviews, 4(3)*, 39-45.

Courrieu, P. (2005b). Fast computation of Moore-Penrose inverse matrices. *Neural Information Processing - Letters and Reviews, 8(2)*, 25-29.

Courrieu, P. (2009). Fast solving of Weighted Pairing Least-Squares systems. *Journal of Computational and Applied Mathematics, 231*, 39-48.

Gärtner, B., & Jaggi, M. (2009). Coresets for polytope distance. *ACM, Proceedings of the 25th annual symposium on Computational geometry*, 33-42.

Maronna, R.A., Martin, R.D., & Yohai, V.J. (2006). *Robust Statistics: Theory and Methods*. New York, John Wiley & Sons Ltd.

Pelillo, M. (1996). A relaxation algorithm for estimating the domain of validity of feedforward neural networks. *Neural Processing Letters, 3*, 113-121.

Solis, F.J.,Wets, R.J-B. (1981). Minimization by random search techniques, *Mathematics of Operations Research, 6(1)*, 19– 30.

Journal of Global Optimization **10:** 37–55, 1997.

The Hyperbell Algorithm for Global Optimization: A Random Walk Using Cauchy Densities

PIERRE COURRIEU
CREPCO (URA CNRS 182), Université de Provence 29 avenue Robert Schuman, F-13621 Aix-en-Provence Cedex 1, France

(Received: 16 May 1995; accepted: 28 May 1996)

Abstract. This article presents a new algorithm, called the "Hyperbell Algorithm", that searches for the global extrema of numerical functions of numerical variables. The algorithm relies on the principle of a monotone improving random walk whose steps are generated around the current position according to a gradually scaled down Cauchy distribution. The convergence of the algorithm is proven and its rate of convergence is discussed. Its performance is tested on some "hard" test functions and compared to that of other recent algorithms and possible variants. An experimental study of complexity is also provided, and simple tuning procedures for applications are proposed.

Key words: global optimization, random search, Cauchy distributions.

1. Introduction

Optimization problems vary in difficulty, depending on the properties of the function to be optimized. Uniextremal functions are generally the easiest to optimize because "local search" procedures can be used. These procedures, such as usual gradient based methods among others, are efficient and guarantee finding the solution. But many functions encountered in practice have multiple extrema, making it necessary to use "global search" methods, which can generally only guarantee finding the solution with a given degree of probability. Extensive efforts have been made within the past few years to solve hard optimization problems (see Horst & Pardalos, 1995). Certain recent methods are deterministic (Baritompa, 1993; Breiman & Cutler, 1993; Wood, 1992), but they will not be considered here. Most of the known approaches in global optimization are based on stochastic processes (see Zhigljavsky, 1991). Certain algorithms rely on the principle of a random walk converging to an extremum of the objective function (Dekker & Aarts, 1991; Romeijn & Smith, 1994; Solis & Wets, 1981; Zabinsky *et al*., 1993). In other methods, global distributions of probabilities represented by samples of points ("populations") belonging to the search domain are made to converge (Courrieu, 1993; Goldberg, 1989; Holland, 1975). In still other methods, convergence is achieved by the global distribution control of a set of local searches (Boender, 1982; Rinnooy Kan & Timmer, 1987a, 1987b). Branch-and-Bound methods (Pintér, 1988; Zhigljavsky, 1991) construct a partition of the search domain and

sequentially reject those subsets which have the lowest probability of containing a global optimum. Other approaches exist, although many of them appear to be more interesting from a theoretical than practical point of view.

Presented below is a new stochastic global optimization algorithm, which will be called the "Hyperbell Algorithm". This algorithm relies on the principle of a monotone improving random walk whose steps are generated around the current position according to an n-dimensional Cauchy distribution which is gradually scaled down. The name "Hyperbell" refers to the shape of the probability density function. The algorithm is applicable to the search for the global extrema of a numerical function f defined on a bounded domain Ω of $\mathbb{R}^n$. It is of interest to situate this approach within the theoretical framework proposed by Solis and Wets (1981) because the Hyperbell Algorithm is compatible with their "conceptual algorithm". The choice of Cauchy distributions is motivated by the search for a good compromise between the speed and the guarantee of convergence. As pointed out by Solis and Wets, guaranteeing convergence requires that the sampling strategy must not indefinitely ignore any subset (which has a positive Lebesgue mesure) of the search domain . Using a uniform distribution on the whole search domain would satisfy this requirement, but unfortunately this leads to very slow convergence. In order to improve the speed of convergence, a common strategy is to gradually concentrate the search on the most "promising" regions. This is the strategy of Branch-and-Bound methods, for example, but the definitive rejection of certain subsets of the search domain does not enable one to guarantee convergence in all cases. Hence, we must concentrate the search on certain subsets while not completely ignoring any subset of the search domain. This can be done by using a bell- shaped probability density of sampling centered on the current position in the search domain, and whose scale can be controlled. Certain common bell-shaped densities, like Gauss or logistic densities, decrease exponentially as the distance from their center increases (in scale units). Hence, the density rapidly tends to zero after a certain distance, resulting in a distribution whose support is "quasi-bounded", and whose behavior for finite samplings is very similar to that of bounded support distributions. This results in a relatively high probability for the search to be trapped in local extrema basins of many functions, and there is poor guarantee of convergence in practice. Cauchy densities are more appropriate because the density decreases very slowly as the distance from the center of the distribution increases. Hence the probability of sampling is truly non-zero for any subset of the search domain which has a non-zero Lebesgue mesure. The position and the quartile deviations of Cauchy densities are easy to control in order to concentrate the search on appropriate regions. Note that the use of Cauchy distributions was previously shown to be more effective than the use of exponential distributions in the framework of Simulated Annealing algorithms (see Ingber, 1989; Ingber & Rosen, 1992). However, the Hyperbell algorithm is monotonically improving and does not use any concept equivalent to the "temperature" of Simulated Annealing.

2. Hyperbell Algorithm

The algorithm is presented in Pseudo-Pascal. The term "random", followed by an interval, denotes a random value taken from a uniform distribution of probabilities on the specified interval. The algorithm shown here concerns the search for minima of a function defined on a bounded feasible region Ω of $\mathbb{R}^n$ (remember that the search for maxima of a function f is equivalent to the search for minima of $-f$). In order to initialize the Cauchy distribution scales $(s_1, s_2, \ldots, s_n)$, it is useful to determine the upper and lower boundary of each variable such that $a_i \leqslant x_i \leqslant b_i$ and Ω is included in the hyperrectangle $\prod_{i=1}^{n}[a_i, b_i]$.

{***Parameters:***
$\alpha \in [0.8, 1)$: scale reduction parameter;
$\varepsilon > 0$: arbitrarily small constant;
DLS : variant selector}

{***Scale initialization (indicative example)***}
for $i := 1$ **to** n **do** $s_i := \dfrac{b_i - a_i}{2tg(\pi(0.5)^{1/n}/2)}$;

{***Starting point***}
repeat
for $i := 1$ **to** n **do** $x_i :=$ random $[a_i, b_i]$;
until $(X \in \Omega)$;

{***Search***}
repeat
  repeat
    for $i := 1$ **to** n **do** $y_i := s_i$ tangent(π random$(-1/2, 1/2)) + x_i$;
  until $(Y \in \Omega)$;
  if DLS **then** $Y := Y - r\nabla f(Y)$; {compute r using a bisection method}
  if $f(Y) < f(X)$ **then** $X := Y$ **else for** $i := 1$ **to** n **do** $s_i := \alpha(s_i - \varepsilon) + \varepsilon$;
until (stopping rule);

The Parameters

The parameter α is the reduction coefficient of Cauchy's distribution scale, when this scale is reduced. Increasing α lowers the speed of convergence and the probability of being trapped in local minima of complex functions. Note that function complexity should not be assessed solely on the basis of the number of local extrema. We shall see in the experimental section below that there are functions with an infinite number of local minima which are easy to optimize with the Hyperbell Algorithm. The parameter ε has little effect in practice, but it guarantees that the s_i scales will not drop to zero and thereby prevents the process from freezing

indefinitely. ε is arbitrarily small and it is generally chosen to be smaller than the desired precision for the result in Ω .

About Scale Initialization

The example of scale initialization given in the algorithm statement corresponds to a probability of 1/2 of generating a new point inside the critical hyperrectangle if the starting point is the center of this hyperrectangle. This choice may seem somewhat arbitrary, however it was empirically found to be usually appropriate. Note that ε must be chosen to be smaller than the initial scales.

Generation of Points

The coordinates of the sampled points are generated independently by means of the generating function: $y_i = s_i tg(\pi u_i) + x_i$, with u_i taken at random from a uniform distribution on $(-1/2, 1/2)$. The random variable defined as such obeys an n-dimensional Cauchy law with the density:

$$g_n(Y; X, S) = \prod_{i=1}^{n} \frac{1}{\pi s_i} \frac{1}{1 + \left(\dfrac{y_i - x_i}{s_i}\right)^2} .$$

The point X is the center of the distribution (the x_i's are the modes and medians of the marginal distributions), and corresponds to the current position of the random walk in the search domain. The scale parameters s_i are the quartile deviations. Cauchy's law has no moments, and in particular its variance is infinite, which is a reflection of the fact that the density decreases slowly and is never negligible. Given that points generated outside Ω are rejected, the density is in fact a conditional distribution, denoted h, given by:

$$\begin{aligned} h_n(Y; X, S) &= \frac{g_n(Y; X, S)}{\int_\Omega g_n(Z; X, S)\, \mathrm{d}Z} \geqslant g_n(Y; X, S), \quad \text{for } Y \in \Omega, \\ h_n(Y; X, S) &= 0, \quad \text{for } Y \notin \Omega. \end{aligned}$$

DLS Variant

Certain functions to be optimized have special properties, making the use of particular local transformations of the generated points interesting. One of these transformations, which we will refer to as the "Directional Local Search" (DLS) variant, consists simply of one standard local search step: $Y := Y - r\nabla f(Y)$, where the step length r is positive, and is determined using a standard bisection method (see Zhigljavsky, 1991, pp. 21–22) with the constraint that $Y \in \Omega$. The gradient calculation supposes that the function is C^1 continuous and is easily derivable, but

an approximation can also be used by replacing the partial derivatives with the corresponding finite increase ratios.

Stopping Rules

Various stopping rules can be used, depending on the requirements for the application. One can simply limit the number of function calls (consumption criterion), or wait until the generation scales are close to ε (precision criterion on Ω).

Convergence

THEOREM 1. *Let f be a function defined on a bounded subset Ω of $\mathbb{R}^n$. Let $X_t \in \Omega$ be the point generated by the Hyperbell Algorithm at time t of the search for a minimum of f on Ω. If f is continuous at a global minimum then:* $\forall \delta > 0, \lim_{k\to\infty} Prob\{\exists t \leqslant k; |f(X_t) - \min_{X\in\Omega} f(X)| < \delta\} = 1$. *(Replace "min" with "max" for a maximum search).*

Proof.

(1)- $\forall i \leqslant n; s_i(t+1) \geqslant \alpha(s_i(t) - \varepsilon) + \varepsilon \Rightarrow \forall i \leqslant n; \forall t; s_i(t) \geqslant \varepsilon > 0$.

(2)- If Ω is bounded then $\forall X, Y \in \Omega; \|Y - X\| < \infty$.

(3)- (1) and (2) $\Rightarrow \exists \lambda > 0; \forall t; \forall X, Y \in \Omega; h_n(Y; X, S(t)) \geqslant \lambda$.

(4)- Set $\Omega_\delta = \{Y \in \Omega; |f(Y) - \min_{Z\in\Omega} f(Z)| < \delta\}$,
let μ be the Lebesgue measure on the subsets of $\mathbb{R}^n$, if f is continuous at a global minimum then $\forall \delta > 0; \mu(\Omega_\delta) > 0$.

(5)- (3) and (4) $\Rightarrow \int_{\Omega_\delta} h_n(Y; X, S)\, dY \geqslant \lambda\mu(\Omega_\delta) > 0$.

(6)- (5) $\Rightarrow \forall \delta > 0, \text{Prob}\{\exists t \leqslant k; |f(X_t) - \min_{X\in\Omega} f(X)| < \delta\} \geqslant 1 - (1 - \lambda\mu(\Omega_\delta))^k$.

(7)- Finally $\lim_{k\to\infty} 1 - (1 - \lambda\mu(\Omega_\delta))^k = 1$, which completes the proof. □

The situation is very similar using the DLS variant of the algorithm, since this variant is only an additional process which locally favours sampling the best points.

One can also use the global search convergence theorem of Solis and Wets (1981) since the Hyperbell Algorithm is clearly a case of their "conceptual algorithm" and it satisfies their conditions $H1$ and $H2$. The condition $H1$ is trivially satisfied by the acceptance rule of steps. Proving $H2$ is quite similar to the above proof, where one replaces Ω_δ by any subset of Ω with non-zero Lebesgue mesure.

Concerning the rate of convergence with Cauchy distributions, one can use equation 11 from Ingber (1989), or Ingber and Rosen (1992), and conclude that it is a sufficient condition, for obtaining stochastic convergence in any problem, that the scales decrease no faster than $s(0)/t^{1/n}$. However, this limit rate does not take into account properties of the problem other than the dimension (n), and faster convergence can almost always be obtained in practice. This usually requires the use of "free parameters", like α for the Hyperbell algorithm whose stochastic

convergence was stated above. Let ε tend to 0, then the ith scale at time t is approximately equal to $s_i(0)\alpha^{t-w(t)}$, where $w(t)$ is the number of improving steps at time t. Clearly, the convergence rate has an exponential form, but it depends on the frequency of winning trials. The scales at a given stage of the process tend to stabilize as long as they are productive of improving steps, and they rapidly decrease when the frequency of winning trials decreases. The choice of α depends on the properties of the problem to be solved. Unfortunately, the author does not know any theory which would enable this choice to be generally optimized. To date, the most usual method consists of empirically determining an α-function appropriate to the class of problems to be solved in a given application. This is not generally difficult, but the cost of the tuning procedure has to be added to the implementation cost.

3. Experimental Study of Performance

Box-Constrained Test Functions

Presented below is an experimental study of the performance of the Hyperbell Algorithm on some hard minimization problems. Most test functions used in the current literature have only a small number of local extrema, which severely limits the scope of the tests (see, however, Floudas & Pardalos, 1990). We were nevertheless able to find two families of standard hard functions for the test: Csendes functions (Csendes, 1985; see also Zhigljavsky, 1991, p. 16) and Griewank functions (Griewank, 1981; see also Rinnooy Kan & Timmer, 1987b, p. 76). A third family of hard functions, previously used by Courrieu (1993), was also selected. This family will be called the W functions. The three families of functions were used with 2 and 10 variables in the comparative study. They were used with 10 variables in the study of densities. Only the W family was used in the complexity study.

Csendes test functions

$$Cn(X) = \sum_{i=1}^{n} x_i^6 \left(2 + \sin\frac{1}{x_i}\right), \quad -1 \leqslant x_i \leqslant 1.$$

Contrary to what might appear, this function is defined at $X = 0$, precisely where it has its unique global minimum (0). The function possesses a countable infinity of local minima on the search domain, and the oscillation frequency tends towards infinity on the neighborhood of the global solution. This property makes it practically impossible to minimize by applying local searches. However, Csendes functions have the following feature: they oscillate between two convex "hulls" which approach each other on the neighborhood of the solution.

Note: what we refer to as a "hull" of a function is a hypersurface which joins all the extrema of the same type (maxima or minima) of the function, and which itself has the minimum number of extrema.

W functions

$$Wn, k(X) = \frac{1}{n}\sum_{i=1}^{n} 1 - \cos(kx_i)\exp(-x_i^2/2), \quad -\pi \leqslant x_i \leqslant \pi.$$

W functions have their unique global minimum (0) at $X = 0$. The number of local minima on the search domain is k^n (for k odd) or $(k+1)^n$ (for k even). Only $k = 10$ was used in the comparative study, which gave 121 local minima for $n = 2$, and more than 2.59×10^{10} local minima for $n = 10$. The function oscillates between two hulls of constant mean $(= 1)$ whose distance from each other is maximal on the neighborhood of the solution.

Griewank test functions

$$Gn(X) = 1 + \sum_{i=1}^{n} x_i^2/d - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}),$$

For $n = 2 : d = 200, -100 \leqslant x_i \leqslant 100$. For $n = 10 : d = 4000, -600 \leqslant x_i \leqslant 600$. These functions have their unique global minimum (0) at $X = 0$, and have a large number of local minima. They have many "trap" basins on a large area around the solution. However, in the neighborhood of certain points, Griewank functions have some special directional local properties. To get an idea of these properties, assign one of the variables a value like $x_i = (2k+1)\pi\sqrt{i}/2, k \in Z$. In this case, the partial derivatives for all other variables point to the solution, which is a stable attractor for the variables that reach its neighborhood.

Comparative Study for Box-Constrained Problems

Reference algorithms

Because the experimental results for problems like these are rare in the literature, three reference algorithms were used.

Simulated Annealing: We chose the Dekker and Aarts (1991) algorithm, which is a recent version of the Simulated Annealing algorithm adapted to numerical spaces. It is like a sequential Multistart algorithm in that it uses multiple directional local search steps, whose global distribution is governed by a simulated annealing rule. The algorithm was implemented in compliance with the indications provided by the authors, except that the stopping rule was simplified in an experimentally adapted way. This rule was replaced by a "temperature" thresholding, involving a complete local search starting from the best point found whenever the temperature fell below

10^{-8}. For the test functions used, this order of magnitude for the temperature was indeed found to correspond to a "freezing" point in the process, after which no progress was observed. We used the tuning values proposed by the authors (namely: $\chi_0 = 0.9, L_0 = 10, t = 0.75$), with $m_0 = 100$, except for the speed parameter that usually had to be reduced since a δ of 0.1 proved to be unsuitable to problems this difficult. To determine δ, a trial and error procedure was used to obtain 10 successive runs with no error in the result for each problem. This search succeeded for three out of six problems (W2, G2, G10), but it was impossible to obtain an exact result for the remaining problems, even once. The value used for δ, then, was simply the minimum value found in those problems which were solved (0.005).

Improving Hit-and-Run: we chose the algorithm proposed by Zabinsky *et al.* (1993) (see also Romeijn & Smith, 1994). It is a converging random walk algorithm which generates an asymptotically uniform distribution on the feasible region, and it was implemented as it is described by the authors (with $H = I$). No tuning is necessary for the Improving Hit-and-Run algorithm, and the stopping rule was fixed at 500000 evaluations of the function.

Distributed Search: this algorithm proposed by Courrieu (1993) was chosen because it uses Cauchy distributions like the Hyperbell Algorithm, but it is not a random walk. The Distributed Search uses a population of M points in the feasible region, each of them being the center of a Cauchy distribution. The convergence is obtained by estimating the most appropriate positions and scales of the M distributions at successive stages of the search. The speed of convergence is controlled by a parameter (α), and the algorithm has a DLS variant which is quite similar to that of the Hyperbell Algorithm. To determine M, α and the eventual necessity of applying the DLS variant, a trial and error procedure was used to obtain 10 successive runs with no error in the result for each problem.

Tuning the Hyperbell Algorithm

The Hyperbell Algorithm parameter α was also estimated by trial and error until we obtained 10 successive runs with no error in the result of any of the problems. This criterion obviously did not imply that the probability of error was equal to zero, which *a priori* is impossible in finite time. The results were considered exact (for all algorithms) when the process found a function minimum equal to 0, within the precision range of the compiler (TURBO-PASCAL 5.0 on a COMPAQ Deskpro 486 computer). The parameter ε was set at 10^{-20}. The DLS variant was applied only if the tuning procedure failed without it (this occured only for the function G10).

Results

The results are presented in Table I. For each algorithm and each problem, the table gives the values of the tuning parameters and the mean number of function calls

Table I. Tuning, and mean performance (with standard deviation) on 10 successive runs for the four algorithms and six test functions.

		TESTED ALGORITHMS			
		Dekker & Aarts (1991)	Zabinsky *et al*. (1993)	Courrieu (1993)	Hyperbell Algorithm
C2	tuning	δ = 0.005	–	M=100,α =1.00	α = 0.93
	f calls	46549 (13149)	287292 (100493)	7028 (949)	663 (21)
	f error	3.38E–15 (1.01E–14)	5.49E–30 (1.11E–29)	No error	No error
	X error	2.42E–03 (1.65E–03)	8.14E–06 (5.60E–06)		
C10	tuning	δ = 0.005	–	M=200,α =1.00	α =0.993
	f calls	430301 (41406)	465682 (31070)	89453 (2304)	6621 (88)
	f error	1.30E–10 (1.18E–10)	2.03E–23 (1.79E–23)	No error	No error
	X error	3.76E–02 (4.76E–03)	2.47E–04 (5.21E–05)		
W2	tuning	δ = 0.005	–	M=100,α =0.75	α =0.99
	f calls	29485 (11534)	363364 (108840)	4161 (371)	2313 (41)
	f error	No error	2.71E–08 (3.09E–08)	No error	No error
	X error		2.75E–05 (1.88E–05)		
W10	tuning	δ = 0.005	–	M=250,α =0.75	α =0.99978
	f calls	221752 (29660)	496511 (4753)	119799 (2617)	105723 (899)
	f error	0.210 (0.028)	0.372 (0.058)	No error	No error
	X error	2.927 (0.494)	4.187 (0.936)		
G2	tuning	δ = 0.005	–	M=150,α =0.80	α =0.995
	f calls	52793 (4741)	402978 (98100)	5712 (393)	4687 (93)
	f error	No error	6.38E–07 (6.92E–07)	No error with DLS	No error
	X error		1.21E–03 (6.27E–04)		
G10	tuning	δ = 0.01	–	M=300,α =0.60	α =0.995, DLS
	f calls	251870 (10208)	471011 (31388)	205584 (4084)	106799 (10583)
	f error	No error	0.161 (0.106)	No error with DLS	No error
	X error		23.98 (8.12)		

for the 10 runs (f calls). Each gradient calculation was counted as a calculation of the function. This is particularly justified in the case of separable functions where the calculation cost of the gradient is very close to the calculation cost of the function. Also given are the mean error on the function value (f error) and the mean Euclidean distance between the best point found and the global minimizer (X error). Standard deviations (with 9 degrees of freedom) are shown in parentheses to the right of the corresponding means.

Table II. Performance of the random walk using Gauss and logistic densities for three test functions (ten successive runs).

f		Gauss density	Logistic density
C10	tuning	α = 0.989	α = 0.99
	f calls	4305 (66)	4705 (58)
		No error	No error
W10	tuning	α = 0.9999	α = 0.9999
	f calls	250781 (10306)	247011 (11470)
	f error	0.293 (0.082)	0.268 (0.063)
	X error	3.492 (0.951)	3.397 (0.773)
G10	tuning	α = 0.99, DLS	α = 0.99, DLS
	f calls	59094 (1667)	58922 (2187)
		No error	No error

The results are quite clear. The Hyperbell Algorithm solved all the problems with the smallest number of function evaluations. The most difficult function for the algorithm was G10 whose optimization clearly required the DLS variant. Without the DLS variant, the process frequently became trapped by local minima. Note that the minimization of Griewank functions is relatively easy for the Dekker and Aarts (1991) algorithm; this is logical given its strong directional local component. However, note also that Rinnooy Kan and Timmer (1987b) reported some disappointing results concerning the minimization of the same Griewank functions by a "Multi Level Single Linkage" algorithm. The Zabinsky *et al.* (1993) algorithm found good approximations of the minimum for functions C2, C10, W2 and G2, but the convergence was slow. The Distributed Search method found the exact solution for all the problems, but as one can see in Table I, that algorithm converges much slower than the Hyperbell Algorithm.

Study of Densities: Using Other Bells

The functions C10, W10 (with $k = 10$) and G10 were used for testing the performance of the random walk when one replaces the Cauchy density with gradually scaled down Gauss and logistic densities. Approximations of Gaussian variables were generated using a sum of 12 independent uniform random variables. Each of the 10 standard deviations was initialized at $s_i(0) = (b_i - a_i)/3.664$, given that Prob$\{|z| < 3.664/2\} = 0.5^{1/10}$. Logistic random variables were generated using the formula $y_i = -s_i \,\mathrm{Ln}(1/u_i - 1) + x_i$, with u_i taken at random from a uniform distribution on (0,1), and $s_i(0) = (b_i - a_i)/2(-\mathrm{Ln}(2/(1 + 0.5^{1/n}) - 1)), n = 10$. The tuning procedure was similar to that employed for the Hyperbell Algorithm with Cauchy densities. Means and standard deviations of performance on ten suc-

Table III. Tuning of the Hyperbell Algorithm and mean number of function calls (with standard deviation) as a function of the number of variables (ten successive runs without error).

Number of variables	Tuning (α)	Function calls		Number of variables	Tuning (α)	Function calls	
3	0.9500	504	(18)	18	0.9949	5045	(205)
6	0.9810	1339	(66)	21	0.9958	6002	(146)
9	0.9870	1982	(73)	24	0.9965	7247	(262)
12	0.9910	2875	(113)	27	0.9971	8786	(350)
15	0.9938	4127	(133)	30	0.9973	9529	(376)

cessive runs are presented in Table II. As one can see, Gauss and logistic densities provided good performance on function C10, and also on function G10 with the DLS variant. However, it was impossible to solve the W10 problem, even once, using Gauss or logistic densities. Clearly, the use of exponential densities does not lead to algorithms as robust as the Hyperbell Algorithm with Cauchy densities. The behaviour of exponential densities looks like the behaviour of bounded support densities, they provide fast but uncertain convergence. In contrast, approximations of uniform densities (e.g. Improving Hit-and-Run) provide slow, truly guaranteed convergence. The use of Cauchy densities allows for a good compromise between speed and accuracy requirements.

Complexity Study

The behaviour of the Hyperbell Algorithm was studied varying two usual factors of complexity: the number of variables and the number of minima. The W test function family was used since these complexity factors are easy to control for these functions.

Effect of the number of variables: For this study, the W test function was used with $k = 0$ and a number of variables varying from 3 to 30 (step 3). This is a set of ten uniminimal functions. The tuning procedure of the Hyperbell Algorithm was similar to that employed in the preceding studies. Results are presented in Table III. The number of function calls (t) increased as a function of the number of variables (n). This function was close to linear. The best regression equation is $t = 343.4n^{1.3} - 922$, with a very high correlation coefficient $r = 0.997$. A simple linear regression gave $r = 0.994$. The tuning was very well predicted by the relation $\alpha = 0.2064(1 - n^{-1.3}) + 0.7934, r = 0.999$.

Effect of the number of minima: For this study, the W test function with $n = 1$ and 13 even values for k was used, giving a range of 1 to 101 minima. Results are presented in Table IV. As one can see, the number of function calls increased monotonically as a function of the number of minima (m). The best simple relation which was found is $t = 47.83m^{0.87} + 156.97, r = 0.968$. However, this type of

Table IV. Tuning of the Hyperbell Algorithm and mean number of function calls (with standard deviation) as a function of the number of minima (ten successive runs without error).

Number of minima	Tuning (α)	Function calls		Number of minima	Tuning (α)	Function calls	
1	0.820	134	(15)	15	0.9720	764	(64)
3	0.830	149	(17)	17	0.9725	823	(28)
5	0.860	178	(10)	19	0.9730	826	(16)
7	0.940	372	(30)	21	0.9736	867	(43)
9	0.965	629	(36)	51	0.9800	1173	(40)
11	0.969	672	(73)	101	0.9923	2956	(126)
13	0.971	758	(47)				

relation cannot be generalized without caution since we have seen that it is easy for the Hyperbell Algorithm to minimize Csendes functions, which have theoretically an infinity of minima. In fact, we have to suspect that the number of minima *per se* is not relevent, but that it is linked to more relevent caracteristics in the case of W functions.

Comparative Study for Nonlinearly Constrained Problems

The Hyperbell algorithm was initialy designed for searching global extrema of multiextremal functions defined on simple box-constrained feasible regions. However, certain recent global optimization algorithms are designed for solving optimization problems on feasible regions with relatively complex constraints (Zabinsky *et al.*, 1993; Romeijn & Smith, 1994). Hence, it appeared interesting to complete this experimental study with standard nonlinearly constrained problems. The two most difficult nonlinearly constrained problems used by Romeijn & Smith (1994) were selected (problem 1 and 2, pp. 119–120). These authors reported the best performance which was obtained with Hide-and-Seek type algorithms (including Improving Hit-and-Run) for solving these problems with a precision of 1%. For comparison, the same precision criterion (stopping rule) was used in the present study. Table V presents the mean number of constraint evaluations necessary for finding an initial point inside the feasible region (i.c.e), the mean number of function evaluations (f.e.), and the mean number of constraint evaluations (c.e.). These means were obtained with 20 successive runs, and the corresponding standard deviations are given in parentheses for the Hyperbell algorithm. Results concerning the Hide- and-Seek algorithm are taken from Romeijn & Smith (1994). As one can see, the Hide-and-Seek algorithm found an initial feasible point faster than the Hyperbell algorithm for problem 2. Hide-and-Seek algorithm has a special procedure for finding an initial feasible point. When the constraint function defines a subset with relatively low Lebesgue mesure, this procedure is more effective than

Table V. Mean performance (with standard deviations) for 20 successive runs on two nonlinearly constrained problems solved with 1% precision.

	Hyperbell	Romeijn & Smith (1994)
Problem 1	α = 0.992	
i.c.e	6.55 (6.39)	10.1
f.e.	329.75 (131.70)	530.6
c.e.	744.90 (276.10)	1876.3
Problem 2	α = 0.990	
i.c.e.	835.85 (770.82)	158.1
f.e.	168.95 (51.30)	281.9
c.e.	8323.45 (2730.87)	13893.1

the very simple initialization procedure of the Hyperbell algorithm. However, this was the only advantage of Hide-and-Seek in the present experiment, and globally, the Hyperbell algorithm always solved the problems faster.

4. Tuning Procedures for Applications

Tuning the algorithm for standard test problems is quite easy because the exact solution of these problems is known *a priori*. However, this is not the case, in general, for realistic applications. Moreover, certain applications require strict control of computational effort, and more or less precision (or reliability) for the result. Usually, the tuning is performed at the time of implementation ("off-line tuning"). In this case, one must determine an α value or an α-function (of complexity variables) that properly generalizes to the (infinite) set of problems which the application should be able to solve. However, in certain cases, it is not possible to find an appropriate α-function for the entire set of potential problems. Such a situation requires the use of an "on-line tuning" procedure. To date, we do not know of any general tuning method with low computational cost. In this section, we first examine the behavior of the algorithm as a function of its tuning. Based on the resulting observations, examples of simple tuning procedures are defined and experimental results are presented.

Test Problem

For this study, we selected a family of problems which has practical applications and whose exact solution is not known to date. This is the so called "elliptic Fekete points' problem (of order d)" (Pardalos, 1995; Shub & Smale, 1993):

$$\begin{array}{ll} \text{global max} & f_d(x) = \prod_{1 \leqslant i \leqslant j \leqslant d} \|x_i - x_j\|, \quad x \in \mathbb{R}^{3d}, \\ \text{subject to} & \|x_i\| = 1, i = 1, ..., d. \end{array}$$

Table VI. Behavior of the Hyperbell Algorithm (means and standard deviations for 10 runs) as a function of α for a 12 Fekete points' problem.

α	Maximum of f_{12} found	Solving time
0.8	1.78405E+8 (3.46344E+8)	392 (15)
0.9	4.62357E+8 (4.25513E+8)	810 (34)
0.95	1.55778E+9 (2.98085E+8)	1607 (44)
0.975	2.05888E+9 (1.12839E+8)	3166 (61)
0.9875	2.19609E+9 (1.00874E+8)	6192 (132)
0.99375	2.35477E+9 (6.96366E+7)	12275 (225)
0.996875	2.40710E+9 (2.66601E+7)	22209 (2480)
0.9984375	2.41785E+9 (0)	25802 (5062)
0.99921875	2.41785E+9 (0)	54298 (6732)
0.999609375	2.41785E+9 (0)	91316 (21832)
α	Frequency of winning trials	Time to stop
0.8	0.491 (0.020)	395 (15)
0.9	0.477 (0.021)	816 (32)
0.95	0.459 (0.014)	1619 (42)
0.975	0.446 (0.009)	3198 (50)
0.9875	0.430 (0.010)	6262 (108)
0.99375	0.425 (0.008)	12434 (179)
0.996875	0.370 (0.055)	22880 (1902)
0.9984375	0.031 (0.006)	29599 (192)
0.99921875	0.011 (0.001)	58010 (85)
0.999609375	0.004 (0.001)	115301 (81)

The constraint defines a set of zero mesure in $\mathbb{R}^3$ (the unit sphere surface). Hence we have to use a projection of $\mathbb{R}^3$ points on S^2:

$$x_i := z_i/\|z_i\|, z_i \in [-1,1]^3 \backslash 0.$$

In doing this, all points z_i generated along a given radial direction are equivalent, the number of independent variables theoretically reduces to $2d$, and one can easily verify that the convergence properties of the Hyperbell Algorithm are not modified.

Effects of Tuning

In order to illustrate a typical behavior of the Hyperbell Algorithm as a function of α , a 12 Fekete points' problem was approximately solved with 10 different values of α according to: $\alpha_{k+1} = (\alpha_k + 1)/2, \alpha_0 = 0.8, k = 0, ..., 9$. The stopping criterion was $s(t) \leqslant 1.10\varepsilon, \varepsilon = 10^{-20}$, and 10 runs were performed for each α value. Table VI reports the means (and standard deviations) of the maximum found for the objective function, the solving time (number of function calls for finding

the result), the frequency of winning trials at stopping criterion, and the time to stop (number of function calls for reaching the stopping criterion).

One can see in Table VI that the approximation of the function global maximum is improved as α increases, and that one obtains a stable result (with variance close to 0) after a critical value of α is reached. The computational cost monotonically increases, and the frequency of winning trials decreases as a function of α . Note also that the mean frequency of winning trials is lower than 1/2 for usual values of $\alpha(\geqslant 0.8)$. A stable result can be considered as the best solution which can be provided by the algorithm in the limit of a given computational effort. However, if no validity test of the result is available, one can only guess that this is a global optimum.

Cost Constrained Tuning

This is an on-line tuning method which enables one to choose the range of the computational cost allowed for the search. Choose a large integer T, a small real $\omega > \varepsilon$, and a stopping criterion of the form $s(t_\omega) \leqslant \omega$, with the (approximative) constraint that $t_\omega \leqslant T$. If the scales of the different variables are not equal, consider only the largest one, or eventually choose an ω value for each variable in such a way that $\omega_i = c(s_i(0) - \varepsilon) + \varepsilon, i = 1...n$, where c is a small positive constant. Clearly T is an approximation of the maximum number of function evaluations allowed for the search, with a precision criterion ω in Ω . After the algorithm statement, we have:

$$s(t_\omega) = (s(0) - \varepsilon)\alpha^{(1-w_\omega)t_\omega} + \varepsilon = \omega \Rightarrow$$
$$t_\omega = \frac{1}{(1 - w_\omega)\ln\alpha} \ln\left(\frac{\omega - \varepsilon}{s(0) - \varepsilon}\right),$$

where w_ω is the frequency of winning trials at stopping criterion time. w_ω is a random variable whose law depends on the problem and on the tuning, however one can estimate that $0 < w_\omega \leqslant 1/2$ in most cases. Assuming this, we obtain:

$$\ln\left(\frac{\omega - \varepsilon}{s(0) - \varepsilon}\right) / \ln\alpha < t_\omega \leqslant 2ln\left(\frac{\omega - \varepsilon}{s(0) - \varepsilon}\right) / \ln\alpha \simeq T,$$

then we take:

$$\alpha = \exp\left(2\ln\left(\frac{\omega - \varepsilon}{s(0) - \varepsilon}\right)/T\right) = \left(\frac{\omega - \varepsilon}{s(0) - \varepsilon}\right)^{2/T}.$$

If T is sufficiently large, one can obtain a stable result (or a global extremum), but this cannot be verified using only one run. If no validity test of the result is available, one can use two different costs (say T1 and T2) and perform the two corresponding runs. Then the global cost will be comprised between (T1+T2)/2 and T1+T2. In general, there is little chance of obtaining two similar results if these are not stable solutions.

Off-Line Tuning

Performing an off-line tuning, one must determine an α value or an α-function (of complexity variables) which will provide reliable results with minimal computational cost for a large class of problems. Given a particular problem, we first must find a minimal α value which provides stable results. Given the convergence properties illustrated in Table VI, this reduces to a one variable uniextremal optimization. Hence, variants of usual local search methods, such as bisection type procedures, can be used, and the convergence towards an optimal tuning can be guaranteed. Now, for the purpose of an application, it is not necessarily relevent to find an exact optimal tuning for a particular problem, and one has to take into account the computational cost of the tuning procedure itself. In particular, we have to take into account the fact that the computational cost increases quickly as α tends to 1, since this cost is approximately proportional to $|1/\ln\alpha|$ (see previous section). Hence, we must define a tuning procedure which avoids large overestimating of α and repeated runs using overestimated α values. As an example, the following procedure provides quite good results:

{***parameters:*** α_0, α_1: two initial low values of α;
N: critical number of repeated runs; }
{***first approximation***} $f_0 :=$ Hyperbell(f, α_0);
BEST := f_0; $f_1 :=$ Hyperbell(f, α_1);
if (f_1 *better than* BEST) **then** BEST := f_1;
$k := 1$;
while ($f_{k-1} \neq$ BEST) **or** ($f_k \neq$ BEST) **do**
begin
$k := k + 1$;
$\beta_k := \left(\dfrac{|f_{k-1} - f_{k-2}|}{|f_{k-1}| + |f_{k-2}|}\right)^{0.1}$;
$\alpha_k := (\alpha_{k-1} + \beta_k)/(1 + \beta_k)$;
$f_k :=$ Hyperbell(f, α_k);
if (f_k *better than* BEST) **then** BEST := f_k;
end;

{***final tuning***}
$k := k - 1$;
$r := 1$;
repeat
repeat
$f_k :=$ Hyperbell(f, α_k);
$r := r + 1$;
until $(r = N)$ **or** ($f_k \neq$ BEST);

if ($f_k \neq$ BEST) **then begin**
$r := 0$;
$k := k + 1$;
$\beta_k := \left(\dfrac{|f_{k-1} - \text{BEST}|}{|f_{k-1}| + |\text{BEST}|} \right)^{0.1}$;
$\alpha_k := (\alpha_{k-1} + \beta_k)/(1 + \beta_k)$;
if (f_{k-1} *better than* BEST) **then** BEST $:= f_{k-1}$;
end;
until ($r = N$);

Note that the differences ($\neq$) have to be tested with finite precision (e.g. 15 significant decimal digits). The expression of β_k can be modified. For example, using $\beta_k = 1$ provides an α sequence similar to the one reported in Table VI (with α_0 = 0.8 and α_1 = 0.9). The choice of N (critical number of repeated runs) depends on the desired reliability of the tuning. The "first approximation" part of the procedure can also be used as an on-line tuning procedure. This can be interesting because the procedure ends when a result stability criterion is satisfied, however the computational cost is not controlled for.

The tuning procedure was applied to elliptic Fekete points' problems of various orders. Main results concerning orders 10, 11 and 12 are reported in Table VII. Given the α-function (of the number of variables) previously obtained for uniextremal functions (see the "complexity study" section), it was assumed that $\alpha \geqslant \alpha_1 = 0.8 + 0.2(1 - 1/d)$. Tuning costs are total numbers of function evaluations. The first approximation tuning cost can be viewed as an on-line tuning cost with stability criterion. The estimated maximum of f was always obtained in the first approximation procedure with 15 significant digits. The final tuning procedure only minimized the solving time for the specified reliability criterion ($N = 10$).

The remaining problem concerns the generalization of tuning to a large class of problems. Depending on the application, the problems can differ only by a random set of data, or (also) by systematic complexity factors (e.g. number of variables). In the first case, a quite evident method consists of selecting a sample of problems and performing the tuning for each problem, which provides a sample of tunings. Then one can determine an upper boundary of α using a simple confidence interval method. However, the distribution of α, considered as a random variable, is very asymetrical. Hence, it is better to compute the confidence interval for the t_ω variable (by usual statistical methods), and then to compute the corresponding upper bound of α using the relations stated in the "cost constrained tuning" section. The second case is the most frequent in practice. Unfortunately, it is also the most problematic case because available empirical complexity factors are not always the relevent ones, or the α-function is not a simple monotonic function of these factors. Sometimes, it is easy to find an appropriate approximation of the α-function, as we did in the "complexity study" section. However, if one considers for example

Table VII. Results provided by the tuning procedure for Fekete points' problems of order 10, 11, and 12. $\alpha_0 = 0.8, \alpha_1 = 0.8 + 0.2(1 - 1/d), N = 10$.

	f_{10}	f_{11}	f_{12}
α	0.999609313189144	0.999654352379111	0.998184163562489
maximum of f_d	5.74088185070187E+6	9.99798997082430E+7	2.41785163922926E+9
mean solving time	90807	122889	21645
first approx. cost	712488	1051620	121779
total tuning cost	1798733	2591413	352330

the set of elliptic Fekete points' problems, the α-function of d seems hard to predict. In such a situation, a practical solution would probably be to implement the tuning values for the most frequent problems, and to implement an on-line tuning procedure for the remaining cases.

5. Conclusion

The Hyperbell Algorithm clearly exhibits high performance levels on difficult global optimization problems. The use of Cauchy distributions in this type of framework is visibly more appropriate than the use of more usual distributions. Complexity effects which were experimentally tested are close to linear, however not all the relevent complexity factors are identified. Simple tuning procedures are available, however verifying reliability always requires a non negligible amount of computational effort. Further research efforts are needed to develop an efficient way of determining the best algorithm tuning for each problem or application. This requires a theory of problem complexity which is not available to date. Despite these remaining questions, the Hyperbell algorithm is a useful tool in practice. It is very easy to implement and, clearly, it provides reliable results with fast convergence.

References

Baritompa, W. (1993), Customizing methods for global optimization – a geometric viewpoint. *Journal of Global Optimization* **3**, 193–212.

Boender, C.G.E., Rinnooy Kan, A.H.G., Stougie, L., Timmer, G.T. (1982), A stochastic method for global optimization, *Mathematical Programming* **22**, 125–140.

Breiman, L., Cutler, A. (1993), A deterministic algorithm for global optimization, *Mathematical Programming* **58**, 179–199.

Courrieu, P. (1993), A distributed search algorithm for global optimization on numerical spaces. *RAIRO: Recherche Opérationnelle / Operations Research,* **27(3)**, 281–292.

Csendes, T. (1985), A simple but hard-to-solve global optimization test problem, *IIASA Workshop on Global Optimization,* Sopron (Hungary).

Dekker, A., Aarts, E. (1991), Global optimization and simulated annealing, *Mathematical Programming* **50**, 367–393.

Floudas, C.A., Pardalos, P.M. (1990), *Collection of Test Problems for Constrained Global Optimization Algorithms,* Springer-Verlag, Lecture Notes in Computer Science 455.

Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts.
Griewank, A.O. (1981), Generalized descent for global optimization, *Journal of Optimization Techniques and Application* **34**, 11– 39.
Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.
Horst, R., Pardalos, P.M. (1995), *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht.
Ingber, L. (1989), Very fast simulated re-annealing, *Mathl. Comput. Modeling* **12(8)**, 967–973.
Ingber, L., Rosen, B. (1992), Genetic algorithms and very fast simulated reannealing: a comparison. *Mathl. Comput. Modelling* **16(11)**, 87–100.
Pardalos, P.M. (1995), An open global optimization problem on the unit sphere, *Journal of Global Optimization* **6**, 213.
Pintér, J. (1988), Branch-and-Bound methods for solving global optimization problems with Lipschitzian structure, *Optimization* **19(1)**, 101–110.
Rinnooy Kan, A.H.G., Timmer, G.T. (1987a), Stochastic global optimization methods. Part I: clustering methods, *Mathematical Programming* **39**, 27–56.
Rinnooy Kan, A.H.G., Timmer, G.T. (1987b), Stochastic global optimization methods. Part II: multi level methods, *Mathematical Programming* **39**, 57–78.
Romeijn, H.E., Smith, R.L. (1994), Simulated Annealing for constrained global optimization, *Journal of Global Optimization* **5**, 101–126.
Shub, M., Smale, S. (1993), Complexity of Bezout's theorem III. Condition number and packing, *Journal of Complexity* **9**, 4– 14.
Solis, F.J., Wets, R.J-B. (1981), Minimization by random search techniques, *Mathematics of Operations Research* **6(1)**, 19– 30.
Wood, G.R. (1992), The bisection method in higher dimensions, *Mathematical Programming* **55**, 319–337.
Zabinsky, Z.B., Smith, R.L., McDonald, J.F., Romeijn, H.E., Kaufman, D.E. (1993), Improving Hit-and-Run for global optimization. *Journal of Global Optimization* **3**, 171–192.
Zhigljavsky, A.A. (1991), *Theory of Global Random Search*, Kluwer Academic Publishers, Dordrecht.

Journal of Computational and Applied Mathematics 231 (2009) 39–48

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Fast solving of weighted pairing least-squares systems

Pierre Courrieu *

LPC, UMR 6146, CNRS-University of Provence, Marseille, France

ARTICLE INFO

Article history:
Received 1 September 2008

Keywords:
Weighted pairing least-squares
Generalized inverses
Generalized Cholesky factor

ABSTRACT

This paper presents a generalization of the "weighted least-squares" (WLS), named "weighted pairing least-squares" (WPLS), which uses a rectangular weight matrix and is suitable for data alignment problems. Two fast solving methods, suitable for solving full rank systems as well as rank deficient systems, are studied. Computational experiments clearly show that the best method, in terms of speed, accuracy, and numerical stability, is based on a special {1, 2, 3}-inverse, whose computation reduces to a very simple generalization of the usual "Cholesky factorization-backward substitution" method for solving linear systems.

1. Introduction

In this paper, we consider weighted least-squares problems in the following generalized form:

$$E(V; X, Y, W) = \sum_{i=1}^{m1} \sum_{j=1}^{m2} w_{ij} \left\| X_{i,:}V - Y_{j,:} \right\|^2,$$

$$C = \arg \min_{V \in R^{n1 \times n2}} E(V; X, Y, W), \tag{1}$$

where three matrices are given: $X \in R^{m1 \times n1}$, $Y \in R^{m2 \times n2}$, and $W = (w_{ij})$ is a rectangular $m1 \times m2$ "weighted pairing" matrix whose coefficients are non-negative real numbers. $X_{i,:}$ denotes the ith row of X, and $Y_{j,:}$ denotes the jth row of Y. Note that this generalization is not the same as the so-called "Generalized Least-Squares" [1]. In the special case where $m1 = m2 = m$ and W is a diagonal matrix, the above problem clearly reduces to an ordinary weighted least-squares problem, that is:

$$C = \arg \min_{V \in R^{n1 \times n2}} \sum_{i=1}^{m} w_{ii} \left\| X_{i,:}V - Y_{i,:} \right\|^2 = \arg \min_{V \in R^{n1 \times n2}} \left\| W^{1/2}XV - W^{1/2}Y \right\|^2. \tag{2}$$

In Section 2, we show that, in fact, every problem having the form (1) can be reduced to a problem having the form (2). In such problems, each equation of the least-squares system receives a specific weight that typically depends on some estimate of the reliability of the data used in that equation. The usual non-weighted case corresponds to $W = I$ (identity matrix). Ordinary weighted least-squares (2) are commonly used to solve regression problems with noisy data [2], and in "iteratively re-weighted least-squares" procedures for computing robust regression statistics such as M-estimators [3,4]. The generalization (1) is potentially relevant in "data alignment" problems, where there is no given one-to-one correspondence between X data points and Y data points (rows), but one has some non-negative "adequacy" or "plausibility" measure for each possible data pair, which is represented by W. Data alignment is a hard to solve problem commonly encountered in

* Corresponding address: LPC, CNRS - UMR 6146, Université de Provence, Centre St Charles, Bat. 9, Case D, 3 Place Victor Hugo, 13331 Marseille Cedex 3, France.
E-mail addresses: Pierre.Courrieu@univ-provence.fr, courrieu@up.univ-mrs.fr.

0377-0427/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2009.01.016

image processing and pattern recognition [5]. In this paper, $\|.\|$ denotes the Euclidean norm for vectors, and the Frobenius norm for matrices. As in Matlab, the notation "$a : b$" denotes an index interval of bounds "a" and "b", and if the bounds are not specified (:), this corresponds to the whole index range.

Whenever $W^{1/2}X$ is of full column rank in (2), the solution to this problem is unique and the normal equations lead to the well-known result:

$$C = (X'WX)^{-1}X'WY, \tag{3}$$

where X' denotes the transpose of X (or the conjugate transpose in the complex case).

One of the fastest ways of numerically obtaining the factor $(X'WX)^{-1}$ that appears in (3) consists of computing a Cholesky factorization LL' of the positive definite Gram matrix $X'WX$, then one inverts the upper triangular factor L' by a simple backward substitution method, and one obtains $(X'WX)^{-1} = L'^{-1}L^{-1}$. However, if $W^{1/2}X$ is not of full column rank, then the above method does not work because the matrix $X'WX$ is singular, and in this case, the weighted least-squares system is said to be rank deficient. The solution of rank deficient systems requires more robust methods, which are also slower than the above mentioned, in general. Among the fastest methods, we can consider those based on the use of suitable generalized inverses, such as the popular Moore–Penrose inverse [6–8]. A solution to (2) is then:

$$C = (W^{1/2}X)^{\dagger}W^{1/2}Y = (X'WX)^{\dagger}X'WY, \tag{4}$$

where the Moore–Penrose inverse is known to provide the least-squares solution C whose each column has the minimum Euclidean norm [6, p. 109].

However, one must note that the solution of least-squares problems does not specifically require the use of the Moore–Penrose inverse, and that other types of generalized inverses, such as {1, 3}-inverses whose numerical computation is possibly faster, can as well be used. According to ([6], pp. 104–105), one has always a solution to (2) with:

$$C = (W^{1/2}X)^{(1,3)}W^{1/2}Y, \tag{5}$$

where $A^{(1,3)}$ denotes any {1, 3}-inverse of the matrix A (see Section 3.2).

In fact, the problem of the computational cost is crucial in many practical applications, where one must repeatedly solve large least-squares systems. On the other hand, most practical problems lead to full rank systems that could be solved fast using (3), however, rank deficient systems can occasionally appear, and it is commonly not acceptable to obtain a "fatal error" diagnostic at run time. Thus, in order to optimize the performance of applications, we present in Section 3.2 a quite fast solution of type (4), and in Section 3.3 a solution of type (5) whose computational cost is similar to that of (3), which has the advantage of being fast while providing a suitable least-squares solution in all cases, even if the system is rank deficient. These solutions apply to problem (2) and to problem (1) as well.

2. The weighted pairing least-squares problem

In this section, we consider the generalization of the weighted least-squares (WLS) problem stated in (1), which we refer to as the "weighted pairing least-squares" (WPLS) problem.

Theorem 1. *Every WPLS problem of type (1) reduces to a WLS problem of type (2) since:*

$$\arg\min_{V\in R^{n1\times n2}} \sum_{i=1}^{m1}\sum_{j=1}^{m2} w_{ij}\left\|X_{i,:}V - Y_{j,:}\right\|^2 = \arg\min_{V\in R^{n1\times n2}} \sum_{i=1}^{m1} h_{ii}\left\|X_{i,:}V - Z_{i,:}\right\|^2,$$

where:
$H = (h_{ij})$ *is a diagonal matrix with diagonal coefficients* $h_{ii} = \sum_{j=1}^{m2} w_{ij},\ 1 \le i \le m1$,
$Z = H^{\dagger}WY$,
$H^{\dagger} = (h_{ij}^{\dagger})$ *is the Moore–Penrose inverse of* H, *with* $h_{ii}^{\dagger} = 1/h_{ii}$ *if* $h_{ii} > 0$, *and* $h_{ij}^{\dagger} = 0$ *if* $h_{ij} = 0$.

Proof. Set

$$d_{ik} = \sum_{j=1}^{m2} w_{ij}y_{jk}^2 - h_{ii}^{\dagger}\left(\sum_{j=1}^{m2} w_{ij}y_{jk}\right)^2, \quad 1 \le i \le m1,\ 1 \le k \le n2. \tag{6}$$

Then one has:

$$\begin{aligned}\sum_{i=1}^{m1}\left(h_{ii}\left\|X_{i,:}V - Z_{i,:}\right\|^2 + \sum_{k=1}^{n2} d_{ik}\right) &= \sum_{i=1}^{m1}\sum_{k=1}^{n2} h_{ii}\left(X_{i,:}V_{:,k} - h_{ii}^{\dagger}\sum_{j=1}^{m2} w_{ij}y_{jk}\right)^2 + d_{ik}\\ &= \sum_{i=1}^{m1}\sum_{k=1}^{n2}\left(h_{ii}^{1/2}X_{i,:}V_{:,k} - h_{ii}^{1/2}h_{ii}^{\dagger}\sum_{j=1}^{m2} w_{ij}y_{jk}\right)^2 + d_{ik}\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{m1}\sum_{k=1}^{n2} h_{ii}(X_{i,:}V_{:,k})^2 - 2h_{ii}h_{ii}^{\dagger}(X_{i,:}V_{:,k})\sum_{j=1}^{m2} w_{ij}y_{jk} + h_{ii}h_{ii}^{\dagger 2}\left(\sum_{j=1}^{m2} w_{ij}y_{jk}\right)^2 + d_{ik}\\
&= \sum_{i=1}^{m1}\sum_{k=1}^{n2} h_{ii}(X_{i,:}V_{:,k})^2 - 2(X_{i,:}V_{:,k})\sum_{j=1}^{m2} w_{ij}y_{jk} + h_{ii}^{\dagger}\left(\sum_{j=1}^{m2} w_{ij}y_{jk}\right)^2 + d_{ik}\\
&= \sum_{i=1}^{m1}\sum_{k=1}^{n2}\left(\sum_{j=1}^{m2} w_{ij}(X_{i,:}V_{:,k})^2\right) - 2\left(\sum_{j=1}^{m2} w_{ij}y_{jk}(X_{i,:}V_{:,k})\right) + \left(\sum_{j=1}^{m2} w_{ij}y_{jk}^2\right)\\
&= \sum_{i=1}^{m1}\sum_{k=1}^{n2}\sum_{j=1}^{m2} w_{ij}((X_{i,:}V_{:,k})^2 - 2(X_{i,:}V_{:,k})y_{jk} + y_{jk}^2)\\
&= \sum_{i=1}^{m1}\sum_{j=1}^{m2} w_{ij}\left\|X_{i,:}V - Y_{j,:}\right\|^2.
\end{aligned}
$$

Noting that the additional terms (d_{ik}) given by (6) are independent of V, one obtains Theorem 1. □

Corollary 1. (i) *If $H^{1/2}X$ is of full column rank, then (1) has the unique solution:*

$$C = (X'HX)^{-1}X'HZ = (X'HX)^{-1}X'WY.$$

(ii) *No matter $H^{1/2}X$ is not of full column rank, (1) has the minimum norm solution:*

$$C = (H^{1/2}X)^{\dagger}H^{1/2}Z = (X'HX)^{\dagger}X'WY.$$

(iii) *No matter $H^{1/2}X$ is not of full column rank, (1) has all solutions of the form:*

$$C = (H^{1/2}X)^{(1,3)}H^{1/2}Z,$$

where H and Z are defined as in Theorem 1.

Proof. This directly follows from Theorem 1 and Eq. (3) for (i), Eq. (4) for (ii), and Eq. (5) for (iii). □

3. Fast solutions based on generalized inverses

3.1. Generalized Cholesky factors

Several generalizations of the Cholesky factorization can be found in the literature. A well-known generalized Cholesky factorization for solving the so-called "augmented linear systems" is available in [9] and [10]. Another type of generalization of the Cholesky factorization has been proposed in [11], and this approach has been successfully used to define a fast numerical method for computing the Moore–Penrose inverse [7]. The fundamental result for the generalized Cholesky factorization is:

Theorem 2 (*From [11]*). *Let G be a symmetric positive semi-definite matrix of order $n \times n$. Then there is an upper triangular matrix R such that $R'R = G$, $r_{ii} \geq 0$, $1 \leq i \leq n$, and if for an index i one has $r_{ii} = 0$, then $r_{ij} = 0$, $1 \leq j \leq n$. Moreover, the matrix R with these properties is unique.*

Proof. A proof of this is available in ([11], Theorem 4). □

The corresponding algorithm for computing the generalized Cholesky factor R defined in Theorem 2 is a very simple variant of the usual Cholesky factorization algorithm, and its computational complexity is the same. However, the generalization has the advantage of providing a suitable factor in all cases, even if the matrix G is singular.

Algorithm 1 (*Generalized Cholesky Factor R of the given matrix G*).

$r_{ij} \leftarrow 0, \quad 1 \leq i, j \leq n$ {*initialization of R*}
$r_{11} = \sqrt{g_{11}}$
for $j \leftarrow 2$ to n
  for $i \leftarrow 1$ to j
    if $i = j$ then $r_{ii} \leftarrow \sqrt{g_{ii} - \sum_{k=1}^{i-1} r_{ki}^2}$
      else if $r_{ii} > 0$ then $r_{ij} \leftarrow (g_{ij} - \sum_{k=1}^{i-1} r_{ki}r_{kj})/r_{ii}$
        {*else $r_{ij} = 0$ as a result of the initialization*}.

By construction, the output of Algorithm 1 is an upper triangular factor R with r non-zero rows, and $n - r$ zero rows, where r is the rank of G. The algorithm complexity is in $O(n^3)$, but the exact operations count depends on r and the indices of zero rows. When $r = n$, this count is maximum, and it is equal to that of the classical Cholesky factorization (plus $n(n-1)/2$ low cost tests).

3.2. Fast Moore–Penrose inverse based solution

Using Algorithm 1, one can define a fast method for computing the Moore–Penrose inverse of every finite matrix. Before examining this method, we rapidly recall some definitions and notations concerning generalized inverses.

Every finite matrix A has (possibly an infinite number of) generalized inverses (hereafter denoted B) that satisfy one or several of the following four Penrose equations:

$$\begin{array}{ll} ABA = A & (P1) \\ BAB = B & (P2) \\ (AB)' = AB & (P3) \\ (BA)' = BA & (P4). \end{array}$$

Every matrix B that satisfies the equation set $\{Pi, Pj, \ldots\}$ is said to be a $\{i, j, \ldots\}$-inverse of A, and it is usually denoted $A^{(i,j,\ldots)}$. The Moore–Penrose inverse of A is the unique matrix $A^\dagger = A^{(1,2,3,4)}$. For a complete explanation, the reader can see [6].

There are several methods for computing the Moore–Penrose inverse, the most usual being based on the singular value decomposition (SVD). This method is numerically very stable, however it is computationally heavy and hardly usable in many practical applications. Another usual method is based on Gram–Schmidt orthonormalization, which is clearly faster than SVD. However, the classical Gram–Schmidt orthonormalization (CGS or GSO) is known to be numerically instable. A simple and effective remediation to this drawback has been proposed in the form of a re-orthogonalization additional step, leading to the CGS2 method [12]. However, the additional step in CGS2 slows down the process, while it has been observed that CGS is not the fastest method for computing the Moore–Penrose inverse [7]. In fact, it turned out that among the most usual methods, including Greville's method, SVD, CGS/GSO, and iterative methods, the fastest known numerical method for computing the Moore–Penrose inverse is based on Algorithm 1 and on the following result [7]:

Theorem 3 (*From [7]*). *Let A be an $m \times n$ matrix, with $m \geq n$, set $G = A'A$, compute the generalized Cholesky factorization $G = R'R$ using Algorithm 1, remove all zero rows from R, which results in a full row rank matrix S of size $r \times n$, with $r \leq n$, and such that $S'S = G$. Then:*

$$A^\dagger = S'(SS')^{-1}(SS')^{-1}SA'.$$

Proof. The proof is available in [7]. Since it is short, we provide it hereafter.
We start with Eq. (3.2) from [8], that is:

$$(EF)^\dagger = F'(E'EFF')^\dagger E'. \tag{7}$$

Setting $E = A$, and $F = I$ in (7), one obtains:

$$A^\dagger = (A'A)^\dagger A' = G^\dagger A'.$$

Setting $E = S'$, and $F = S$ in (7), one obtains:

$$G^\dagger = (S'S)^\dagger = S'(SS'SS')^\dagger S = S'(SS')^{-1}(SS')^{-1}S, \tag{8}$$

since SS' is invertible because S is of full row rank. □

If A is an $m \times n$ matrix, with $m < n$, it suffices to use the relation $A^\dagger = ((A')^\dagger)'$. Note also that (8) provides a simple formula for the Moore–Penrose inverse of any symmetric positive semi-definite matrix, and that if S is of full rank $r = n$, then $G^\dagger = G^{-1}$.

Corollary 2. *Set $A = H^{1/2}X$ in Theorem 3, then the minimum norm solution of (1) is:*

$$C = S'(SS')^{-1}(SS')^{-1}SX'WY,$$

where S is defined as in Theorem 3.

Proof. This immediately follows from Theorem 3 and Corollary 1 (ii). □

3.3. Fast {1, 2, 3}-inverse based solution

Although Corollary 2 provides a fast solution to (1), this is not necessarily the fastest way of solving this problem. Moreover, observing Eq. (8), one can suspect a potential numerical instability whenever the matrix SS' is ill-conditioned, worsened by the fact that the factor $(SS')^{-1}$ is repeated. In this section, we describe a {1, 2, 3}-inverse based solution to (1) whose computational complexity is similar to that of (3), using Algorithm 1 and a simple variant of the backward substitution method for inverting upper triangular matrices. The simplicity of this solution allows us to expect not only faster computation, but also better numerical stability than the Moore–Penrose inverse based approach. We first define the generalization of the backward substitution method for computing generalized inverses of generalized Cholesky factors as

they are defined in Theorem 2. The computational complexity of the generalized algorithm is the same as that of the original backward substitution method ($O(n^3)$). The algorithm is designed to solve in U the following equation:

$$RU = I_R, \tag{9}$$

where I_R is a diagonal $n \times n$ matrix whose ith diagonal coefficient is equal to 0 if the ith row of R is zero, else this diagonal coefficient is equal to 1. The algorithm to solve (9) is then:

Algorithm 2 (*{1, 2, 3}-inverse U of the given generalized Cholesky factor R*).

$u_{ij} \leftarrow 0, \quad 1 \le i, j \le n$ {initialization of U}
for $j \leftarrow n$ downto 1
  if $r_{jj} \neq 0$ then {note: this test is optional}
    for $i \leftarrow j$ downto 1
      if $r_{ii} \neq 0$ then
        if $i = j$ then $u_{ii} \leftarrow 1/r_{ii}$
          else $u_{ij} \leftarrow -(\sum_{k=i+1}^{j} r_{ik}u_{kj})/r_{ii}$.

The test at the third line of Algorithm 2 is optional because it has no influence on the result. However, including this test allows to save a number of useless floating-point operations (whose result is zero) whenever R is singular.

By construction, the output of Algorithm 2 is an upper triangular matrix U that has the property that if the ith row of R is zero, then both the ith row and the ith column of U are zero. Note that the rank of U is equal to the rank of R, which is itself equal to the rank of the factorized matrix G and to the trace of I_R. Note also that if R is not invertible (in the usual sense), then $UR \neq I_R$, however, UR is always idempotent since $URUR = UI_RR = UR$.

Theorem 4. *Let R be a generalized Cholesky factor as defined in Theorem 2, let U be the corresponding output of Algorithm 2, then U is a {1, 2, 3}-inverse of R.*

Proof. U is a {1}-inverse of R since $RUR = I_RR = R$,
U is a {3}-inverse of R since $RU = (RU)' = I_R$,
U is a {2}-inverse of R since U is a {1}-inverse of R and has the same rank as R, then the conclusion follows from ([6], p. 46). Alternatively, one can easily verify that $URU = UI_R = U$. □

Theorem 5. *Let A be an $m \times n$ matrix, with $m \ge n$, set $G = A'A$, compute the generalized Cholesky factorization $G = R'R$ using Algorithm 1, compute $U = R^{(1,2,3)}$ using Algorithm 2. Then:*

(i) *The equation $A = QR$ has a solution in Q such that $Q'Q = I_R$. This solution is $Q = AU$.*
(ii) *The matrix $B = UU'A'$ is a {1, 2, 3}-inverse of A.*

Proof.

$$Q'Q = U'A'AU = U'R'RU = (RU)'RU = I_RI_R = I_R,$$
$$AU = QRU = QI_R = Q,$$

which proves (i).

Since (i) implies that $B = UQ'$, one has:
B is a {1}-inverse of A since $ABA = QRUQ'QR = QI_RI_RR = QR = A$,
B is a {2}-inverse of A since $BAB = UQ'QRUQ' = UI_RI_RQ' = UQ' = B$,
B is a {3}-inverse of A since $AB = QRUQ' = QI_RQ' = (AB)'$,
which proves (ii), and then completes the proof of Theorem 5. □

Corollary 3. *Set $A = H^{1/2}X$ in Theorem 5, then a solution to (1) is given by:*

$$C = UU'X'HZ = UU'X'WY,$$

where U is defined as in Theorem 5.

Proof. This immediately follows from Theorem 5 (ii) and Corollary 1 (iii). □

One can note that if $H^{1/2}X$ is of full column rank, then $U = R^{-1}$, and the solution provided by Corollary 3 is equal to that of Corollary 1 (i). Moreover, if $H^{1/2}X$ is of column rank $r \le n1$, then each column of the solution C to problem (1) provided by Corollary 3 has at most r non-zero coefficients, since the first factor (U) of the solution has $n1 - r$ zero rows. Note, however, that one can find particular examples showing that the above solution is not always the one having the minimum number of non-zero coefficients.

4. Computational test

4.1. Implementation of methods

The methods defined in Corollaries 2 and 3 for solving (1) have been implemented in Matlab code (version 7.5), and are listed in Appendix. The Matlab function corresponding to Corollary 2 is named "WPLSdagger", and the Matlab function corresponding to Corollary 3 is named "WPLS123". This makes available various implementation details that are not specified in the formal definition of algorithms, such as the way of testing the equivalence to zero of floating-point diagonal coefficients, or the way of avoiding an a posteriori removing of zero rows in Corollary 2 solution. In order to make the performance of the two tested methods comparable, we avoided the use of high level Matlab operators such as "inv()", "chol()", or "\", whose implementation is hidden and compiled.

4.2. Test problems

In order to test the performance of methods for solving (1) in terms of speed, accuracy, and numerical stability, we must build test problems in a way that allows strict control of relevant characteristics such as the size and the rank of the equation system, the exact weighted least-squares residue norm, and the ratio of extreme non-zero singular values of the system matrix (which can be seen as a kind of generalized condition number). Note that the solution (C) itself is not relevant for comparisons, since it is not unique in the case of rank deficient systems. Building coherent test problems having all required properties is not so easy, and we propose the following method.

First, one chooses the size parameters $m1, n1, m2, n2$, the rank parameter $r \leq n1$, and the maximum ratio, denoted κ_r, of non-zero eigenvalues of the Gram matrix $X'HX$ to be built. For practical reasons, one must choose the size parameters such that $m2 \geq m1 > n1$. Then one builds two orthogonal Householder matrices:

$$M = I - 2\frac{uu'}{u'u}, \quad \text{with a random column vector } u \in R^{m1},$$

$$N = I - 2\frac{vv'}{v'v}, \quad \text{with a random column vector } v \in R^{n1},$$

where the identity matrices (I) have the appropriate size in each case. One also builds a $r \times r$ diagonal matrix D, whose ith diagonal coefficient is equal to $\kappa_r^{(r-i)/2(r-1)}$, $1 \leq i \leq r$. Then one selects the first r columns of M and the first r rows of N, and one builds the matrix:

$$A = M_{:,1:r} D N_{1:r,:}.$$

The $m1 \times n1$ matrix A is of rank r, the greatest eigenvalue of $A'A$ is equal to κ_r, and the lowest non-zero eigenvalue of $A'A$ is equal to 1. Thus, we can set $X'HX = A'A$, that is $H^{1/2}X = A$.

For the next step, one builds a random $(m1 - r) \times n2$ real matrix F, and one sets:

$$P = M_{:,(r+1):m1} F,$$

where we note that the columns of P are orthogonal to those of A.

One can now build a suitable diagonal matrix $H = (h_{ii})$, $1 \leq i \leq m1$, by taking:

$$h_{ii} = \max\left(\left|\sum_{j=1}^{n1} a_{ij}\right|, \left|\sum_{j=1}^{n2} p_{ij}\right|\right)^2,$$

which guarantees that both A and P can be factorized with $H^{1/2}$ as the first factor, in order to build a coherent problem, and one obtains the first matrix of problem (1) by:

$$X = (H^{1/2})^{\dagger} A.$$

For the next step, one builds a random $n1 \times n2$ real matrix V, and one sets:

$$HZ = H^{1/2}(AV + P).$$

It remains to build a $m1 \times m2$ matrix W, with non-negative coefficients, such that $\sum_{j=1}^{m2} w_{ij} = h_{ii}$, $1 \leq i \leq m1$, and such that there is an $m2 \times n2$ matrix Y that is solution of the equation $WY = HZ$. Unfortunately, there is no available deterministic method for factorizing HZ in a suitable way, thus we must use a random "trials and errors" approach, as follows. Repeat the following four steps until $WY = HZ$ (at the working precision):

- build a $m1 \times m2$ matrix $T = (t_{ij})$ with non-negative random coefficients,
- compute the diagonal matrix K with $k_{ii} = \sum_{j=1}^{m2} t_{ij}$, $1 \leq i \leq m1$,
- set $W = HK^{-1}T$,
- set $Y = W^{\dagger}HZ$.

Table 1
Mean solving time (in milliseconds) of WPLS problems by the two methods.

$n1$	128			256			512		
κ_r	16	256	4096	16	256	4096	16	256	4096
Full rank									
WPLSdagger	126	126	127	608	610	607	3438	3438	3434
WPLS123	109	109	109	523	523	523	2868	2869	2869
Rank deficient									
WPLSdagger	103	102	102	502	497	498	2805	2803	2803
WPLS123	89	89	89	433	434	433	2412	2412	2468

Table 2
Mean accuracy of the two methods in solving WPLS problems.

$n1$	128			256			512		
κ_r	16	256	4096	16	256	4096	16	256	4096
Full rank									
WPLSdagger	8.9E−6	9.9E−3	1.532	1.4E−6	1.3E−3	0.436	0.2E−6	0.2E−3	0.099
WPLS123	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$
Rank deficient									
WPLSdagger	3.6E−6	11.9E−3	2.872	0.9E−6	2.1E−3	0.570	0.1E−6	0.3E−3	0.137
WPLS123	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$

The Moore–Penrose inverse $W^\dagger$ can be obtained using an accurate (slow) SVD method. In general, one obtains a suitable solution quite rapidly when $m2 > m1$, and κ_r is not too large. However, one can observe that the above process frequently fails for large systems if $\kappa_r > 2^{12}$, which seems to be a practical limit for generating problems in common computational environments such as Matlab.

We have now suitable matrices X, Y, and W for problem (1), and it remains to compute the exact weighted sum of quadratic residues (i.e. the minimized E function of (1)) corresponding to this problem as a reference value for testing the accuracy of solving methods. In order to do this, we use the fact that the columns of the matrix P are orthogonal to those of A, and the proof of Theorem 1. Then one obtains:

$$E_{\text{exact}} = \|P\|^2 + \sum_{i=1}^{m1}\sum_{k=1}^{n2} d_{ik},$$

where the additional terms (d_{ik}) are defined as in (6).

4.3. Results

Using the procedure described in Section 4.2, we generated various problems of type (1) with the parameter sets $n1 = \{128, 256, 512\}$, $\kappa_r = \{16, 256, 4096\}$, $r = \left\{n1, \frac{7}{8}n1\right\}$ (corresponding to "full rank" and "rank deficient" systems, respectively), while $m1 = 2\,n1$, $m2 = 2\,m1$, $n2 = 32$. Using all parameter combinations, one obtained 18 types of problems, and 10 problems of each type were randomly generated. Each problem was solved by both the WPLSdagger function (fast Moore–Penrose inverse based solution), and the WPLS123 function (fast {1, 2, 3}-inverse based solution). In each case, the solving time was recorded in milliseconds (in Matlab 7.5, on a MacBook computer), and the accuracy of each method was measured by $(E_{\text{method}} - E_{\text{exact}})/E_{\text{exact}}$. The mean solving times are reported in Table 1, and the mean accuracy values are reported in Table 2. All differences between the two methods are statistically significant ($p < .01$) using the sign test.

As one can see in Table 1, the {1, 2, 3}-inverse based method is faster than the fast Moore–Penrose inverse based method, in all cases. One can also note that rank deficient systems are solved faster than full rank systems of the same size by both methods, which is a consequence of the zeroing of certain rows in Algorithm 1. Moreover, an inspection of Table 2 clearly shows that the {1, 2, 3}-inverse based method is accurate in all cases, while the fast Moore–Penrose inverse based method is less accurate and highly sensitive to the parameter κ_r, thus numerically instable. In summary, it seems that the {1, 2, 3}-inverse based method is globally preferable to other known methods suitable for solving problem (1), except if, for some particular reason, one requires a minimum norm solution. However, in this last case, it is certainly preferable to use an accurate and numerically stable method for computing the Moore–Penrose inverse, but the price to be paid for this is, in general, a quite long computation time.

5. Conclusion

We have first generalized "weighted least-squares" (WLS) to "weighted pairing least-squares" (WPLS) problems. This generalization, which allows using a rectangular weight matrix, includes, as particular cases, the classical weighted and non-weighted least-squares problems, and it is more particularly suitable in the framework of data alignment problems.

We have shown that WPLS problems can always be reduced to problems having the same form as WLS problems, and we have studied two fast methods for solving such problems in the case of rank deficient systems as well as of full rank systems. Numerical experiments clearly showed that the best solving method, in terms of speed, accuracy, and numerical stability, is based on a special {1, 2, 3}-inverse whose computation is very simple. In contrast, approaches based on the Moore–Penrose inverse lead to slow computation, or alternatively to numerical instability.

Appendix

The following codes are provided for example, and for academic use only. The code is not optimized and exception cases are not managed.

Matlab code (version 7.5) of the WPLSdagger function:

```
function [C,Emethod,time] = WPLSdagger(X,Y,W)
% Moore--Penrose inverse based solution of a WPLS problem
tic % start the clock
[m1,n1]=size(X); [m2,n2]=size(Y); H=sum(W,2);g=X'*((H*ones(1,n1)).*X);
% s = full row rank generalized Cholesky factor of g
tol=n1*eps(norm(g,inf));
s=zeros(n1,n1); ii=0;
for i=1:n1
    ii=ii+1;
    v=g(i,i:n1)-s(1:(ii-1),i)'*s(1:(ii-1),i:n1);
    if v(1)>tol
        s(ii,i)=sqrt(v(1));
         if  i<n1
              s(ii,(i+1):n1)=v(2:end)/s(ii,i);
           end
          else ii=ii-1; end
end
rs=ii; s=s(1:rs,:);
% r = classical upper Cholesky factor of ss'
g=s*s';
r=zeros(rs,rs);
for i=1:rs
    v=g(i,i:rs)-r(1:(i-1),i)'*r(1:(i-1),i:rs);
    r(i,i)=sqrt(v(1));
    if i<rs
       r(i,(i+1):rs)=v(2:end)/r(i,i);
    end
end

% u = classical inverse of r
u=zeros(rs,rs);

for j=rs:-1:1
   for i=j:-1:1
       if i==j
            u(i,i)=1/r(i,i);
       else
           u(i,j)=-r(i,(i+1):j)*u((i+1):j,j)/r(i,i);
       end
    end
end
% iss = inverse of ss'
iss=u'*u;
% solution
C=s'*iss*iss*s*X'*W*Y;
time=toc; % record the solving time
% compute the weighted sum of quadratic residues
XC=X*C;
Emethod=0;
```

```
for i=1:m1
    for j=1:m2
        Emethod=Emethod+W(i,j)*sum((XC(i,:)-Y(j,:)).^ 2,2);
    end
end
end
```

Matlab code (version 7.5) of the WPLS123 function:

```
function [C,Emethod,time] = WPLS123(X,Y,W)
% {1,2,3}-inverse based solution of a WPLS problem
tic       % start the clock
[m1,n1]=size(X);  [m2,n2]=size(Y); H=sum(W,2); g=X'*((H*ones(1,n1)).*X);
% r = generalized Cholesky factor of g
tol=n1*eps(norm(g,inf));
r=zeros(n1,n1);
for i=1:n1
    v=g(i,i:n1)-r(1:(i-1),i)'*r(1:(i-1),i:n1);
    if v(1)>tol
        r(i,i)=sqrt(v(1));
        if i<n1
            r(i,(i+1):n1)=v(2:end)/r(i,i);
        end
    end
end
% u = {1,2,3}-inverse of r
u=zeros(n1,n1);
for j=n1:-1:1
    if r(j,j)~=0
        for i=j:-1:1
            if r(i,i)~=0
               if i==j
                   u(i,i)=1/r(i,i);
               else
                   u(i,j)=-r(i,(i+1):j)*u((i+1):j,j)/r(i,i);
               end
            end
        end
    end
end
% solution
C=u*u'*X'*W*Y;
time=toc; % record the solving time
% compute the weighted sum of quadratic residues
XC=X*C;
Emethod=0;
for i=1:m1
    for j=1:m2
        Emethod=Emethod+W(i,j)*sum((XC(i,:)-Y(j,:)).^ 2,2);
        end
    end
end
```

References

[1] J.Y. Yuan, Numerical methods for generalized least squares problems, Journal of Computational and Applied Mathematics 66 (1996) 571–584.
[2] T. Zhou, D. Han, A weighted least squares method for scattered data fitting, Journal of Computational and Applied Mathematics 217 (2008) 56–63.
[3] K.P. Bube, R.T. Langan, Hybrid l^1/l^2 minimization with applications to tomography, Geophysics 62 (4) (1997) 1183–1195.
[4] R.A. Maronna, R.D. Martin, V.J. Yohai, Robust Statistics: Theory and Methods, John Wiley & Sons Ltd, New York, 2006.
[5] S.-H. Lai, Robust image matching under partial occlusion and spatially varying illumination change, Computer Vision and Image Understanding 78 (2000) 84–98.
[6] A. Ben-Israel, T.N.E. Greville, Generalized Inverses: Theory and Applications, 2nd ed., Springer-Verlag, New York, 2003.
[7] P. Courrieu, Fast computation of Moore–Penrose inverse matrices, Neural Information Processing – Letters and Reviews 8 (2) (2005) 25–29.
[8] M.A. Rakha, On the Moore–Penrose generalized inverse matrix, Applied Mathematics and Computation 158 (2004) 185–200.

[9] W. Wang, J. Zhao, Perturbation analysis for the generalized Cholesky factorization, Applied Mathematics and Computation 147 (2004) 601–606.
[10] J. Zhao, The generalized Cholesky factorization method for saddle point problems, Applied Mathematics and Computation 92 (1998) 49–58.
[11] P. Courrieu, Straight monotonic embedding of data sets in Euclidean spaces, Neural Networks 15 (2002) 1185–1196.
[12] L. Giraud, J. Langou, M. Rozloznik, J. van den Eshof, Rounding error analysis of the classical Gram–Schmidt orthogonalization process, Numerische Mathematik 101 (2005) 87–100.

II.F Méthodes de validation de modèles et bases de données comportementales

L'expérimentation classique utilise des plans factoriels pour tester l'existence d'effets différenciant des conditions expérimentales, suivant des hypothèses de travail engendrées par des théories plus ou moins précises selon les cas. Le développement des modèles de simulation numériques conduit à des prédictions souvent beaucoup plus précises qui portent sur des niveaux d'analyse fins, que l'on a coutume d'appeler "items". La définition académique du mot "item" est "élément d'un ensemble organisé" (Petit Larousse), ce qui laisse une certaine latitude quant à son utilisation, par exemple pour "symbole d'un alphabet", ou "mot d'un lexique", ou encore "question d'un questionnaire". La communauté de recherche sur la reconnaissance visuelle des mots a, depuis quelques années, développé des efforts importants en matière de modélisation, mais aussi en matière de constitution de grandes bases de données comportementales permettant de tester les modèles et de réaliser des expériences virtuelles. C'est ainsi qu'ont été successivement publiés le "English Lexicon Project" (ELP: Balota, Yap, Cortese, Hutchison, Kessler, Loftis, Neely, Nelson, Simpson, & Treiman, 2007), qui fournit des données de décision lexicale et de dénomination pour 40481 mots anglais, le "French Lexicon Project" (FLP: Ferrand, New, Brysbaert, Keuleers, Bonin, Méot, Augustinova, & Pallier, 2010), qui fournit des données de décision lexicale pour 39840 mots français, et le "Dutch Lexicon Project" (DLP: Keuleers, Diependaele, & Brysbaert, 2010), qui fournit des données de décision lexicale pour 14089 mots hollandais.

Toutefois, alors que les plans factoriels classiques permettent de tester des effets par des techniques du type analyse de la variance, le test des prédictions fines fournies par les modèles au niveau des items fait plutôt appel à des techniques de régression ou corrélation visant à rendre compte d'une part maximale de la variance liée aux items. Le problème est que les données empiriques contiennent une certaine part de variance systématique, dont les modèles peuvent espérer rendre compte, mais aussi une part de variance aléatoire (bruit) qui échappe en principe à la modélisation cognitive. La pratique courante consiste à sélectionner les modèles de telle manière que les "meilleurs" modèles sont ceux qui rendent compte de la plus grande part de variance à l'aide du minimum de paramètres libres, et un certain nombre d'indices statistiques ont été construits dans ce but (Pitt & Myung, 2002). Mais ceci ne permet pas de répondre à la simple question: "est-ce que ce modèle rend convenablement compte de ces données?" Pour répondre à cette question, il faut savoir assez précisément quelle est la part de variance systématique dans les données empiriques. Si l'on rend compte

d'une part de variance moindre, c'est un "sous-ajustement" qui révèle une insuffisance du modèle. Si l'on rend compte d'une part de variance supérieure à la part de variance systématique, c'est un "sur-ajustement" qui indique que les prédictions sont corrélées avec du bruit aléatoire, ce qui est généralement dû à l'utilisation d'un trop grand nombre de paramètres libres dans le modèle et entraîne une faible capacité de généralisation de ce dernier.

Curieusement, si l'aspect "compétition" entre modèles concurrents a été très étudié, la question plus fondamentale de leur validation a été largement laissée de côté jusqu'à un passé récent. Dans une première approche de cette question, Rey, Courrieu, Schmidt-Weigand, et Jacobs (2009, article ci-joint) ont identifié une loi établissant le lien entre le nombre de participants à une collecte de données et la proportion de variance systématique dans ces données. Cette loi a la forme d'un coefficient de corrélation intraclasse (ICC) particulier, et rend bien compte de temps d'identification perceptive de mots anglais. Sur cette base, Courrieu, Brand-D'Abrescia, Peereman, Spieler, et Rey (2011, article ci-joint) ont développé une méthodologie rigoureuse permettant de tester la validité de modèles sur des bases de données comportementales, et cette approche s'est avérée satisfaisante pour des temps de dénomination de mots anglais et français. Enfin, Rey et Courrieu (2010, article ci-joint) ont établi les statistiques utiles (ICC et intervalles de confiance) pour la base de données DLP (Keuleers et al., 2010). Cette méthodologie récente nous a par ailleurs permis de conclure que les modèles de lecture actuellement les plus performants, tels que CDP++ (Perry, Ziegler, & Zorzi, 2010), sont encore assez loin de rendre compte de la totalité de la variance systématique des données (Rey, Brand-d'Abrescia, Peereman, Spieler, & Courrieu, 2010).

Références

Balota, D.A., Yap, M.J., Cortese, M.J., Hutchison, K.A., Kessler, B., Loftis, B., Neely, J.H., Nelson, D.L., Simpson, G.B., & Treiman, R. (2007). The English Lexicon Project. *Behavior Research Methods, 39(3)*, 445-459.

Courrieu, P., Brand-D'Abrescia, M., Peereman, R., Spieler, D., & Rey, A. (2011). Validated intraclass correlation statistics to test item performance models. *Behavior Research Methods*. doi: 10.1007/s13428-010-0002-7

Ferrand, L., New, B., Brysbaert, M., Keuleers, E., Bonin, P., Méot, A., Augustinova, M., Pallier, C. (2010). The French Lexicon Project: Lexical decision data for 38,840 French words and 38,840 pseudowords. *Behavior Research Methods, 42*, 488-496.

Keuleers, E., Diependaele, K., & Brysbaert, M. (2010). Practice effects in large-scale visual word recognition studies: A lexical decision study on 14,000 Dutch mono- and disyllabic words and nonwords.. *Frontiers in Psychology* 1:174. doi:10.3389/fpsyg.2010.00174

Perry, C., Ziegler , J.C., & Zorzi, M. (2010). Beyond single syllables: Large-scale modeling of reading aloud with the Connectionist Dual Process (CDP++) model. *Cognitive Psychology, 61*, 106–151.

Pitt, M.A., & Myung, I.J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences, 6(10)*, 421-425.

Rey, A.*,* Brand-d'Abrescia, M., Peereman, R., Spieler, D., & Courrieu, P. (2010). The nanopsycholinguistic approach: Item performance in disyllabic word naming. Oral communication presented at the *51st Annual Meeting of the Psychonomic Society*, St Louis, USA, November 18-21.

Rey, A., & Courrieu, P. (2010). Accounting for item variance in large-scale databases. *Frontiers in Psychology* 1:200. doi:10.3389/fpsyg.2010.00200

Rey, A., Courrieu, P., Schmidt-Weigand, F., & Jacobs, A.M. (2009). Item performance in visual word recognition. *Psychonomic Bulletin & Review, 16(3)*, 600-608.

Psychonomic Bulletin & Review
2009, 16 (3), 600-608
doi:10.3758/PBR.16.3.600

Item performance in visual word recognition

Arnaud Rey and Pierre Courrieu
CNRS and Université de Provence, Marseille, France

Florian Schmidt-Weigand
Universität Kassel, Kassel, Germany

and

Arthur M. Jacobs
Freie Universität Berlin, Berlin, Germany

Standard factorial designs in psycholinguistics have been complemented recently by large-scale databases providing empirical constraints at the level of item performance. At the same time, the development of precise computational architectures has led modelers to compare item-level performance with item-level predictions. It has been suggested, however, that item performance includes a large amount of undesirable error variance that should be quantified to determine the amount of reproducible variance that models should account for. In the present study, we provide a simple and tractable statistical analysis of this issue. We also report practical solutions for estimating the amount of reproducible variance for any database that conforms to the additive decomposition of the variance. A new empirical database consisting of the word identification times of 140 participants on 120 words is then used to test these practical solutions. Finally, we show that increases in the amount of reproducible variance are accompanied by the detection of new sources of variance.

The precision of theoretical accounts in the field of visual word recognition has significantly increased over recent years. Indeed, cognitive modelers have proposed several detailed descriptions of the structure and dynamics of the reading system (e.g., Ans, Carbonnel, & Valdois, 1998; Coltheart, Rastle, Perry, Langdon, & Ziegler, 2001; Grainger & Jacobs, 1996; Harm & Seidenberg, 2004; Perry, Ziegler, & Zorzi, 2007; Plaut, McClelland, Seidenberg, & Patterson, 1996; Seidenberg & McClelland, 1989). The fine-grained precision of these models has led to the development of so-called *computational models of reading* that can generate precise quantitative predictions. As a consequence, by making fine-grained assumptions about the cognitive architecture of visual word recognition, theorists have also remarkably increased the resolution of theoretical predictions.

This progress in theory has been accompanied by a corresponding gain of precision for empirical data. In a seminal study, Spieler and Balota (1997) asked 31 participants to read aloud a list of 2,870 English monosyllabic words and compared the mean naming latency for each item with the predictions of two computational models of word reading (i.e., Plaut et al., 1996; Seidenberg & McClelland, 1989). The results were somewhat surprising, since both of these models accounted for only a small amount of the item variance (3.3% for Plaut et al.'s model, 10.1% for Seidenberg and McClelland's). Spieler and Balota also noticed that the models explained the amount of variance less well than did the linear combination of three simple linguistic predictors: log frequency, word length, and neighborhood density (which accounted for 21.7% of the variance). Finally, when variables related to onset phonemes were added to the analysis, the simple predictors were able to account for 42% of the item variance. Item-level data therefore seem to provide a critical test for computational models of reading.

Seidenberg and Plaut (1998) claimed, however, that two reasons might explain the relatively low item variance accounted for by these models. First, item means are affected by several factors that are not addressed by these models. For example, they do not specify the processes involved in letter recognition or in the production of articulatory output. Balota and Spieler (1998) noticed, however, that the performance of these models remains surprisingly weak, since they fail to explain more variance than do three simple predictors (i.e., log frequency, word length, and neighborhood density) that are, in principle, captured by these models. Their second, and probably more critical, argument is based on the fact that item data include a substantial amount of error variance. The question is how substantial this amount of error variance is. Comparing Spieler and Balota's database with a similar database recorded by Seidenberg and Waters (1989),[1] they found a .54 correlation between item latencies in the two databases. This relatively low correlation indicates that a large amount of the variance in one database is absent from the other.

In the present study, in line with Seidenberg and Plaut's (1998) criticism, we address the issue of error variance in item databases (for a similar approach, see Rouder & Lu, 2005). More specifically, we propose practical solutions to estimate the amount of error variance as a function of the number of participants. Increasing the number of participants obviously decreases the amount of error variance (related to noise) while preserving the amount of reproducible variance (related to items). This outcome might appear trivial, but, paradoxically, none of the existing databases has seriously considered this issue.

In the next sections, we first provide a simple analysis of this problem. Second, we present a new empirical database consisting of the word identification scores of 140 participants on 120 words, and we use it to quantitatively estimate the amount of variance that should be accounted for as a function of the number of participants. Then, we propose a method to estimate the amount of reproducible variance from any database, and we give an

A. Rey, arnaud.rey@univ-provence.fr

example. Finally, we show that an increase in reproducible variance is accompanied by the detection of new sources of variance.

Problem Analysis

Let I be a population of items, let P be a population of participants, and let x be an experimental measure (e.g., response time) on $I \times P$. The usual additive decomposition model is

$$x = \mu + \alpha + \beta + \varepsilon, \tag{1.0}$$

where μ is the mean value of x on $I \times P$ and α, β, and ε are three independent random variables with mean 0 and variance σ_α^2, σ_β^2, and σ_ε^2, respectively.[2] The variable α is the *participant effect*, which takes a constant value for each given participant; β is the *item effect*, which takes a constant value for each given item; ε is considered as random noise. It is clear that the variable β, whose values characterize the items, is the variable of interest in this study.

One can derive from x another measure, denoted $x^{(n)}$, which is the arithmetic mean of x over n randomly selected distinct participants, and then obtain from Equation 1.0 the following decomposition:

$$x^{(n)} = \mu + \alpha^{(n)} + \beta + \varepsilon^{(n)}, \tag{1.1}$$

where the random variables $\alpha^{(n)}$, β, and $\varepsilon^{(n)}$ are always independent with 0 means, but their variances are now σ_α^2/n, σ_β^2, and σ_ε^2/n, respectively. When n increases, the contributions of $\alpha^{(n)}$ and $\varepsilon^{(n)}$ to the variance of $x^{(n)}$ clearly decrease. These reductions in the amounts of variance related to participants and noise lead to an increase in the amount of reproducible variance related to items.

One way to estimate the evolution of reproducible variance as a function of the number of participants is to compare two independent realizations of $x^{(n)}$. The amount of variance that is common to these two groups provides a good estimate of the total amount of reproducible variance, and it can be estimated by the squared correlation between two independent groups of n participants. Starting from Equation 1.1, a simple derivation, which is stated in Appendix A, leads to the following equations, which relate the population correlation coefficient r to the number of participants n and to the ratio between σ_β^2 and σ_ε^2.

To simplify the notation, one can define the ratio

$$q = \frac{\sigma_\beta^2}{\sigma_\varepsilon^2}, \tag{2.0}$$

so that the correlation between two independent realizations of $x^{(n)}$ is

$$\rho = \frac{nq}{nq + 1}, \tag{2.1}$$

which implies that

$$q = \frac{\rho}{n(1 - \rho)} \tag{2.2}$$

and also that

$$n = \frac{\rho}{q(1 - \rho)}. \tag{2.3}$$

Clearly, given any two of the three quantities ρ, q, and n, one can easily find the third.

In practice, one does not know the population parameters (ρ or q), and one must estimate at least one of them from a finite sample of the measure x on a sample of m items by n participants. As we shall see below, in the sections on estimation with large and small samples, the sample of x can be used to estimate ρ by means of Pearson's r correlation statistic.[3] Before this discussion, we present the experimental data that will be used to calculate the estimates.

The Database

The present database has two primary characteristics. First, it was collected using a standard perceptual identification task, the luminance-increasing paradigm (Rey, Jacobs, Schmidt-Weigand, & Ziegler, 1998; Rey & Schiller, 2005). As in most perceptual identification paradigms, participants generate a simple motor response as soon as they have identified a target stimulus. This experimental procedure therefore simplifies the model-to-data connection, since it can be assumed that word identification times can be directly compared with word identification latencies in a localist connectionist model like that of Grainger and Jacobs (1996). Second, 140 participants were recorded in this experiment. This large number makes it possible to estimate the amount of error variance in item mean latencies by comparing independent groups of participants consisting of 20, 30, . . . , up to 70 participants.

Participants. One hundred forty-four undergraduate students at Arizona State University participated in the experiment in partial fulfillment of a course requirement.[4] All of them were native English speakers and had normal or corrected-to-normal vision.

Stimuli. The words used in the experiment were a random sample of 120 monosyllabic, five-letter English words taken from a list of all monosyllabic five-letter words reported in the CELEX lexical database (Baayen, Piepenbrock, & van Rijn, 1993). The random selection was applied to provide a representative distribution for a maximum number of statistical word features.

Procedure. The experiment was run on an IBM PC 486 DX2 computer.[5] The stimulus words were typed in lowercase using letters created from table zero of the computer BIOS, in which each letter is defined in an 8 × 14 pixel matrix. To obtain a progressive increase in bottom-up information, the screen contrast was set to its maximum value. The background therefore was as dark as possible, and the stimulus luminance was as bright as possible. The experiment was done in a dark room lit only by a lamp placed behind the participants, to keep the keyboard visible without causing reflections on the screen.

The participants were seated 50 cm in front of the computer screen. The experiment started with a training session in which 6 of the 12 training items were presented. Data recording began with the 6 remaining training items, and, without transition, the 120 experimental trials were presented in a randomized order for each participant. Each trial began with a 1-sec presentation of a fixation mark ("+") in the center of the screen, which was replaced immediately by the target word. However, the target word

was initially written in black, just like the background, and so remained invisible to the participants. By incrementing the value of one of the RGB (red, green, blue) counters every 100 msec, the luminance of the target word increased progressively. Every counter was initially set to 0. After 100 msec, the red counter was set to 1 (with the green and blue counters remaining at 0). After another cycle (i.e., 200 msec after stimulus presentation), RGB was set to 1–1–0, then to 1–1–1 after three cycles, to 2–1–1 after four, and so forth. As soon as the participants could identify the target word, they interrupted the luminance-increasing process by pressing the space bar. Immediately after this response, the item was replaced by a pattern mask #####, which contained two more # characters than there were letters in the target word. Finally, the participants had to type in the word they had seen and press "return" to start the next trial. The screen remained black for 500 msec before the fixation point appeared again. For each trial, the response time was recorded as the time between the onset of the luminance-increasing procedure and the pressing of the space bar. The participants were told to concentrate on accuracy rather than on speed. Each experimental pass lasted about 25 min.

Results. After correcting obvious typing errors, 467 errors (2.7% of the data) remained. Four participants produced more than 10% errors and were excluded for this reason from further analysis. A trimming procedure excluded response times more than three standard deviations above and below a participant's mean (0.9% of the data). The resulting database was composed of 120 (words) × 140 (participants) word identification times, which included about 4% missing data.

Estimating the Amount of Reproducible Variance

Using this database of 120 items × 140 participants, it is now possible to estimate parameter q from Equation 2.1 by conducting a Monte Carlo study in the following way. Among the 140 participants, we randomly selected two independent, equally sized groups. Item means for each group were calculated, and correlation coefficients (r) were computed between the two groups on these item means. This procedure was repeated 1,000 times and for various group sizes (i.e., for 1, 5, 10, 15, . . . , 70 participants per group) to generate distributions of r (i.e., 1,000 correlations for each group size).

To test model validity, we computed the q value (Equation 2.1) that minimizes the standard prediction error of the observed correlations. One can easily obtain a first approximation of q—say, q_0—by applying Equation 2.2 (in which one replaces ρ with r) to the mean correlation observed for each group size, and then averaging all resulting q values. In the present case, we obtained $q_0 = 0.0618$. This first approximation was used as the starting point for a local search (MATLAB fminsearch procedure) to minimize the standard prediction error (i.e., the square root of the mean quadratic error). The obtained result was $q = 0.0607$, providing a standard prediction error of 0.0044. Figure 1 shows the mean observed correlations (with standard deviations) and the correlations predicted by Equation 2.1 (using the q value above) as functions of the number of participants per group. The result is that the predicted values are practically indistinguishable from the observed ones.

Using the q value above with Equations 2.1 and 2.3, one can now calculate the amount of reproducible variance obtained with a database composed of n participants, or the number of participants who are engaged in an experiment, in order to obtain a given amount of reproducible variance. For instance, using Equation 2.1 with $q = 0.0607$ and $n = 140$, one obtains $r = .89$, which means that by averaging the data of the 140 participants, one obtains an item data vector with 80% reproducible variance. Similarly, if one desires 90% reproducible variance, the corresponding r is $\sqrt{0.9} = .9487$, and via Equation 2.3, one finds that about 304 participants would be required.

Now, it is clear that another experiment, using a different task and different experimental conditions, would probably provide a different value for q. However, we can say that any experimental variable that conforms to the additive decomposition model (Equation 1.0) necessarily conforms to Equations 2.1–2.3, with an appropriate q value.[6]

Practical Method for Small Samples of Participants

The method used in the previous section is suitable for large samples of participants. However, experimenters commonly use participant samples of only 20–40. Thus, there is clearly a need to develop practical methods to estimate the percentage of variance that is reproducible when the number of participants is not large.

The proposed solution is similar to the one adopted in the section above, and it uses a Monte Carlo approach, which has the advantage of being distribution-free, thus avoiding the need for unverifiable hypotheses concerning the Gaussian nature of the variables. The principle used here is *permutation resampling* (Good, 1994; Opdyke, 2003). We describe this method hereafter and, in Appendix B, show an implementation in MATLAB code that is easy to use in practice. We then provide a model for the implementation details.

Given a data table of m items × n participants, first choose a group size n_g that is the greatest integer such that $n_g \leq n/2$. Then, randomly sample two independent groups of n_g participants, calculate item means for each group, and compute the correlation coefficients r between the resulting item means. When this has been repeated T times, one can sort the obtained r values in increasing order and easily find in this array the limits for any chosen confidence level. However, the obtained r estimates concern samples of n_g participants. To obtain the corresponding estimates for the whole sample of n participants, one can compute the q values corresponding to the obtained r values by using Equation 2.2, with n_g as the sample size parameter, and then use Equation 2.1 to compute the r values corresponding to the q values with sample size n. As for the choice of T, $T > 1{,}900$ provides precise enough estimates for most applications (Opdyke, 2003), so one can use $T = 2{,}000$.

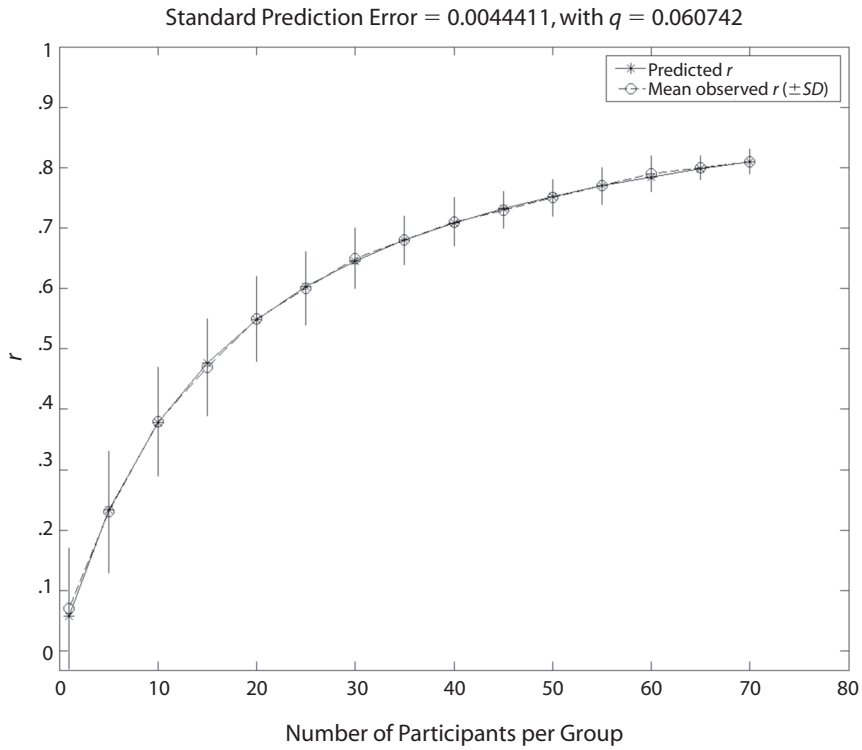


Figure 1. Means (with error bars for standard deviations) of the observed correlation coefficient distributions, with the predicted correlations (Equation 2.1), as a function of the number of participants per group.

To test the above method, we randomly selected from our database 14 subsamples whose size (n) varied from 10 to 140 participants (in steps of 10). The permutation resampling procedure (the function permuqr listed in Appendix B) was applied to each subsample, using a 95% confidence interval. Figure 2 shows the obtained mean r values, with confidence limits, as a function of the number of selected participants. Also plotted are the r values corresponding to the reference value $q = 0.0607$ (the best estimate of q for the whole database). As the figure shows, the estimates vary randomly in the neighborhood of the reference values, and they closely converge to these values as the number of participants increases. In these examples, the reference values were always within the 95% confidence interval provided by the method. We performed 25 independent replications of this experiment, corresponding to a total of 350 tests of the procedure. Globally, the reference values fell outside the 95% confidence interval only 8 times (and always for $n \leq 40$). This frequency of about .02 is smaller than, but not too far from, the expected .05 risk.

These observations suggest that the mean r is a reliable estimate for $n \geq 100$. However, for small samples of participants, it is better to use the lower limit of the 90% confidence interval—that is, the quantile of probability .05, hereafter denoted $r(.05)$—and then to provide the user with a statement in the form Prob[$r > r(.05)$] = .95.

We rapidly illustrate this approach using the MATLAB function permuqr, listed in Appendix B. First, we randomly selected a sample of 30 participants from our database; the resulting data table was named RT30. Then we applied the permuqr function as follows,

```
[q, confq, r, confr, ndr] = permuqr(RT30, 0, 0.90),
```

obtaining the output

```
q = 0.0713, confq = [0.0523; 0.0968], r = .6814,
   confr = [0.6106; 0.7439].
```

As we can see in this example, the q parameter was overestimated; however, the reference value (0.0607) is within the 90% confidence interval. The lower confidence limit of r is .6106, so one can state: Prob($r > .6106$) = .95. In other words, one can guarantee with 95% confidence that the reproducible percentage of item variance in the sample average vector is greater than 37.27%. This tells us that a model that accounts for about 40% of item variance, given this sample of 30 participants, is reasonably good in terms of its performance predictions. In a similar way, we can consider the upper confidence limit of r—that is, .7439—and state that Prob($r < .7439$) = .95. In other words, a model that accounts for more than 55.34% of the item variance (in our example) probably overfits the data by using too many free parameters, and thus actually accounts for a substantial part of the random noise.

Increasing the Amount of Reproducible Variance

If we assume that the amount of reproducible variance in naming with 30 participants is close to the value ob-

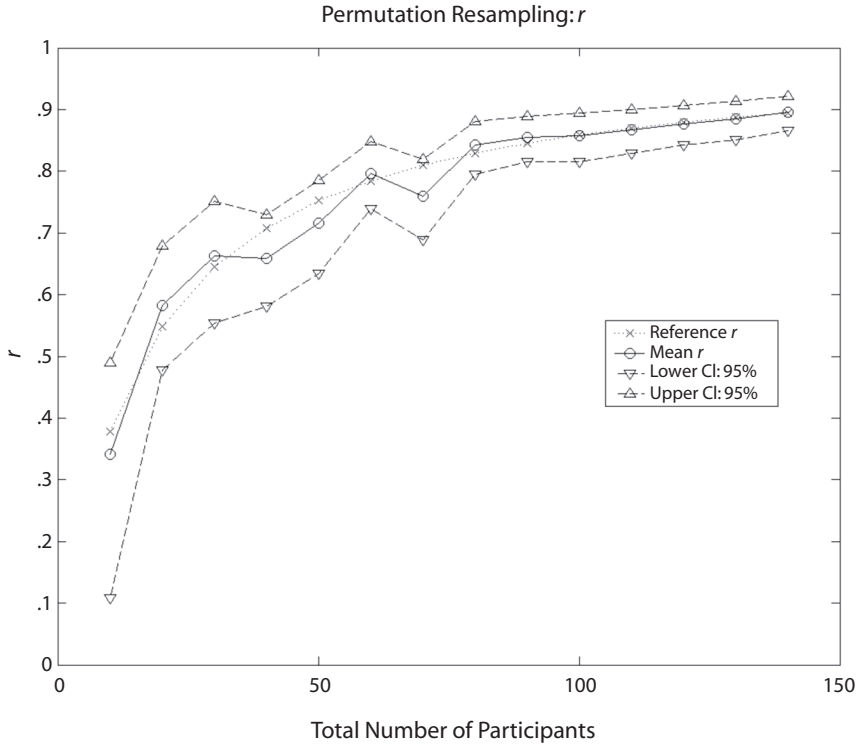


Figure 2. Mean and 95% confidence limits of the permutation resampling *r* distribution as a function of the total number of participants. The reference *r* values correspond to $q = 0.0607$.

tained in the present perceptual identification task (i.e., around 40%), cognitive modelers may come up against a critical problem. Indeed, we have already mentioned that Spieler and Balota (1997) reported that phonemic features, together with word frequency, neighborhood density, and length, accounted for 42% of the variance of item-naming latencies. Similarly, Perry et al. (2007), when testing the CDP+ model against the same item databases, were able to account for a similar amount of variance. From these results, one might conclude that psycholinguistic research has fully solved the problem of visual word recognition processes, since all of the reproducible variance at the level of items has been accounted for. The only remaining debate would concern the format of the explanation: Should we prefer a simple, linear model description or a sophisticated computational account?

A solution to this potential dilemma would be to increase the amount of reproducible variance by simply recording the performance of more participants. However, although the reproducible variance would then increase, the amount of variance explained by the linear or the computational model could likewise increase. If, by increasing the number of participants, one obtains 60% of the reproducible variance at the level of items, the linear model might also account for about 60% of the variance, and this result would probably mark the end of psycholinguistic research. Alternatively, as the amount of reproducible variance increases, other sources of variance might also become visible, such as variance related to morphological, syntactic, or semantic processes. This second, more optimistic outcome would then open the race for a new generation of more sophisticated models.

To determine which of these two outcomes is the correct one, we used the Monte Carlo study described above, in which mean item latencies were calculated for different sizes of participant groups. We then systematically correlated these item means to the log frequency of items. Figure 3 displays the evolution of correlation coefficients as a function of the number of participants per group when independent groups are compared (i.e., for estimating the amount of reproducible variance) and when item means are correlated with the log frequency.

The result is that an increase in the amount of reproducible variance is not accompanied by a proportional increase in the variance explained by log frequency. For example, with groups of 30 participants, on average 41% of the variance is reproducible, and 18% of the item variance is accounted for by log frequency. With groups of 70 participants, these values are now 66% and 23%, respectively. Thus, when increasing from 30 to 70 participants, an increase of 25% is observed in the reproducible variance, whereas an increase of only 5% is obtained in the variance accounted for by log frequency. This result suggests that the increase in reproducible variance allows for capturing new sources of variance that were initially not visible.

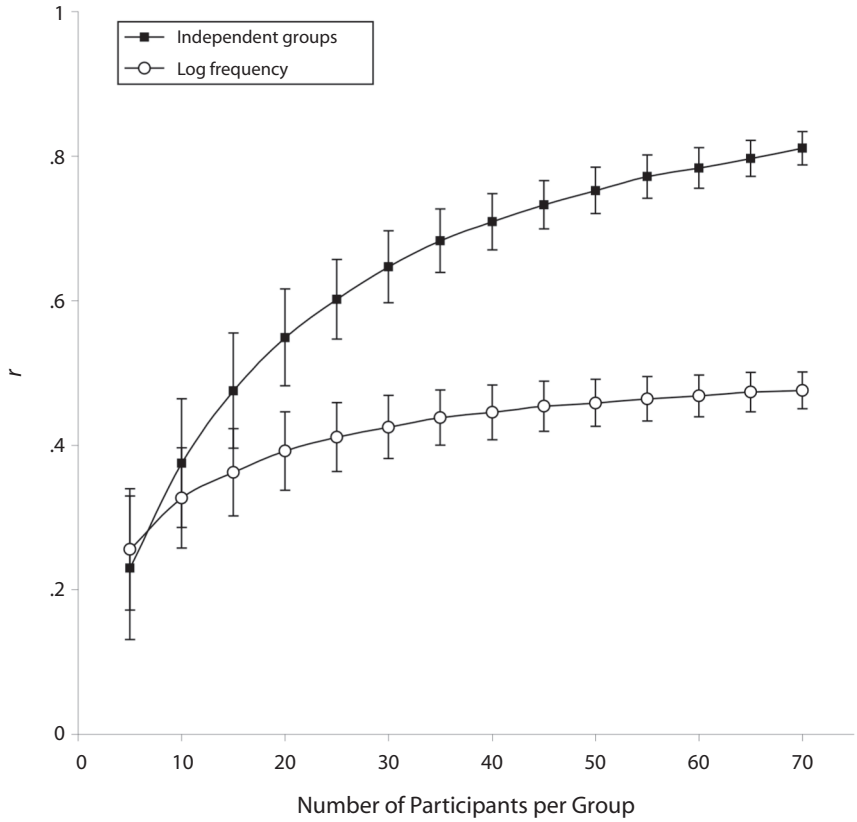


Figure 3. Means (with error bars for standard deviations) of correlations between independent groups composed of 5–70 participants (line with black squares) and between item means, computed for groups of 5–70 participants, and log frequency (line with white circles).

Discussion

Starting with a mathematical description of the reliability of item-level databases, we have proposed a method of estimating the amount of variance that models should account for when they are tested against a database with *n* participants. When *n* is sufficiently large (i.e., larger than 100), we have shown that the function relating the amount of reproducible variance and the number of participants in a given experimental paradigm can be approximated precisely. When *n* is relatively small, calculating confidence intervals using a permutation resampling method is still possible and is useful for estimating the boundaries of the amount of reproducible variance.

Following Balota and Spieler (1998) and Seidenberg and Plaut (1998), the present study provides new arguments concerning the amount of variance that models should account for. The main result concerns the relation between reproducible variance and the total number of participants involved in the computation of item means. On the basis of a common statistical model, we can confidently state that the present set of item mean response times (computed on the basis of the performance of 140 participants and recorded in this specific experimental setup[7]) is composed of 80% reproducible variance and 20% error variance. This information is of major importance, because one can now clearly evaluate the descriptive adequacy of computational models and the amount of variance that a given hypothesized cognitive architecture can account for.[8]

Conclusion

One can assume that the interaction between a given word, characterized by a set of properties (e.g., visual, phonological, or semantic), and the population of adult readers (supposed to share a similar cognitive architecture for processing written words) can be quantified by a measure of the processing time required to read the word in a given experimental situation. Likewise, if one considers a list of such words, it is a priori possible to rank those words according to their processing time and to evaluate the ability of computational models to reproduce this ranking. Using this item-level ranking might be misleading, however, if intra-item variability is greater than between-item variability. In this case, item-level databases only reflect general trends, and the fine-grained ranking of items remains hidden in an undesirable source of error variance.

The aim of the present study was to quantify the respective amounts of reproducible and error variance to determine the amount of variance that models should account for in item-level databases. The methodology we presented offers such quantification, together with practical solutions for estimating the amount of reproducible variance for any database. The conclusion is that collect-

ing large-scale databases composed of both many items and many participants will provide genuine challenges to future generations of computational models of word recognition.

AUTHOR NOTE

We are grateful to David A. Balota, Stephen D. Goldinger, Jeffrey N. Rouder, and one anonymous reviewer for their helpful comments. Correspondence should be sent to A. Rey, Laboratoire de Psychologie Cognitive, CNRS–Université de Provence, 3 place Victor Hugo, 13331 Marseille Cedex 3, France (e-mail: arnaud.rey@univ-provence.fr).

REFERENCES

ANS, B., CARBONNEL, S., & VALDOIS, S. (1998). A connectionist multiple-trace memory model for polysyllabic word reading. *Psychological Review*, **105**, 678-723.

BAAYEN, R. H., PIEPENBROCK, R., & VAN RIJN, H. (1993). The CELEX lexical database (CD-ROM). Philadelphia: Linguistic Data Consortium, University of Pennsylvania.

BALOTA, D. A., & SPIELER, D. H. (1998). The utility of item-level analyses in model evaluation: A reply to Seidenberg and Plaut. *Psychological Science*, **9**, 238-240.

COLTHEART, M., RASTLE, K., PERRY, C., LANGDON, R., & ZIEGLER, J. (2001). DRC: A dual route cascaded model of visual word recognition and reading aloud. *Psychological Review*, **108**, 204-256.

GOOD, P. (1994). *Permutation tests: A practical guide to resampling methods for testing hypotheses*. New York: Springer.

GRAINGER, J., & JACOBS, A. M. (1996). Orthographic processing in visual word recognition: A multiple read-out model. *Psychological Review*, **103**, 518-565.

HARM, M. W., & SEIDENBERG, M. S. (2004). Computing the meanings of words in reading: Cooperative division of labor between visual and phonological processes. *Psychological Review*, **111**, 662-720.

OPDYKE, J. D. (2003). Fast permutation tests that maximize power under conventional Monte Carlo sampling for pairwise and multiple comparisons. *Journal of Modern Applied Statistical Methods*, **2**, 27-49.

PERRY, C., ZIEGLER, J. C., & ZORZI, M. (2007). Nested incremental modeling in the development of computational theories: The CDP+ model of reading aloud. *Psychological Review*, **114**, 273-315.

PLAUT, D. C., MCCLELLAND, J. L., SEIDENBERG, M. S., & PATTERSON, K. (1996). Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review*, **103**, 56-115.

REY, A., JACOBS, A. M., SCHMIDT-WEIGAND, F., & ZIEGLER, J. C. (1998). A phoneme effect in visual word recognition. *Cognition*, **68**, B71-B80.

REY, A., & SCHILLER, N. O. (2005). Graphemic complexity and multiple print-to-sound associations in visual word recognition. *Memory & Cognition*, **33**, 76-85.

ROUDER, J. N., & LU, J. (2005). An introduction to Bayesian hierarchical models with an application in the theory of signal detection. *Psychonomic Bulletin & Review*, **12**, 573-604.

SEIDENBERG, M. S., & MCCLELLAND, J. L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review*, **96**, 523-568.

SEIDENBERG, M. S., & PLAUT, D. C. (1998). Evaluating word-reading models at the item level: Matching the grain of theory and data. *Psychological Science*, **9**, 234-237.

SEIDENBERG, M., & WATERS, G. S. (1989). Word recognition and naming: A mega study. *Bulletin of the Psychonomic Society*, **27**, 489.

SPIELER, D. H., & BALOTA, D. A. (1997). Bringing computational models of word naming down to the item level. *Psychological Science*, **8**, 411-416.

ZIMMERMAN, D. W., ZUMBO, B. D., & WILLIAMS, R. H. (2003). Bias in estimation and hypothesis testing of correlation. *Psicológica*, **24**, 133-158.

NOTES

1. In this study, 30 McGill University undergraduates named aloud 2,900 monosyllabic English words.

2. It is not necessary to assume that the random variables are Gaussian, but one can assume that the variances are finite and that $\sigma_{\varepsilon}^2 > 0$.

3. Indeed, if an m-dimensional vector $\mathbf{B}$ is hypothesized to be an approximation of β (or of any affine function of β) for the item set under consideration, a common procedure consists of averaging the n columns of the data table and comparing the resulting m-dimensional vector to $\mathbf{B}$ by means of Pearson's r statistic. However, even if $\mathbf{B} = \beta$, there is no chance that $r = 1$, because of the data random variance. In the best case, one could expect a correlation on the order of ρ, as defined by Equation 2.1, given that r is known to be a consistent asymptotically unbiased estimator of ρ (Zimmerman, Zumbo, & Williams, 2003).

4. We are indebted to Guy Van Orden, who allowed one of us to run the experiment in his laboratory.

5. We thank David Chesnet and Jonathan Grainger for providing us with the computer program to implement the luminance-increasing procedure.

6. This is visibly the case for our data, and note that the additive decomposition model has for many years been the most commonly used model for the analysis of experimental data (usually with the additional assumption that the variables are Gaussian or conform to the conditions of the central-limit theorem). If this model is grossly false for common experimental tasks and variables, this is a serious problem that greatly exceeds the focus of the present study.

7. Although the estimations generated with the present database provide a concrete example of estimating the amount of reproducible variance from any database having either a small or a large sample, such estimations may vary greatly from one database to another. Notably, there might be important differences between experimental paradigms (e.g., perceptual identification vs. naming), and the estimations given here therefore cannot generalize from one experimental setup to another.

8. This does not mean that models cannot handle error variance. Solutions have been proposed in which error variance or noise could be simulated by adding, for example, a variable response mechanism based on a normally distributed parameter (see, e.g., Grainger & Jacobs, 1996). Here, we simply wish to dissociate the modeling of reading processes, which can theoretically be considered as free of error variance, from the addition of noise within cognitive models. One may, however, argue that modeling should necessarily incorporate the presence of noise in the studied systems.

APPENDIX A

Let us consider the bivariate distribution of pairs (X, Y), where X and Y are independent realizations of $x^{(n)}$; that is, the n participants are never the same for X and for Y. The population correlation between X and Y, varying the items, is given by

$$\rho(X,Y) = \mathrm{Cov}(X,Y) \Big/ \sqrt{\mathrm{Var}(X)\,\mathrm{Var}(Y)},$$

where, using Equation 1.1, one has

$$\mathrm{Cov}(X,Y) = \mathrm{Cov}\left(\beta + \varepsilon_X^{(n)}, \beta + \varepsilon_Y^{(n)}\right) = \mathrm{Var}(\beta) = \sigma_\beta^2,$$

because the terms that are constant with respect to the item variable (μ and $\alpha^{(n)}$) play no role in the correlation, and the variables β, $\varepsilon_X^{(n)}$, and $\varepsilon_Y^{(n)}$ are independent.

For the same reasons, one has also

$$\mathrm{Var}(X) = \mathrm{Var}\left(\beta + \varepsilon_X^{(n)}\right) = \mathrm{Var}(\beta) + \mathrm{Var}\left(\varepsilon_X^{(n)}\right) = \sigma_\beta^2 + \sigma_\varepsilon^2 \big/ n,$$

and similarly,

$$\mathrm{Var}(Y) = \mathrm{Var}\left(\beta + \varepsilon_Y^{(n)}\right) = \mathrm{Var}(\beta) + \mathrm{Var}\left(\varepsilon_Y^{(n)}\right) = \sigma_\beta^2 + \sigma_\varepsilon^2 \big/ n.$$

Thus, finally,

$$\rho(X,Y) = \frac{\sigma_\beta^2}{\sigma_\beta^2 + \sigma_\varepsilon^2 \big/ n} = \frac{n\sigma_\beta^2 \big/ \sigma_\varepsilon^2}{n\sigma_\beta^2 \big/ \sigma_\varepsilon^2 + 1}.$$

Not surprisingly, the expression above is similar to that of an intraclass correlation coefficient.

APPENDIX B

Here is MATLAB code of the permuqr function, which provides expected q and r values, with confidence intervals of chosen probabilities confp, from a data table x. For ease of reading, structural coding is set in boldface and comments in italics.

```
function [q,confq,r,confr,ndr] = permuqr(x,missing,confp,dr)
% Permutation Resampling to estimate q, r, and confidence
% intervals of given probabilities "confp" (row vector),
% from the m-items by n-participants data table x,
% where "missing" is the code for missing data in x.
% The first (second) row of "confq" corresponds to the
% lower (upper) confidence limit(s), similarly for "confr."
% An optional desired r (dr) provides the necessary n (ndr)
% r^2 is the reproducible proportion of item variance when
% one averages the n columns of x.
resample = 2000; % T > 1900 (see Opdyke, 2003)
[m,n] = size(x); confp = 1-confp; % Proba to alpha
ng = fix(n/2); % Number of participants per group
rt = zeros(resample,1);
for t = 1:resample
ok = false;
while ~ok
xp = x(:,randperm(n)); % Random participant permutation
ng1 = zeros(m,1); mg1 = ng1; ng2 = ng1; mg2 = ng1;
for i = 1:m
for j = 1:ng  % First group
if xp(i,j) ~ = missing
ng1(i) = ng1(i) + 1; mg1(i) = mg1(i) + xp(i,j);
end
end
for j = (ng + 1):(2*ng) % Second group
if xp(i,j) ~ = missing
ng2(i) = ng2(i) + 1; mg2(i) = mg2(i) + xp(i,j);
end
end
end
if (min(ng1) > 0) && (min(ng2) > 0), ok = true; end
end
mg1 = mg1./ng1; mg2 = mg2./ng2;
```

APPENDIX B (Continued)

```
rr = corrcoef([mg1, mg2]); rt(t) = rr(1,2);
end
q = rn2q(mean(rt),ng); r = qn2r(q,n); rt = sort(rt);
nconf = length(confp); confindex = zeros(2,nconf);
confindex(1,:) = round(resample*confp/2) + 1;
confindex(2,:) = round(resample*(1-confp/2));
confq = rn2q(rt(confindex),ng); confr = qn2r(confq,n);
if nargin = = 4, ndr = round(qr2n(q,dr)); else ndr = []; end
function q = rn2q(r,n) % Provides q given r and n
q = r./(n.*(1-r));
function r = qn2r(q,n) % Provides r given q and n
r = (n.*q)./(n.*q + 1);
function n = qr2n(q,r) % Provides n given q and r
n = r./(q.*(1-r));
```

(Manuscript received September 10, 2007;
revision accepted for publication January 23, 2009.)

Behav Res
DOI 10.3758/s13428-010-0020-5

Validated intraclass correlation statistics to test item performance models

Pierre Courrieu · Muriele Brand-D'abrescia · Ronald Peereman · Daniel Spieler · Arnaud Rey

Abstract A new method, with an application program in Matlab code, is proposed for testing item performance models on empirical databases. This method uses data intraclass correlation statistics as expected correlations to which one compares simple functions of correlations between model predictions and observed item performance. The method rests on a data population model whose validity for the considered data is suitably tested and has been verified for three behavioural measure databases. Contrarily to usual model selection criteria, this method provides an effective way of testing under-fitting and over-fitting, answering the usually neglected question "does this model suitably account for these data?"

Keywords Model test · Misfit detection · Intraclass correlation · Item performance databases

P. Courrieu (✉) · A. Rey
Laboratoire de Psychologie Cognitive, UMR CNRS 6146,
Université de Provence, Centre Saint Charles,
Bat. 9, Case D, 3 Place Victor Hugo,
13331, Marseille cedex 3, France
e-mail: pierre.courrieu@univ-provence.fr

M. Brand-D'abrescia
LEAD, CNRS, Université de Bourgogne,
Dijon, France

R. Peereman
LPNC, CNRS, Université Pierre Mendès France,
Grenoble, France

D. Spieler
School of Psychology, Georgia Institute of Technology,
Atlanta, GA, USA

Introduction

Theoretical models of perceptual and cognitive processing are commonly able to provide quantitative performance predictions at the item level. For instance, in the field of visual word recognition, recent models of reading are able to predict response times to individual word stimuli (the items) in various tasks, such as lexical decision, word naming or word identification (Coltheart, Rastle, Perry, Langdon, & Ziegler, 2001; Perry, Ziegler, & Zorzi, 2007, 2010; Plaut, McClelland, Seidenberg, & Patterson, 1996; Seidenberg & McClelland, 1989). Empirical databases have been collected to test these theoretical predictions at the item level, which in several cases resulted in disappointing outcomes: the tested models accounted only for a small amount of empirical item variance (Balota & Spieler, 1998; Seidenberg & Plaut, 1998; Spieler & Balota, 1997). This can result from the fact that the tested models are erroneous or incomplete; however, another possibility is that the empirical data are not accurate enough to allow good fits of plausible models, and we miss methods to make clear conclusions on these two points. In fact, it has recently been shown that the amount of reproducible variance of word identification times is related to the number of participants used in the data collection by a simple law having the form of an intraclass correlation coefficient (Rey, Courrieu, Schmidt-Weigand, & Jacobs, 2009). This constitutes a suitable reference for testing item-level performance models, provided we can be sure that the considered empirical data set actually fulfils the above-mentioned law. The main purpose of this paper is to provide an efficient test of this statistical model for every item level data set, to show that this statistical model actually

Published online: 12 November 2010

applies to widely used behavioural measures, and to show how to use validated correlation statistics to test model predictions. This widely extends the prior work of Rey et al., (2009) and provides a complete methodology to test item performance models on empirical databases.

In order to test a model, one usually collects empirical data to be compared to the corresponding model predictions, and one optimizes the free parameters (if any) of the model in order to minimise the prediction error or to optimise some "goodness of fit" measure. At this point, one must judge (a) whether the considered model suitably accounts for the data or not, and (b) whether this model must be preferred or not to other concurrent models. Point (b) is called "model selection", and it has been widely studied in the literature (Akaike, 1974; Hannan & Quinn, 1979; Hansen & Yu, 2001; Kass & Raftery, 1995; Myung, Pitt, & Kim, 2005; Pitt & Myung, 2002; Rissanen, 1996; Schwarz, 1978). Note, however, that an answer to point (b) does not necessarily imply a similar answer to point (a), and surprisingly, this last point has been almost completely neglected in the literature, leading to difficulties in interpreting a number of results (e.g. Spieler & Balota, 1997). There are two ways for a model fit to be bad: it can be "under-fitting" or "over-fitting". Under-fitting results in a large prediction error and is generated by erroneous or incomplete models. Over-fitting is more insidious because it results in a small prediction error for the current data, but the model is not able to generalise suitably, and the results are poorly reproducible. This is a well-known consequence of using too many free parameters in a model to fit the empirical data in such a way that the model encodes a substantial part of the data random noise instead of capturing essentially the data regularities. This is why usual model selection criteria such as the Akaike Information Criterion, abbreviated AIC (Akaike, 1974), or the Bayesian Information Criterion, abbreviated BIC (Schwarz, 1978), require optimising a compromise between the goodness of fit (maximum log-likelihood, for these criteria) and the number of free parameters in the model. However, none of these model selection criteria allows us to detect under-fits or over-fits; they just indicate a "winner" in a given set of competing models.

For few years, considerable efforts have been devoted to collecting and developing large-scale databases that provide behavioural measures at the item level. Each item measure is usually based on an average over a number of participants. For instance, this is the case in the recent English and French Lexicon Projects (Balota, Yap, Cortese, Hutchison, Kessler, Loftis, Neely, Nelson, Simpson, & Treiman, 2007; Ferrand, New, Brysbaert, Keuleers, Bonin, Méot, Augustinova, & Pallier, 2010), which allow researchers to test various hypotheses and reading models on large sets of empirical data. Building factorial designs on such data is quite easy; however, testing item-level performance models remains problematic because one does not know what can be considered as a good model fit for these data. A solution to this problem would be to provide, together with the behavioural measures, some reference "goodness of fit" measure with suitable under-fitting and over-fitting limits. This is the ultimate goal of this work, and the Matlab program named "ECVT" (for "Expected Correlation Validity Test") listed in Appendix A provides an operational solution to the problem, together with an efficient test of the validity of the adopted approach for the data set to be processed. Matlab users can directly copy the code in their Matlab editor and use it, while the listed code can also serve as an implementation model for other platforms. Comments in the code (at right of "%") provide indications for the use of the program, as well as for the actions of its various parts. The reader will also find in Appendix B an example of how to use the ECVT program with helpful comments.

Hereafter, we describe the methods implemented in the ECVT program, we evaluate their efficiency and performance on artificial data, and we test their relevance on three real databases of word identification and word naming times. In "Population model", we present the adopted population model and we derive theoretical correlation functions. In "Correlation estimates", we present statistics suitable to estimate the useful correlations. In "Expected correlation validation test", we present a test to validate (or invalidate) the population model and derived correlations for a given data set. In "Testing the test", we demonstrate the efficiency and effectiveness of this test on artificial data sets. In "Testing real response time databases", we validate the approach on three real behavioural databases. In "Testing regression models on simulated data", we demonstrate the use and performance of the new tool to test models. Finally, we consider recent examples of reading model fits and we conclude in "Discussion and conclusion".

Population model

In this section, we first define a statistical model of the behavioural measures we plan to account for. As we shall see, this is just an additive decomposition model commonly used for continuous variables (Behavioural variable model). From this model, we then derive a measure of the proportion of item variance that is not random, that is, the proportion of item variance that a perfect model should account for. As we shall see, this derivation results in a well-known intraclass correlation coefficient, commonly abbreviated ICC (Item performance correlation between

equal size samples of participants). Finally, we define models' fitting measures that suitably compare to the ICC, and there are mainly two distinct kinds of models with different appropriate fitting measures (Item performance correlation between observed and simulated data and Item performance correlation between a sample of participants and a predictor).

Behavioural variable model

Let I be a population of items, let P be a population of participants, and let X be a behavioural measure on $I \times P$ (e.g. response time). One assumes that X conforms to the usual additive decomposition model:

$$X = \mu + \alpha + \beta + \varepsilon, \tag{1}$$

where μ is the mean value of X on $I \times P$, and α, β, and ε are three independent random variables of mean zero, and of variance σ_α^2, σ_β^2, and σ_ε^2, respectively. The variable α is the participant effect, and it takes a constant value for each given participant. The variable β is the item effect, and it takes a constant value for each given item. The variable ε is considered as a random noise; however, it can as well result from the combination of an item-participant interaction and of a true random noise. The variable β, whose values characterise the items, is the variable of interest in this study.

One can derive from X another measure, denoted $X^{(n)}$, that is the arithmetic mean of X over n randomly selected distinct participants (thus $X^{(1)} = X$); then one obtains from (1) the following decomposition:

$$X^{(n)} = \mu + \alpha^{(n)} + \beta + \varepsilon^{(n)}, \tag{2}$$

where the random variables $\alpha^{(n)}$, β, and $\varepsilon^{(n)}$ are always independent with means zero, but their variances are now σ_α^2/n, σ_β^2, and σ_ε^2/n, respectively.

Item performance correlation between equal size samples of participants

Consider now the bivariate distribution of pairs (x, y), where x and y are independent realisations of $X^{(n)}$. Then the population correlation between x and y, varying the items, is given by:

$$\rho(x,y) = Cov(x,y)/(Var(x)Var(y))^{1/2},$$

where, using (2), one has:

$$Cov(x,y) = Cov(\beta + \varepsilon_x^{(n)},\ \beta + \varepsilon_y^{(n)}) = Var(\beta) = \sigma_\beta^2,$$

because the terms that are constant with respect to the item variable (μ and $\alpha^{(n)}$) play no role in the correlation, and the variables β, $\varepsilon_x^{(n)}$, and $\varepsilon_y^{(n)}$ are independent.

For the same reasons, one has also:

$$Var(x) = Var(\beta + \varepsilon_x^{(n)}) = Var(\beta) + Var(\varepsilon_x^{(n)}) = \sigma_\beta^2 + \sigma_\varepsilon^2/n,$$

and similarly:

$$Var(y) = Var(\beta + \varepsilon_y^{(n)}) = Var(\beta) + Var(\varepsilon_y^{(n)}) = \sigma_\beta^2 + \sigma_\varepsilon^2/n.$$

Thus, finally,

$$\rho(x,y) = \frac{\sigma_\beta^2}{\sigma_\beta^2 + \sigma_\varepsilon^2/n}. \tag{3}$$

One can recognise in (3) the expression of a well-know intraclass correlation coefficient (ICC), that is the "ICC(C, k), Cases 2 and 2A" coefficient, according to the nomenclature of McGraw and Wong (1996). To simplify the notation, it is convenient to define the ratio:

$$q = \sigma_\beta^2/\sigma_\varepsilon^2, \tag{4}$$

so that the correlation between two independent realizations of $X^{(n)}$ is:

$$\rho = \frac{nq}{nq+1}, \tag{5}$$

which implies that:

$$q = \frac{\rho}{n(1-\rho)}, \tag{6}$$

and also that:

$$n = \frac{\rho}{q(1-\rho)}, \tag{7}$$

which are convenient formulas for finding a parameter when one knows the two other ones, usually replacing q an ρ by their estimates.

The ICC therefore provides, for a given dataset, a reference correlation value for model tests. As described in the following sections, a distinction has to be made, however, between two modelling approachesthat are both designed to account for item variance. In a first approach (Item performance correlation between observed and simulated data), one considers theoretical item performance as generated by full simulation models able to simulate participant variability (very rare to date, but probably available in a near future), while in the second approach (Item performance correlation between a sample of participants and a predictor), one provides an account of theoretical item performance as generated by predictors, as in multiple regression approaches (e.g., Spieler & Balota, 1997; Yap & Balota, 2009). Note that recent simulation models are in fact used as simple predictors (e.g., Perry et al. 2010).

Item performance correlation between observed and simulated data

Consider now a variable V, that could be generated, for instance, by a full simulation model, and that is affinely related to X by:

$$V = aX + b \tag{8}$$

where $a \neq 0$, and b are two real numbers. Then one has:

$$V^{(n)} = aX^{(n)} + b = (a\mu + b) + a\alpha^{(n)} + a\beta + a\varepsilon^{(n)}.$$

Let x be a realization of $X^{(n)}$, and let v be an independent realization of $V^{(n)}$. Then, there is a realization y of $X^{(n)}$ such that $v = ay + b$, and:

$$\begin{aligned}\rho(x,v) &= Cov(x,v)/(Var(x)Var(v))^{1/2} \\ &= aCov(x,y)/(Var(x)a^2Var(y))^{1/2} \\ &= sign(a)\rho(x,y),\end{aligned} \tag{9}$$

and thus:

$$|\rho(x,v)| = \rho(x,y). \tag{10}$$

In other words, if a simulation model generates data that fulfil (8), then one can expect that groups of simulated participants provide mean item performance values whose absolute correlation with those of human participant groups of the same size (n) is given by (3)–(5).

Item performance correlation between a sample of participants and a predictor

Instead of building simulation models whose output fulfils (8), modellers commonly try to predict the unknown variable β that appears in (1), without modelling the participant effect and the random variability. So, it is of interest to know what happens if a model generates a variable B affinely related to β by:

$$B = a\beta + b, \tag{11}$$

for some real numbers $a \neq 0$, and b. Let x be a realization of $X^{(n)}$, and let B be defined as in (11), then on has:

$$\begin{aligned}&Cov(x,B) = Cov(\beta + \varepsilon_x^{(n)}, a\beta) = a\sigma_\beta^2, \\ &Var(x) = Var(\beta + \varepsilon_x^{(n)}) = Var(\beta) + Var(\varepsilon_x^{(n)}) = \sigma_\beta^2 + \sigma_\varepsilon^2/n, \\ &Var(B) = a^2\sigma_\beta^2,\end{aligned}$$

and thus:

$$\rho(x,B) = Cov(x,B)/(Var(x)Var(B))^{1/2} = sign(a)\frac{\sigma_\beta}{\left(\sigma_\beta^2 + \sigma_\varepsilon^2/n\right)^{1/2}}, \tag{12}$$

that is:

$$\rho^2(x,B) = \rho(x,y), \tag{13}$$

where, at new, y represents a realization of $X^{(n)}$ independent of x. In other words, if a model generates a variable that fulfils (11), then one can expect that the squared correlation of this variable with the mean item performance of a group of n participants is given by (3)–(5). Note that a coefficient similar to $\rho(x, B)$ is known in the framework of the Generalizability Theory as the "Generalizability Coefficient" (Webb, Shavelson, & Haertel, 2006).

Correlation estimates

In "Population model", we defined suitable correlations at the population level. In "Correlation estimates", we present practical estimates of these correlations for finite data samples.

Intraclass correlation coefficient

We consider two distinct methods to estimate the ICC. The first one is based on the usual analysis of variance (ANOVA) approach. It is fast and accurate, and provides reliable confidence limits for the ICC. However, it assumes that the underlying variables are approximately Gaussian. The second approach is based on a Monte-Carlo method known as "Permutation Resampling". It is distribution free and highly flexible; however, it requires much more computational effort than the ANOVA approach. We observed that the ANOVA approach is less sensitive to missing data than the Permutation Resampling approach; however, this point will not be developed in this paper.

ANOVA approach

In practice, one randomly selects a sample of m items in the item population, a sample of n participants in the participant population, and data are collected in the form of an $m \times n$ matrix of behavioural measures (x_{ij}), $1 \leq i \leq m$, $1 \leq j \leq n$. A standard analysis of variance (ANOVA) of this matrix provides three variation sources:

1. The between rows item effect, whose mean square is denoted MSi, with degrees of freedom $dfi = m - 1$, and expected value $EMSi = n\sigma_\beta^2 + \sigma_\varepsilon^2$.
2. The between-columns participant effect, whose mean square is denoted MSp, with degrees of freedom $dfp = n - 1$, and expected value $EMSp = m\sigma_\alpha^2 + \sigma_\varepsilon^2$.
3. The residual error effect, whose mean square is denoted MSe, with degrees of freedom $dfe = (m-1)(n-1)$,

and expected value $EMSe = \sigma_{\varepsilon}^2$. More generally, let N be the total number of available measures in the matrix, then $dfe = N - 1 - dfi - dfp$.

Then, it is easy to see that one has an estimate of the q ratio (4) with:

$$\widehat{q} = \frac{MSi - MSe}{n\,MSe}, \tag{14}$$

and one can estimate the intraclass correlation coefficient $\rho(x, y)$ of (3)–(5) by:

$$\widehat{\rho} = \frac{n\widehat{q}}{n\widehat{q} + 1} = \frac{MSi - MSe}{MSi}. \tag{15}$$

Moreover, the literature provides confidence limits and test formulas for the intraclass correlation coefficient (McGraw & Wong, 1996; Shrout & Fleiss, 1979). The confidence interval of probability 1-α of (15) is given by:

$$\left[1 - \frac{F_{1-\alpha/2}(dfi, dfe)}{F_{obs}},\quad 1 - \frac{1}{F_{obs} \times F_{1-\alpha/2}(dfe, dfi)}\right], \tag{16}$$

where $F_{obs} = MSi/MSe$, and $F_p(a, b)$ is the quantile of probability p of Fisher F distribution with a (numerator) and b (denominator) degrees of freedom. Take care to the reversed order of degrees of freedom for the upper confidence limit in (16). Note also that, in this context, α denotes the usual type I error risk (not the participant effect).

Special approaches of the intraclass correlation have been developed for the particular case of binary observations (Ahmed & Shoukri, 2010), which can be useful for the analysis of accuracy variables, for instance. However, in this paper, we more particularly focus on continuous behavioural variables such as reaction times.

Permutation Resampling approach

The analysis stated in "Item performance correlation between equal size samples of participants" clearly shows the relation between the intraclass correlation and the correlation of average vectors. This suggests the possible use of a Monte-Carlo type method named "Permutation Resampling" (Opdyke, 2003) to compute the intraclass correlation coefficient. Despite the computational effort this method requires, Rey et al. (2009) preferred it because it is distribution free. Another advantage is the flexibility of this method in what concerns the number of participants taken into account, which will allow us to build a useful test in "Expected correlation validity test" below.

The Permutation Resampling procedure is as follows. Given a data table of m items × n participants, first choose a group size $n_g \leq n/2$. Then randomly sample two independent groups of n_g participants each, calculate item means for each group, and compute the correlation coefficient r between the two resulting vectors of size m. Repeat this T times, then the average of obtained r values is an estimate of the intraclass correlation coefficient (ICC) for a data set of n_g participants. The larger T is, the more accurate the estimate. In order to obtain the ICC for the whole data set with n participants, one uses the average correlation and n_g to obtain an estimate of q by (6), then one extrapolates the desired ICC using (5) with q and n as arguments.

Model correlation

There are two cases that must be distinguished, the case of full simulation models (Item performance correlation between observed and simulated data) and the case of predictors (Item performance correlation between a sample of participants and a predictor). In both cases, human data are summarized in the form of a m components vector of mean item performances:

$$x_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}, \quad i = 1...m. \tag{17}$$

In the case of a full simulation model, the model prediction vector is of the form:

$$v_i = \frac{1}{n}\sum_{j=1}^{n} v_{ij}, \quad i = 1...m, \tag{18}$$

and if the model data fulfil (8), then one has the null hypothesis (10), where the estimate $\widehat{\rho}$ of $\rho(x, y)$ is given by (15), and the estimate of $\rho(x, v)$ is the Pearson r correlation statistic between the vectors (17) and (18).

In the case of a simple predictor, this one is of the form:

$$B = (b_i), \quad i = 1...m, \tag{19}$$

and if it fulfils (11), then one has the null hypothesis (13), where the estimate $\widehat{\rho}$ of $\rho(x, y)$ is given by (15), and the estimate of $\rho(x, B)$ is the Pearson r correlation statistic between the vectors (17) and (19).

In both cases, the model fit statistic is a powered absolute correlation of the form $|r|^c$, $c \in \{1, 2\}$, with c = 1 for simulation models and c = 2 for predictors. Under the null hypothesis (10) or (13), $|r|^c$ must belong to the ICC confidence interval (16) with probability 1-α of this

interval. If it does not, then one can reject the null hypothesis (with risk α) and conclude that the considered model does not suitably fit the data. Given that the ICC is the reference correlation value that model fit statistics must match as closely as possible, we refer to the ICC as the "Expected Correlation" in this context.

Expected correlation validation test

The validity of the approach developed in "Population model" and "Correlation estimates" critically depends on the assumption that the considered behavioural measure fulfils the additive decomposition model (1), or an equivalent variant, which leads to the law (3) for the expected correlation. However, this is not necessarily the case for every experimental variable, and thus a prior condition to the use of an expected correlation like the ICC (15), as a reference value to test models, is that one can verify that the considered data actually fulfil the law (3). In order to do this for their word identification time database, Rey et al. (2009) used a series of Permutation Resampling procedures, like the one described in "Permutation Resampling approach", with distinct participant group sizes (n_g's). Then they computed an estimate of the q ratio that minimized the sum of squared differences between the observed ICC estimates and those predicted by (5) for the various selected n_g values. The predicted and observed ICCs, as functions of the group size, were plotted in order to allow visual comparison, and the similarity of the two graphs appeared impressive, leading to the conclusion that the data suitably fulfilled the expected correlation model (3). The conclusion was correct in this case; however, visual appreciation is not always easy and reliable, as will be shown below. Another available information is the prediction error measure; however, we do not know the critical error magnitude (if any) to reject the model (3) for the considered data. So, we need a clear and easy to use test of validity of the expected correlation model (3) for every item level data set. In fact, such a test can easily be built using a procedure similar to the one described above, but where one replaces the prediction error measure by a suitable statistic whose theoretical distribution is known.

Consider the empirical distribution of T correlation values generated by Permutation Resampling for a given group size n_g (see Permutation Resampling approach). This distribution has an average $\bar{r}_g$, which is possibly an estimate of the ICC for n_g participants, and its variance is denoted s_g^2. Let ρ_g be the true, unknown, expected correlation for group size n_g. Given that the sampled correlation values are independent realizations of the same bounded random variable (in [-1, 1]), all moments of this variable exist, and the Central Limit Theorem does apply. Thus, as T increases, the average correlation r_g rapidly converges to a normally distributed random variable of mean ρ_g and of variance s_g^2/T. This implies that the random variable $T^{1/2}(\bar{r}_g - \rho_g)/s_g$ is normally distributed with mean 0 and variance 1. Now, consider a series of K independent Permutation Resampling estimations for K distinct group sizes, then, by definition of the χ^2 random variable with K degrees of freedom, one has:

$$\sum_{g=1}^{K} \left(T^{1/2}(\bar{r}_g - \rho_g)/s_g \right)^2 \rightarrow \chi^2(K). \tag{20}$$

If one hypothesises that (3) is valid for the considered data set, then there is a constant q such that, by (5), one has the null hypothesis:

$$\rho_g = \frac{n_g q}{n_g q + 1}, \quad g = 1..K. \tag{21}$$

The optimal determination of q is the one that minimises (20), while the ρ_g's in (20) are determined by (21). In practice, this is easy to obtain using a local search procedure such as Newton-Raphson iterations for zeroing the derivative of (20) with respect to q. This is implemented in the Matlab sub-function named "minChi2" listed in Appendix A. In the ECVT program, one uses T = 500, which was found to provide accurate results with an acceptable computational effort.

The choice of the series of K group sizes is somewhat arbitrary, and it is partially constrained by the total number of available participants (n). In the Matlab program listed in Appendix A, the series are built in order to obtain K equally spaced group sizes; while K is as close as possible to 12, the greatest group size is equal to the greatest integer lower or equal to $n/2$, and the lowest group size is minimally greater than or equal to the group size spacing. Note that using a very small group size can cause resampling difficulties in cases where there is a certain amount of missing data in the data set.

Finally, if the $\chi^2(K)$ value, obtained by (20) in the conditions described above, is significant, then the null hypothesis (21) can be rejected (with the chosen risk), which means that (1) and (3) probably do not provide a valid model for the considered data.

Testing the test

In order to examine the performance of the test described in "Expected correlation validation test", we are going to

test artificial data sets that fulfil or do not fulfil the variable model (1) by construction. In order to have an idea of the discrimination power of the test, it is desirable that the data can deviate from (1) at various degrees, including a degree zero, which is simply the conformity to (1). This can be obtained by generalising (1) in the following way:

$$X = \mu + \alpha + \gamma\lambda + \varepsilon, \tag{22}$$

where μ, α, and ε are defined as in (1), λ is the normalised item effect with mean 0 and variance 1, and γ is the "participant sensitivity" to the item effect. The participant sensitivity has a fixed value for each participant (as α), and has global mean $\overline{\gamma}$ and variance σ_γ^2. In the special case where $\sigma_\gamma^2 = 0$, one obtains (1), with $\beta = \overline{\gamma}\lambda$. For the generalised model, it is convenient to define the two following ratios:

$$q = \overline{\gamma}^2/\sigma_\varepsilon^2, \tag{23}$$

$$u = \sigma_\gamma^2/\sigma_\varepsilon^2. \tag{24}$$

Thus, (22) reduces to (1) if $u = 0$, and, as one can verify, (3) is valid only in this case.

Artificial data tables, of 360 items by 120 participants, have been generated using (22) with $q = 1/16$ and four values of $u \in \{0,1/36,1/16,1/4\}$. Figure 1 shows four plots generated by the Matlab function "ECVT" listed in the Appendix. Each plot compares the series of observed $\overline{r}_g$ values with the corresponding ρ_g values predicted by (21) for a given value of u, and the result of the validity test (20) appears in the title of the plot. As one can see, the two graphs are confounded, and the test is clearly non-significant for $u = 0$. However, for all non-zero values of u, the test is highly significant, and thus, the non-conformity of the data to model (1) is detected. Moreover, one can observe that for $u = 1/36$, the test detected the existing difference, while this one is not visible at the ordinary figure scale. In fact, a very small difference becomes visible when the figure is enlarged. This not only suggests that the test (20) is powerful, but also that visual inspection of graphs is not reliable enough in this problem. The four experiments of Fig. 1 were repeated 200 times each, and one recorded the frequency of rejection of the null hypothesis for two conventional type I error risks ($\alpha = 0.01$ and $\alpha = 0.05$) and for each value of u. As one can see in Table 1, the frequency of rejection with $u = 0$ is close to the chosen α risk, as expected. The frequency of rejection is very high with $u = 1/36$, and it is the maximum possible for the two greatest values of u. So, the validity test (20) is visibly efficient, and we can use it on real data.

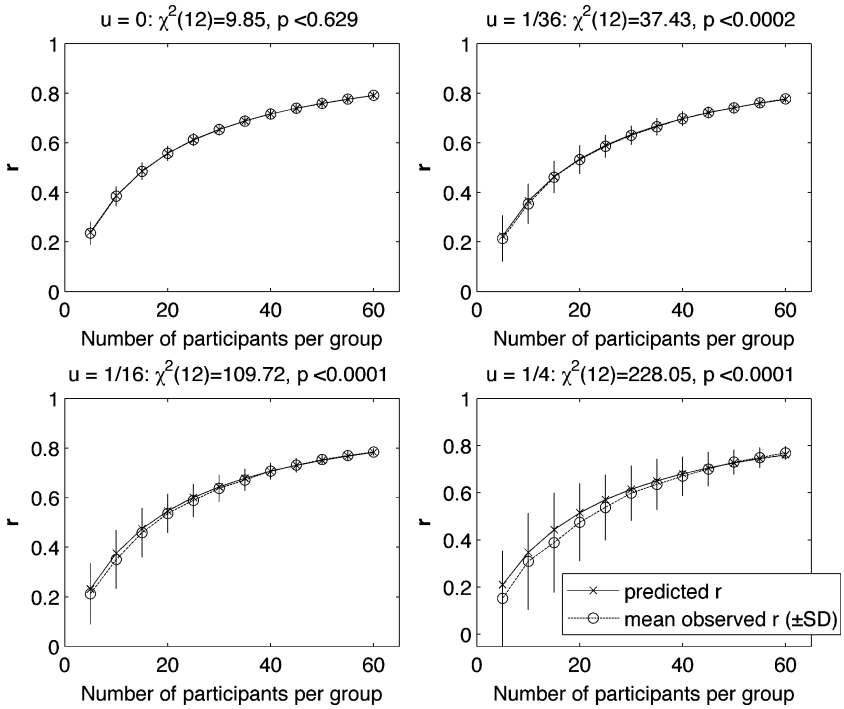


Fig. 1 Predicted and observed mean correlation values (with SD bars) as functions of the number of selected participants per group, in four artificial data sets (both with 360 items × 120 participants, $q = 1/16$), with different u ratios (0, 1/36, 1/16, 1/4). Predicted and observed functions are not significantly different for $u = 0$, but they are significantly different for all non-zero values of u, even in those cases where the difference of graphs is just visible

Table 1 Observed rejection frequencies of the null difference hypothesis for two α risks (.01 and .05) in the validity test applied to artificial data sets with different u ratios (0, 1/36, 1/16, 1/4). The null hypothesis is true in the case $u = 0$ only. In both cases, the data sets had 360 items × 120 participants, with $q = 1/16$, and 200 random data sets were tested for each u value

	u = 0	u = 1/36	u = 1/16	u = 1/4
$\alpha = 0.01$	0.020	0.885	1.000	1.000
$\alpha = 0.05$	0.040	0.960	1.000	1.000

Testing real response time databases

Tests on artificial data allowed us to be sure that our tools work suitably. Now, a crucial question is to know whether or not the proposed statistical model actually applies to real behavioural data. We examine this question hereafter on three real reaction time databases, involving two word reading tasks (word identification and word naming) and two languages (English and French).

Word identification times from Rey et al., (2009)

This database and methodology details are described in Rey et al., (2009). It is a set of 120 items by 140 participants' word identification times, with about 4% missing data. The stimuli were 120 monosyllabic, five-letter English printed words, randomly selected in the Celex lexical database (Baayen, Piepenbrock, & van Rijn, 1993). The used task was a standard perceptual identification in a luminance-increasing paradigm (Rey, Jacobs, Schmidt-Weigand, & Ziegler, 1998; Rey & Schiller, 2005). Participants were undergraduate students at Arizona State University, native English speakers, with normal or corrected-to-normal vision.

The data table was given as an argument to the Matlab function ECVT listed in Appendix A. The output provided an overall ICC equal to 0.9016, with a 99% confidence interval of [0.8655, 0.9315]. The correlation fit plot is shown in Fig. 2, and the test (20) is clearly non-significant [$\chi^2(14) = 8.62$, n.s.]. Thus, the correlation model (3) suitably accounts for these data, and the ICC above is a reliable expected correlation to test models.

English word naming time

Participants Ninety-four undergraduate students from Stanford University participated in the experiment. All participants were native English speakers with normal or corrected-to-normal vision.

Stimuli and apparatus Seven hundred seventy English disyllabic words randomly selected from the Celex Database were used. The words were four to eight letters long, without plural forms.

Procedure Each trial started with a fixation point that was presented for 500 ms on a PC computer screen. It was immediately followed by a word that appeared in the middle of the computer screen in font Courier 24. The word remained on the screen until the participant's response. Participants were instructed to read aloud the target word as quickly and accurately as possible. The interval between trials was 1,500 ms. Response times were measured from target onset to the participant's response. The experimenter sat behind the participant and recorded errors and voice key failures. The experiment started with a training session composed of ten trials. The experiment then started with test words presented in a randomized order for each participant with a break every 150 trials.

The resulting database is a set of 770 items by 94 participants' word naming times, with 3.61% missing data. The data table was given as argument to the Matlab function ECVT listed in Appendix A. The output provided an overall ICC equal to 0.9261, with a 99% confidence interval of [0.9160, 0.9355]. The correlation fit plot is shown in Fig. 3, and the test (20) is clearly non-

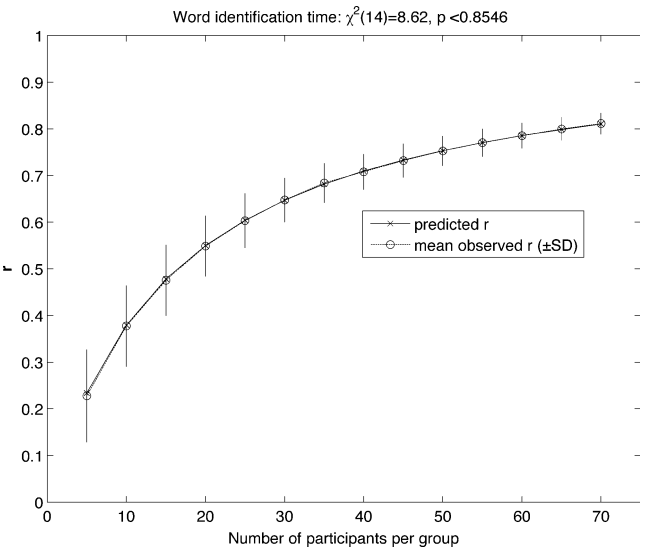


Fig. 2 Predicted and observed mean correlation values (with SD bars) as functions of the number of selected participants per group, in the English word identification time data set (120 words × 140 participants, 4% missing data), from Rey et al. (2009). The two functions are not significantly different [$\chi^2(14) = 8.62$, n.s.]. The overall ICC is equal to 0.9016, with a 99% confidence interval of [0.8655, 0.9315]

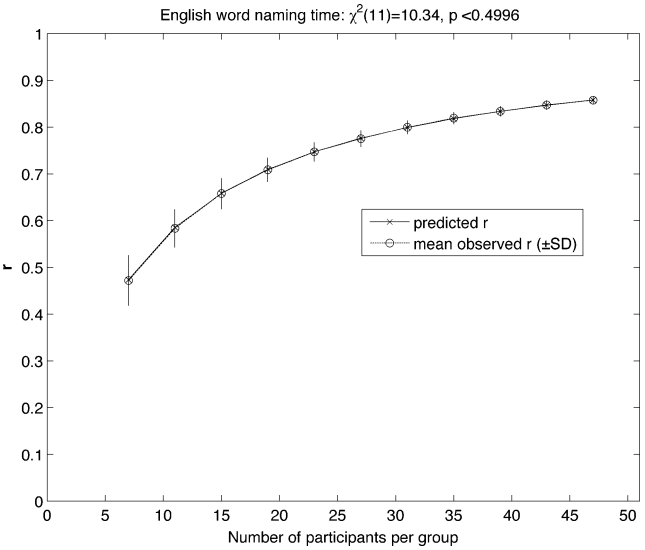


Fig. 3 Predicted and observed mean correlation values (with SD bars) as functions of the number of selected participants per group, in the English word naming time data set (770 words × 94 participants, 3.61% missing data). The two functions are not significantly different [$\chi^2(11) = 10.35$, n.s.]. The overall ICC is equal to 0.9261, with a 99% confidence interval of [0.9160, 0.9355]

significant [$\chi^2(11) = 10.35$, n.s.]. Thus, the correlation model (3) suitably accounts for these data, and the ICC above is a reliable expected correlation to test models. Details of the above analysis are listed in Appendix B as an example of use of the ECVT program, with helpful comments.

French word naming time

Participants One hundred undergraduate students from the University of Bourgogne participated in this experiment. All were native French speakers with normal or corrected-to-normal vision.

Stimuli A list of 615 French disyllabic words randomly selected from the Brulex Database (Content, Mousty & Radeau, 1990) was used. The selection was restricted to four-to-eight letter words and excluded verbs and plural forms.

Procedure The same procedure as the one in the English word naming experiment (English word naming time) was used.

The resulting database is a set of 615 items by 100 participants' word naming times, with 3.94% missing data. The data table was given as argument to the Matlab function ECVT listed in Appendix A. The output provided an overall ICC equal to 0.9578, with a 99% confidence interval of [0.9513, 0.9638]. The correlation fit plot is shown in Fig. 4, and the test (20) is clearly non-significant [$\chi^2(12) = 6.60$, n.s.]. Thus, the correlation model (3) suitably accounts for these data, and the ICC above is a reliable expected correlation to test models.

As a conclusion to "Testing real response time databases", we note that all the examined real data sets seem to fulfil the variable model (1) and the resulting correlation model (3). This is not a trivial result, since the generalized variable model (22), with a non-constant "participant sensitivity" to the item effect, can a priori seem more plausible. Fortunately, the obtained results show that very commonly used behavioural measures such as word identification and word naming times can be analysed in terms of the restrictive model (1), and thus, the methodology derived from (1) to test simulation or regression models can be applied to these variables.

Testing regression models on simulated data

Simulated Data

In order to build a test problem, one first chooses the number m of items, the number n of participants, and the exact number of parameters k_0 that the data-generating

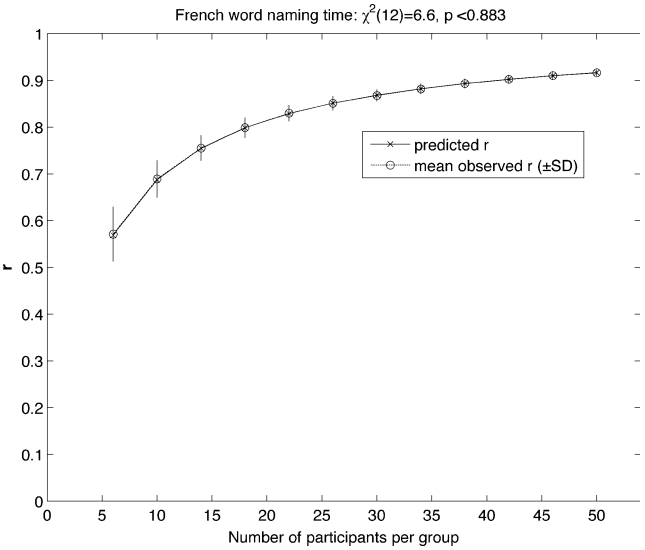


Fig. 4 Predicted and observed mean correlation values (with SD bars) as functions of the number of selected participants per group, in the French word naming time data set (615 words × 100 participants, 3.94 % missing data). The two functions are not significantly different [$\chi^2(12) = 6.60$, n.s.]. The overall ICC is equal to 0.9578, with a 99% confidence interval of [0.9513, 0.9638]

model will use for generating its regular part. One also chooses k_{max}, the maximum number of free parameters that tested models can use to fit the data. One must have the inequalities: $1 < k_0 < k_{max} \leq (k_0 + n) < m$. In addition, one chooses the noise standard deviation (σ_ε), which allows approximate control of the data q ratio.

One uses (1) to generate an m items by n participants data sample matrix (x_{ij}) in the following way:

$$x_{ij} = \mu' + \alpha_j + \beta_i + \varepsilon_{ij}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n.$$

The sample mean (μ') is a random constant. The sample participant effect (α_j) is a random Gaussian vector of length n, with zero mean and unit standard deviation. The sample item effect (β_i) is a random Gaussian vector of length m, with zero mean and unit standard deviation. The sample noise is a random Gaussian $m \times n$ matrix $E = (\varepsilon_{ij})$, with mean zero and standard deviation σ_ε. Each column of the noise matrix is orthogonal to the item effect vector; however, the noise matrix itself is not orthogonal.

In order to build a base for regression models of "predictor" type (with $c = 2$, see Model correlation), one first builds a $m \times k_0$ orthogonal matrix $H = (h_{ij})$ whose first column vector has m equal components ($1/m^{1/2}$), and the remaining k_0 - 1 columns are orthogonal Gaussian random vectors of length m, with zero means and unit norms, whose sum is proportional to the sample item effect, more precisely:

$$\frac{(m-1)^{1/2}}{(k_0-1)^{1/2}} \sum_{j=2}^{k_0} h_{ij} = \beta_i, \quad 1 \leq i \leq m.$$

A predictor with k degrees of freedom (free parameters) uses a base G_k made of the k first columns of H if $k \leq k_0$, to which one adds the k - k_0 first columns of E if $k > k_0$, so G_k is a $m \times k$ matrix, and the predictor parameters (w vector) are optimized as a least-squares solution of the equation $G_k w = x$, where $x = (x_i)$ is the mean item performance vector given by (17). Thus, one has $w = G_k^{\dagger} x$, and the predictor is $B_k = G_k w$. Observe that the exact predictor can be obtained only with $k = k_0$. If $k < k_0$, then the predictor under-fits the data. If $k > k_0$, then the predictor over-fits the data.

Under-fit and over-fit detection using the expected correlation

The method described in "Simulated Data" was used to build artificial problems with the parameter values $n = 40$, $k_0 = 20$, $k_{max} = 60$, $m \in \{61, 610\}$, and two levels of q approximately equal to 1/4 and 1/16, respectively. In each problem, 59 models whose complexity varied from 2 to 60

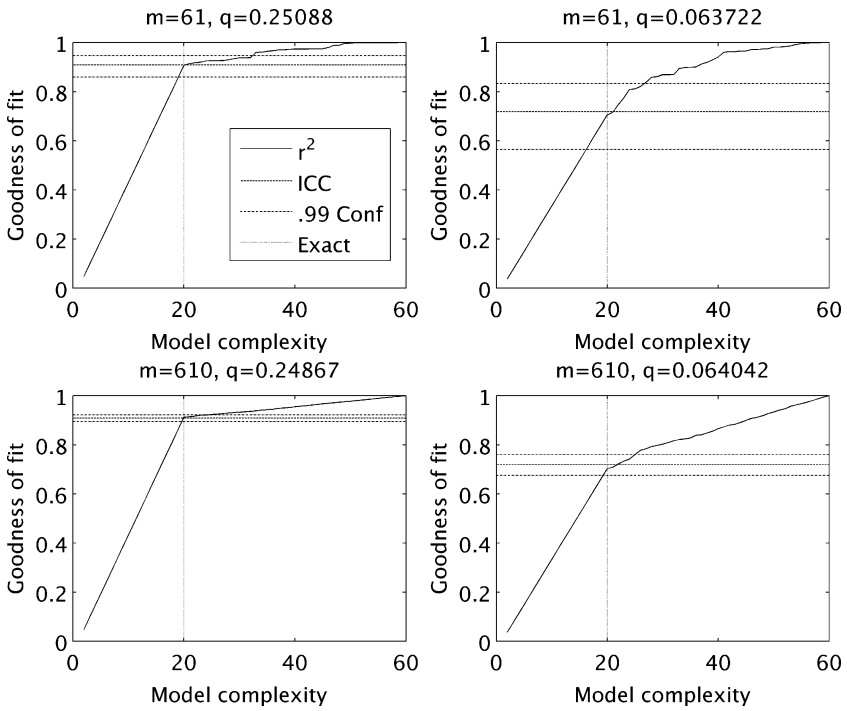


Fig. 5 Variation of r^2 and its intersection with the ICC 99% confidence interval, as a function of the number of free parameters used in least-squares fitted models, while original artificial data were generated by a model using exactly 20 parameters (plus random variables), with 61 or 610 items, 40 participants, and two levels of the q ratio. The r^2 values under the lower ICC confidence limit correspond to under-fitted models, while r^2 values above the upper ICC confidence limit correspond to over-fitted models

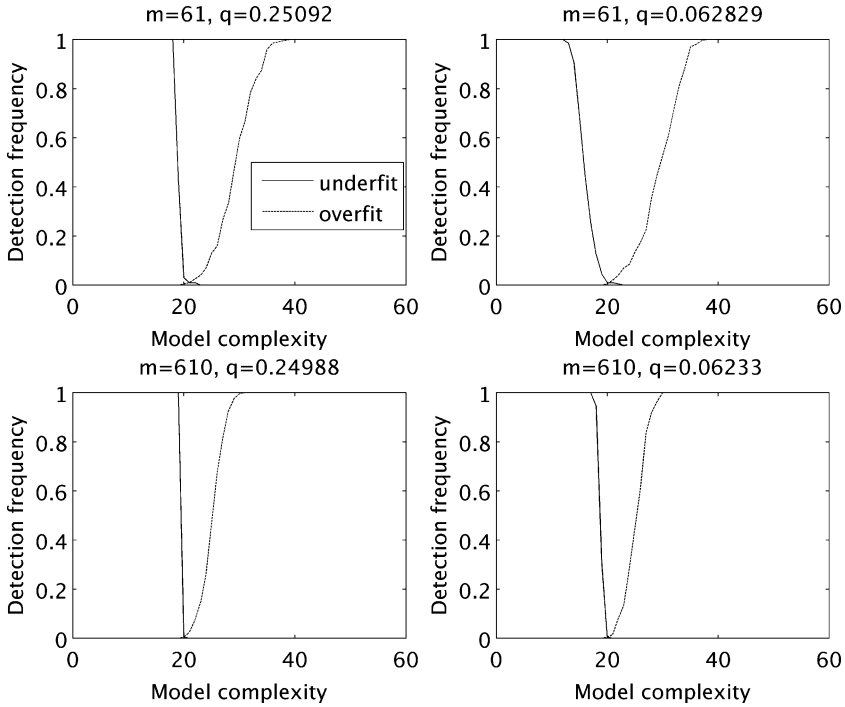


Fig. 6 Detection frequency of under-fits and over-fits by r^2 values outside the ICC 99% confidence interval, as functions of the number of model parameters (complexity) in experiments similar to those of Fig. 5, repeated 200 times each. The exact model complexity corresponds to 20 parameters

free parameters were fitted to the data by the least-squares method, and one computed the squared correlation of each model prediction vector with the data average vector. Figure 5 shows the variation of the squared correlation as a function of the model complexity for the two levels of m and of q. Also in each plot the expected correlation (ICC) and its 99% confidence interval are shown. Note that the squared correlation always intersects the ICC confidence interval in the neighbourhood of the exact complexity (20 parameters). Squared correlations under the lower confidence limit are detected as under-fits, and squared correlations above the upper confidence limit are detected as over-fits. The four experiments of Fig. 5 were repeated 200 times each, in order to observe the frequency of under-fit and over-fit detections as a function of model complexity. The results are shown in Fig. 6, where one can see that the minimum global frequency of misfit detections is always on a close neighbourhood of the exact complexity level (20 parameters). The under-fit detection frequency rapidly increases as the model complexity decreases from the optimum, while the over-fit detection frequency more gradually increases as the model complexity exceeds the optimum. The accuracy of misfit detections increases as m increases, and it is moderately sensitive to the q ratio. Table 2 shows the details of the frequency of abusive detections of under-fits and over-fits at the exact model complexity (20 parameters). For m = 610, this frequency is exactly the expected one, given the α risk (0.01). For m = 61, the misfit detection frequency is a bit greater than expected; however, the discrepancy is small enough to allow practical use, provided that one uses α risks not greater than 0.01.

Discussion and conclusion

We have shown that, provided that the considered behavioural variable fulfils the usual decomposition model (1), one can build a suitable reference correlation (or "expected

Table 2 Detail of the frequency of abusive detections of under-fits and over-fits at the exact model complexity (20 parameters), using the ICC 99% confidence interval, in the experiments of Fig. 6

	m = 61, q ≈ 1/4	m = 61, q ≈ 1/16	m = 610, q ≈ 1/4	m = 610, q ≈ 1/16
Under-fits	0.030	0.010	0.005	0.005
Over-fits	0.005	0.005	0.005	0.005
Total misfits	0.035	0.015	0.010	0.010

correlation") having the form of an intraclass correlation coefficient. The lower and upper confidence limits of this ICC can be considered as under-fitting and over-fitting limits, respectively, for model goodness-of-fit statistics, which are the absolute correlation (for full simulation models) or the squared correlation (for predictors) of model item performance predictions with empirical data averaged over participants. We demonstrated the effectiveness of this approach on artificial data that, by construction, fulfilled the variable decomposition model (1). In order to verify that any given data set fulfils model (1), and thus that the above methodology is suitable for these data, we proposed a test that is able to detect even weak deviances to this model. The performance of this test has been demonstrated on artificial data whose deviance to model (1) was gradually varied. Moreover, we tested real behavioural data sets in order to have an idea of the realism of model (1) and of the suitability of the derived methodology in practice. Three databases were tested: one set of English word identification times (from Rey et al., 2009), one new set of English word naming times, and one new set of French word naming times. It turned out that these three databases were compatible with model (1), demonstrating that the proposed methodology has a wide potential application field. Finally, the Matlab program "ECVT" listed in Appendix A allows Matlab users to directly apply this methodology, while it can also serve as an implementation model for developers on other platforms. In addition, Appendix B provides a commented example of use of the ECVT program with the data of "English word naming time".

As an ultimate illustration, let us consider two word reading model fits recently published in the literature. The models are (1) a multiple regression model with many predictors, which was used by Yap and Balota (2009) to predict word naming latencies, and (2) a simulation model (CDP++) published by Perry et al. (2010), which was used as a simple predictor for data similar to those of Yap and Balota, that is, a subset of the word naming latencies from the ELP database (Balota et al., 2007), corresponding to more than 6000 monomorphemic multisyllabic English words. On these data, Yap and Balota obtained a global fit of $R^2 = 0.612$, while Perry et al. obtained a global fit of $R^2 = 0.494$ after combining CDP++ predictions with usual phonological onset factors.

What can we say about the performance of these models? First, these are currently the best known fits for these two types of models on such data. But are these fits good? On one side, Yap and Balota's result seems better; however, a multiple regression model with many predictors always has an important risk of over-fitting. If one assumes that the ICC of the data set is about 0.5, for instance, then Perry et al.'s model could be the best. Unfortunately, one does not know the ICC of the data set used. However, it is possible to compute an estimation of its order of magnitude.

Firstly, one can reasonably assume that the items used in "English word naming time" (bisyllabic English words) are a random sample of items belonging to the same item population as those used to test the above models. Secondly, there is no reason to think that the participant populations are basically different (American college students). Thirdly, note that the expected ICC strongly depends on the number of participants (Eqs. 3–5), but not on the number of items. In fact, the number of items plays an important role only for the variance of ICC estimators, not for their expected magnitude. The remaining critical element is the q ratio (Eq. 4), which can vary depending on the conditions in which the data were collected (noise). So, we clearly take a risk assuming that the q ratio of the ELP database and the one of the database of "English word naming time" are comparable. With this caution in mind, we can attempt to approximate the ICC of Yap and Balota data using our q ratio for the English word naming times ($q = 0.1333$, see Appendix B) and applying Equation 5 with $n = 25$ participants (which is the number of observations per item for the naming data in ELP). Doing this, one obtains an ICC of about 0.769. Clearly, none of the above models reaches such a fitting level, indicating that the race for new reading models remains open. However, note also that a firm conclusion on this point cannot be drawn as long as one does not know the ICCs of data sets on which models were tested.

As a conclusion, it appears desirable to encourage the use of statistics like the ones presented in this paper, or possible equivalent, in order to allow researchers involved in modelling to have a clear idea of "how far they are from the truth" (the truth of the data of course!) when they test their models. Comparing the performance of various models is probably useful but clearly not sufficient. Having a quite precise idea of the distance from the target result is precious information that can considerably help modellers improving the models. If the fit is quite close to the data ICC, probably minor changes in the model or a simple parameter tuning are sufficient. If the fit is far from the ICC, more important changes are probably necessary. If the model over-fits the data, then one must reduce the number of degrees of freedom of the model. But without a reliable reference fit, such as the data ICC and its confidence limits, the target result is not defined.

Acknowledgments The authors would like to thank the three anonymous reviewers and the action editor, Dr. Ira Bernstein, for their helpful comments. Part of this study was funded by ERC Research Grant 230313.

Appendix A

Matlab code (version 7.5) of the ECVT program

```
function [qAV,icc,conf,r,Chi2,Chi2df,Chi2p,mitem] = ECVT(x,tf,miss,pconf)
% ----------------Expected Correlation Validity Test-----------------------
%                              input:
% x:  (m items) X (n participants) data table
% tf: title of correlation fit plot (default: tf = '' for no figure)
% miss:  numerical code for missing data in table x (default = inf)
% pconf: probabilities of ICC confidence intervals (def. [.95 .99 .999])
%                              output:
% qAV,icc: q ratio and intraclass correlation (ICC) of table x by ANOVA
% conf:  ICC confidence intervals ([probability lower upper])
% r: estimate of the ICC by Permutation Resampling and extrapolation
% Chi2,Chi2df,Chi2p: correlation validity test (Chi^2, d.f., p), where the
%    expected correlation (ICC) is not reliable if the test is significant.
% mitem: (m X 1) column vector of mean performance for each item.
% -------------------------------------------------------------------------

[m,n] = size(x);
if nargin<4, pconf=[0.95 0.99 0.999]; end
if nargin<3, miss=inf; end
if nargin<2, tf=''; end

% -------------Compute the ICC using Analysis of Variance------------------
ti = zeros(m,1); ni = ti; tj = zeros(1,n); nj = tj;
sx2 = 0;
for i = 1:m
    for j = 1:n
        if x(i,j) ~= miss
            ti(i,1) = ti(i,1)+x(i,j);
            ni(i,1) = ni(i,1)+1;
            tj(1,j) = tj(1,j)+x(i,j);
            nj(1,j) = nj(1,j)+1;
            sx2 = sx2 + x(i,j)^2;
        end
    end
end
```

```
mitem = ti./ni;
N = sum(ni); t = sum(ti); ss = sx2 - t^2/N;
ssi = sum(ti.^2./ni) - t^2/N;
ssj = sum((tj.^2./nj),2) - t^2/N;
ssij = ss - ssi - ssj;
dfi = m-1; dfj = n-1; dfij = N-1-dfi-dfj;
msi = ssi/dfi;
vij = ssij/dfij; vi = max(0,(msi-vij)/n);
qAV = vi/vij; icc = vi/(vi+vij/n); Fobs=msi/vij;          % Main statistics
Q1f=quantF(1-(1-pconf)/2,dfi,dfij);  % Compute the ICC confidence intervals
Q2f=quantF(1-(1-pconf)/2,dfij,dfi);
conf=zeros(length(pconf),3);
for i=1:length(pconf)
    conf(i,1)=pconf(i);
    conf(i,2)=1-Q1f(i)/Fobs;
    conf(i,3)=1-1./(Q2f(i)*Fobs);
end

% ---Validity test using Permutation Resampling for several group sizes----
T=500;                      % Resampling size
[ng0,ngstep,nng]=nppg(n,12); ngs=(1:nng)'*ngstep+ng0;  % Group sizes choice
nrsObs=zeros(nng,3);
for p=1:nng                 % Group size loop
ng=ngs(p);                  % Number of participants per group
rt = zeros(T,1);
for t = 1:T                 % Permutation Resampling loop

    ok = false;
  while ~ok
    xp = x(:,randperm(n)); % Random participant permutation
    ng1 = zeros(m,1); mg1=ng1; ng2=ng1; mg2=ng1;
    for i = 1:m
        for j = 1:ng                          % First group
            if xp(i,j) ~= miss
                ng1(i) = ng1(i)+1; mg1(i) = mg1(i)+xp(i,j);
            end
        end
        for j = (ng+1):(2*ng)                % Second group
```

```
            if xp(i,j) ~= miss
                ng2(i) = ng2(i)+1; mg2(i) = mg2(i)+xp(i,j);
            end
        end
    end
    if (min(ng1)>0) && (min(ng2)>0), ok = true; end
  end
  mg1 = mg1./ng1; mg2 = mg2./ng2;
  rr = corrcoef([mg1, mg2]); rt(t)=rr(1,2);
end
nrsObs(p,:) = [ng,mean(rt),std(rt)];   % [group size, average r, S.D. of r]
end
[q,Chi2]=minChi2(nrsObs,T);        % Estimate optimal q
Chi2df=nng;
Chi2p=1-probChi2(Chi2,Chi2df);     % Validity test type I error risk
rPred=qn2r(q,ngs);                 % Predicted r values for all group sizes
r=qn2r(q,n);                       % Extrapolate r for n participants
if ~strcmp(tf,'')
    tit=plotfit(nrsObs,rPred,Chi2,Chi2df,Chi2p,tf);         % Visualization
end
end

function x = quantF(p,d1,d2)
% F distribution quantiles
x = quantbeta(p,d1/2,d2/2);
x = x.*d2./((1-x).*d1);
end


function x = quantbeta(p,a,b)
% Beta distribution quantiles
tol=1e-6;
x0=zeros(size(p)); x1=ones(size(p));
x=0.5*(x0+x1); dp=betainc(x,a,b)-p;
while max(abs(dp(:)))>tol
    x0(dp<=0)=x(dp<=0); x1(dp>=0)=x(dp>=0);
    x=0.5*(x0+x1); dp=betainc(x,a,b)-p;
end
end
```

```
function [q,Chi2]=minChi2(nrs,T)
% Optimize q to minimize Chi^2 (Newton-Raphson method)
tol=1e-9; todo=true; count=0;
s2i=1./nrs(:,3).^2;
q0=s2i'*rn2q(nrs(:,2),nrs(:,1))/sum(s2i);
while todo && (count<30)
    r0=qn2r(q0,nrs(:,1));
    d= T*2*sum((r0/q0).*(nrs(:,2)-r0).*(r0-1)./nrs(:,3).^2);
d2=T*2*sum(((r0/q0).^2).*((r0-1).^2+2*(nrs(:,2)-r0).*(1-r0))./nrs(:,3).^2);
    dq=d/d2; q=q0-dq; count=count+1;
    if abs(dq)<abs(tol*q)
        todo=false;
    else
        q0=q;
    end
end
Chi2=T*sum(((nrs(:,2)-qn2r(q,nrs(:,1))).(nrs(:,3))).^2);
if count>=30, warning('Newton-Raphson method failed to converge'); end
end

function q = rn2q(r,n) % Provides q given r and n
q = r./(n.*(1-r));
end

function r = qn2r(q,n) % Provides r given q and n
r = (n.*q)./(n.*q+1);
end

function p = probChi2(x,df)
% Chi-square cumulative probability function
p=gammainc(x/2,df/2);
end

function [S0,S,K]=nppg(n,ek)
% Optimize the number of participants per group in about ek groups
maxng=fix(n/2);
if maxng<=ek
```

```
    S0=0; S=1; K=maxng;
else
    minerr=inf;
    for s=1:maxng
        k=fix(maxng/s);
        s0=maxng-s*k;
        err=s0+s*abs(k-ek);
        if err<minerr
            K=k; S=s; S0=s0;
            minerr=err;
        end
    end
end
end

function tit = plotfit(nrsObs,rPred,Chi2,Chi2df,Chi2p,tf)
% Plot the correlation fit
ns=nrsObs(:,1); r=nrsObs(:,2); s=nrsObs(:,3);
maxx=2*ns(length(ns))-ns(length(ns)-1);
Chi2=round(Chi2*100)/100; Chi2p=round(Chi2p*10000)/10000;
if Chi2p==0, Chi2p=0.0001; end
plot(ns,rPred,'-xk',ns,r(:,1),'--ok');
axis([0 maxx min(0,min(r-s)*1.05) 1]); set(gca,'FontSize',12);
xlabel('Number of participants per group','FontSize',12);
ylabel('r','FontSize',12,'FontWeight','bold');
legend('predicted r','mean observed r (±SD)','Location','Best');
hold on
for i=1:length(ns)
    plot([ns(i);ns(i)],[r(i)-s(i);r(i)+s(i)],'-k');
end
tit=strcat(tf,': \chi^2(',num2str(Chi2df),')=',num2str(Chi2),...
    ', p <',num2str(Chi2p));
title(tit,'FontSize',12);
hold off
end
```

Appendix B

Example of use of the ECVT program: analysis of the English word naming time database

>> [qAV,icc,conf,r,Chi2,Chi2df,Chi2p] = ECVT(English RT,'English word naming time',0)

Comment: the input argument EnglishRT is the 770 x 94 data table, the string argument 'English word naming time' is the title for the output figure (Fig. 3), and the input argument 0 is the code for missing data in the data table. We omit the last input argument (pconf) in order to obtain the three default confidence intervals of the ICC (95%, 99%, and 99.9%). We omit the last output argument (mitem) because we do not need the 770 average RT vector. Then, we obtain the output:

qAV = 0.1333

This is the q ratio as it is computed by the ANOVA

icc = 0.9261

This is the ICC as it is computed by the ANOVA

conf =

0.9500	0.9185	0.9334
0.9900	0.9160	0.9355
0.9990	0.9130	0.9379

These are the three default confidence intervals of the ICC (probability lower_limit upper_limit)

r = 0.9236

This is the ICC as it is estimated by permutation resampling and extrapolation

Chi^2 = 10.3450

This is the χ^2 test value (equation 20) for the data set

Chi^2df = 11

Number of degrees of freedom of χ^2

Chi^2p = 0.4996

Probability of χ^2 under the null hypothesis. Here, the test is not significant, which means that the data in the EnglishRT table fulfil the variable model (1), and the ICC is a valid reference to test item performance models.

>> print -r600 -dtiff English.tif

We save the correlation plot figure (Fig. 3), which appeared in a separate figure window.

References

Ahmed, M., & Shoukri, M. (2010). A Bayesian estimator of the intracluster correlation coefficient from correlated binary responses. *Journal of Data Science, 8*, 127–137.

Akaike, H. (1974). A new look at the statistical model identification. *I. E.E.E. Transactions on Automatic Control, AC, 19*, 716–723.

Baayen, R. H., Piepenbrock, R., & van Rijn, H. (1993). *The CELEX Lexical Database (CD-ROM). Linguistic Data Consortium*. Philadelphia, PA: University of Pennsylvania.

Balota, D. A., & Spieler, D. H. (1998). The utility of item-level analyses in model evaluation: A reply to Seidenberg and Plaut (1998). *Psychological Science, 9*, 238–240.

Balota, D. A., Yap, M. J., Cortese, M. J., Hutchison, K. A., Kessler, B., Loftis, B., et al. (2007). The English lexicon project. *Behavior Research Methods, 39*(3), 445–459.

Coltheart, M., Rastle, K., Perry, C., Langdon, R., & Ziegler, J. C. (2001). DRC: A dual route cascaded model of visual word recognition and reading aloud. *Psychological Review, 108*, 204–256.

Content, A., Mousty, P., & Radeau, M. (1990). Brulex: Une base de données lexicales informatisée pour le français écrit et parlé. *L'Année Psychologique, 90*, 551–566.

Ferrand, L., New, B., Brysbaert, M., Keuleers, E., Bonin, P., Méot, A., et al. (2010). The French Lexicon Project: Lexical decision data for 38, 840 French words and 38, 840 pseudowords. *Behavior Research Methods, 42*, 488–496.

Hannan, E. J., & Quinn, B. G. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society, B, 41*, 190–195.

Hansen, M. H., & Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association, 96*, 746–774.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association, 90*, 377–395.

McGraw, K. O., & Wong, S. P. (1996). Forming inferences about some intralass correlation coefficients. *Psychological Methods, 1* (1), 30–46.

Myung, I. J., Pitt, M. A., & Kim, W. (2005). Model evaluation, testing and selection. In K. Lamberts & R. Goldstone (Eds.), *Handbook of cognition* (pp. 422–436). Thousand Oaks, CA: Sage.

Opdyke, J. D. (2003). Fast permutation tests that maximize power under conventional Monte Carlo sampling for pairwise and multiple comparisons. *Journal of Modern Applied Statistical Methods, 2*, 27–49.

Perry, C., Ziegler, J. C., & Zorzi, M. (2007). Nested incremental modeling in the development of computational theories: The CDP+ model of reading aloud. *Psychological Review, 114*, 273–315.

Perry, C., Ziegler, J. C., & Zorzi, M. (2010). Beyond single syllables: Large-scale modeling of reading aloud with the Connectionist Dual Process (CDP++) model. *Cognitive Psychology*. doi:10.1016/j.cogpsych.2010.04.001

Pitt, M. A., & Myung, I. J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences, 6*(10), 421–425.

Plaut, D. C., McClelland, J. L., Seidenberg, M. S., & Patterson, K. (1996). Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review, 103*, 56–115.

Rey, A., Courrieu, P., Schmidt-Weigand, F., & Jacobs, A. M. (2009). Item performance in visual word recognition. *Psychonomic Bulletin & Review, 16*(3), 600–608.

Rey, A., Jacobs, A. M., Schmidt-Weigand, F., & Ziegler, J. C. (1998). A phoneme effect in visual word recognition. *Cognition, 68*, 41–50.

Rey, A., & Schiller, N. O. (2005). Graphemic complexity and multiple print-to-sound associations in visual word recognition. *Memory & Cognition, 33*(1), 76–85.

Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transactions on Information Theory, 42*, 40–47.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics, 6*, 461–464.

Seidenberg, M. S., & McClelland, J. L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review, 96*, 523–568.

Seidenberg, M., & Plaut, D. C. (1998). Evaluating word reading models at the item level: Matching the grain of theory and data. *Psychological Science, 9*, 234–237.

Shrout, P. E., & Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin, 86*(2), 420–428.

Spieler, D. H., & Balota, D. A. (1997). Bringing computational models of word naming down to the item level. *Psychological Science, 8*, 411–416.

Webb, N. M., Shavelson, R. J., & Haertel, E. H. (2006). Reliability coefficients and generalizability theory. *Handbook of Statistics, 26*, 81–124.

Yap, M. J., & Balota, D. A. (2009). Visual word recognition of multisyllabic words. *Journal of Memory and Language, 60*, 502–529.

frontiers in PSYCHOLOGY

GENERAL COMMENTARY
published: 24 November 2010
doi: 10.3389/fpsyg.2010.00200

Accounting for item variance in large-scale databases

Arnaud Rey* and Pierre Courrieu

Laboratoire de Psychologie Cognitive, Department of Psychology, National Center for Scientific Research, Provence University, Marseille, France
**Correspondence: arnaud.rey@univ-provence.fr*

A commentary on

Practice effects in large-scale visual word recognition studies: a lexical decision study on 14,000 Dutch mono- and disyllabic words and nonwords.
by Keuleers, E., Diependaele, K., and Brysbaert, M. (2010). Front. Psychol. 1:174. doi: 10.3389/fpsyg.2010.00174.

The Dutch Lexicon Project (DLP, Keuleers et al., 2010) is the third published database providing lexical decision times for a large number of items (after the ELP, Balota et al., 2007, and the FLP, Ferrand et al., 2010). In this commentary, we address the issue of the amount of item variance that models should *really* try to account for in the DLP (Spieler and Balota, 1997).

As noted by Seidenberg and Plaut (1998), to test the descriptive adequacy of simulation models with item-level databases, one needs to estimate the amount of error variance (i.e., sources of variance that are unspecific to item processing and that models cannot, in principle, capture) and, conversely, the amount of item variance that models should try to account for. One way to address this issue is to create independent groups of participants from a single database, and to compute the correlation between the item performances averaged over participants in each group (Courrieu et al., in press; Rey et al., 2009). One can show that the expected value of such correlations has the form of an intraclass correlation coefficient (ICC):

$$\rho = \frac{nq}{nq+1} \qquad (1)$$

where ρ is the ICC, n is the number of participants per group, and q is the ratio of the item related variance on the noise variance for the considered database (for more details, see Courrieu et al., in press or Rey et al., 2009).

As discussed in Courrieu et al. (in press), there are basically two methods for estimating ρ and q. The first one is based on a standard analysis of variance (ANOVA) of the database. This method is fast, accurate, and it provides suitable confidence limits for the ICC estimate. The other method is of Monte Carlo type. It is based on a permutation resampling procedure, which is computationally more demanding and more sensitive to missing data than the ANOVA method. However, this approach is distribution free and much more flexible than the ANOVA.

In order to apply these methods, the database needs to be available in the form of a $m \times n$ table, where m is the number of items, and n is the number of participants. The DLP database clearly fulfils this requirement, with $m = 14089$, and $n = 39$. The ELP and FLP databases are more problematic from this point of view because each participant provided data only for a subset of the whole set of items. A possible solution is to create "virtual" participants by mixing the data of various participants, previously transformed to z-scores (Faust et al., 1999), but this needs further investigations.

Fortunately, no such a problem occurs with the DLP database, however, the important proportion of missing data in this database (16%) prevents from applying the permutation resampling method. Nevertheless, an ANOVA based analysis provided an overall ICC equal to 0.8448, with a 99% confidence interval of (0.8386, 0.8510), indicating that this database contains about 84.5% of reproducible item variance[1]. A model that accounts for less than 83.86% of the empirical item variance probably underfits the data, while a model that accounts for more than 85.10% of the empirical item variance probably over-fits the data (in general because it uses too many free parameters). Of course, this estimation is task-dependent and language dependent. Using a different task, a different language, a different set of items (e.g., monosyllabic or disyllabic words), or a different population sample (e.g., older adults) might generate different outcomes.

[1]Note that the ICC is in the order of a squared correlation, therefore providing a direct estimate of the amount of reproducible variance (for a justification, see Courrieu et al., in press).

Because this analysis has already been applied to different large-scale databases using different experimental paradigms and different languages (i.e., a naming task with English and French disyllabic words, Courrieu et al., in press, and a perceptual identification task with English monosyllables, Rey et al., 2009), it is now possible to directly compare these results. Indeed, for each database, a different q ratio has been estimated and one can now plot the resulting evolution of the ICC as a function of the number of participants for each database (see **Figure 1**). This figure clearly shows that there are important variations across experimental paradigms and languages (or population samples, which is still a confounded factor in the present situation) and that these variations can be explicitly quantified. For example, to reach the same amount of reproducible variance obtained in the DLP database (i.e., 84.5% with 39 participants), one would need to have 90 participants in the English perceptual identification task from Rey et al. (2009).

To conclude, the purpose of the present commentary was to provide a precise estimate of the amount of reproducible variance that is present in the DLP database and to compare the evolution of the reproducible variance across tasks or languages. By providing this information, it is now possible to precisely test the descriptive adequacy of any model that could generate item-level predictions trying to account for item variance in the DLP database.

ACKNOWLEDGMENT

Part of this study was funded by ERC Research Grant 230313.

REFERENCES

Balota, D. A., Yap, M. J., Cortese, M. J., Hutchison, K. A., Kessler, B., Loftis, B., Neely, J. H., Nelson, D. L., Simpson, G. B., and Treiman, R. (2007). The English Lexicon Project. *Behav. Res. Methods* 39, 445–459.

Courrieu, P., Brand-D'Abrescia, M., Peereman, R., Spieler, D., and Rey, A. (in press). Validated intraclass correlation statistics to test item performance models. *Behav. Res. Methods* doi: 10.1007/s13428-010-0002–7. [Epub ahead of print].

Faust, M. E., Balota, D. A., Spieler, D. H., and Ferraro, F. R. (1999). Individual differences in information-processing

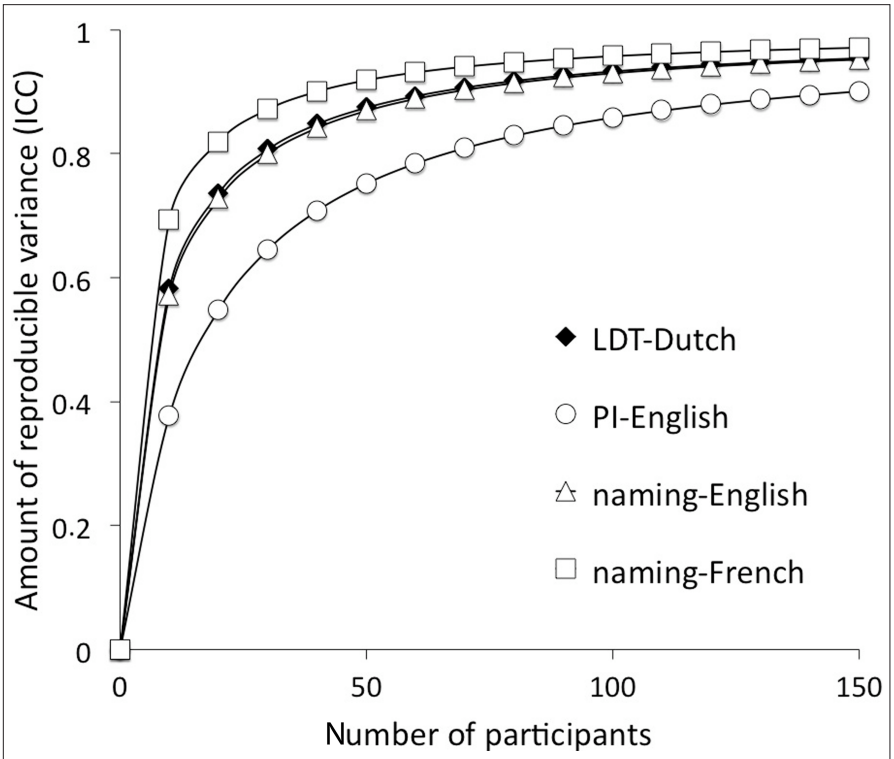


FIGURE 1 | Evolution of the amount of reproducible variance ICC as a function of the number of participants in four databases: the DLP (Lexical decision task in Dutch, LDT-Dutch), Rey et al. (2009; Perceptual identification task with English monosyllables, PI-English), Courrieu et al. (in press; Naming disyllables in English and French: naming-English + naming-French). For each of these databases, the estimated q parameter was respectively: 0.1396, 0.0607, 0.1333, 0.2269.

rate and amount: implications for group differences in response latency. *Psychol. Bull.* 125, 777–799.

Ferrand, L., New, B., Brysbaert, M., Keuleers, E., Bonin, P., Méot, A., Augustinova, M., and Pallier, C. (2010). The French Lexicon Project: lexical decision data for 38,840 French words and 38,840 pseudowords. *Behav. Res. Methods* 42, 488–496.

Keuleers, E., Diependaele, K., and Brysbaert, M. (2010). Practice effects in large-scale visual word recognition studies: a lexical decision study on 14,000 Dutch mono- and disyllabic words and nonwords. *Front. Psychol.* 1:174. doi: 10.3389/fpsyg.2010.00174.

Rey, A., Courrieu, P., Schmidt-Weigand, F., and Jacobs, A. M. (2009). Item performance in visual word recognition. *Psychon. Bull. Rev.* 16, 600–608.

Seidenberg, M., and Plaut, D. C. (1998). Evaluating word reading models at the item level: matching the grain of theory and data. *Psychol. Sci.* 9, 234–237.

Spieler, D. H., and Balota, D. (1997). Bringing computational models of word naming down to the item level. *Psychol. Sci.* 8, 411–416.

Received: 24 October 2010; accepted: 25 October 2010; published online: 24 November 2010.
Citation: Rey A and Courrieu P (2010) Accounting for item variance in large-scale databases. Front. Psychology ***1****:200. doi: 10.3389/fpsyg.2010.00200*
This article was submitted to Frontiers in Language Sciences, a specialty of Frontiers in Psychology.

II.G Travaux en cours et perspectives

Dans cette dernière section, je présente un projet de recherche axé sur trois thèmes principaux: l'approximation rapide des fonctions multivariées par l'humain, les techniques statistiques de validation de modèles, et enfin un modèle de lecture qui devrait être capable de lire des écritures manuscrites. Il s'agit en fait de trois projets à peu près indépendants, chacun ayant déjà produit un certain nombre de publications, ou faisant l'objet de publications en cours ou en préparation. Je décrirai aussi quelques perspectives à moyen terme associées à chacun des thèmes de recherche évoqués.

Approximation rapide des fonctions multivariées

L'apprentissage de fonctions chez l'humain a été très étudié, principalement dans les cas de fonctions booléennes multivariées ou de fonctions continues univariées. Il est cependant une capacité particulière dont nous faisons usage quotidiennement, et qui jusqu'à présent n'a fait l'objet d'aucune publication. Je veux parler de la faculté que nous avons d'estimer immédiatement (sans apprentissage) la valeur d'une fonction en un point quelconque du plan sur le base de valeurs fournies en d'autres points. Un exemple familier est celui des cartes de prévisions météorologiques où des températures sont prédites en un certain nombre de localités, mais souvent pas pour la localité qui nous intéresse. Dans ce cas, nous estimons la température probable dans cette localité en fonction des valeurs fournies pour des localités voisines. Autrement dit, nous réalisons une approximation rapide de fonction (température) en un point de généralisation d'un support à deux variables (longitude et latitude), à partir d'un échantillon fini de points de cette fonction (carte des prévisions).

J'ai eu la curiosité de tenter d'identifier les procédures que nous mettons en oeuvre pour résoudre de tels problèmes. Pour ce faire, j'ai soumis à des participants des problèmes artificiels du type décrit ci-dessus, et j'ai enregistré leurs réponses de généralisation (Courrieu, submitted-1). La figure 1 visualise quelques uns des ces problèmes (noir sur blanc) et les réponses de généralisation moyennes de 16 participants (blanc sur noir). Il se trouve que les réponses de généralisation des sujets présentent une certaine variabilité, mais sont cependant extrêmement consistantes puisque l'ICC des données est de 0.985, avec un intervalle de confiance à 99.9% de [0.959, 0.997] (Courrieu et al., 2011, voir Section II.F).

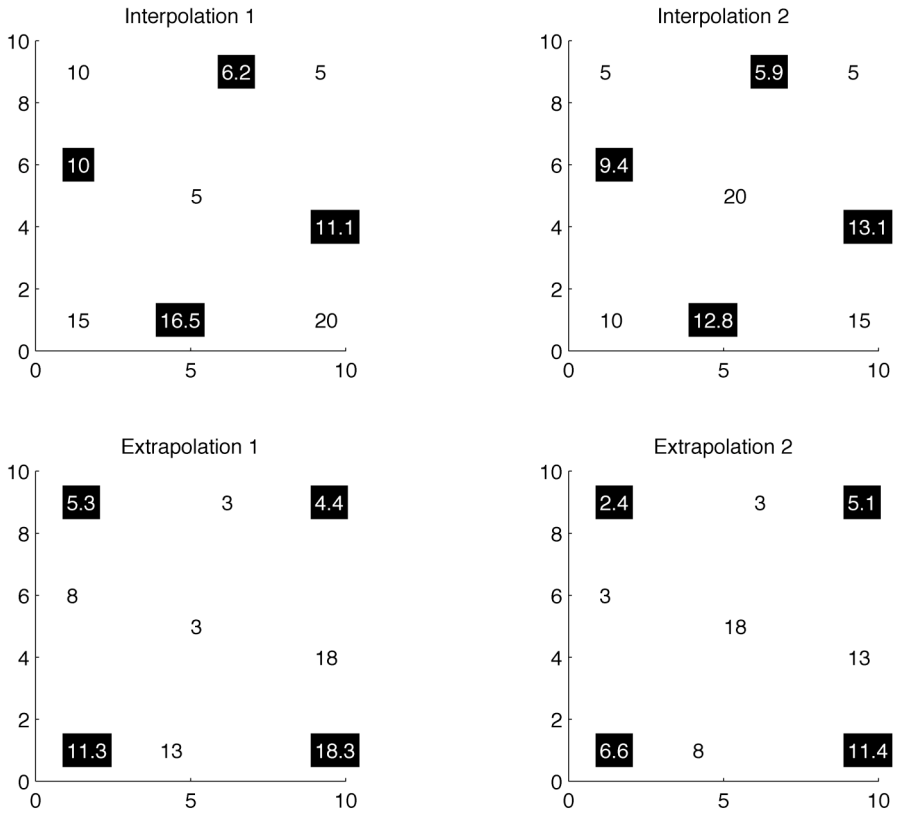


Figure 1. Visualisation de 16 problèmes d'approximation rapide de fonctions bivariées. Les données des problèmes sont en noir sur fond blanc, et les réponses moyennes de 16 sujets en des points de généralisation sont en blanc sur fond noir (Courrieu, submitted-1).

J'ai ensuite comparé à ces données les prédictions d'un échantillon représentatif de 10 modèles connus d'approximation des fonctions. Il se trouve que le meilleur prédicteur était le modèle de Courrieu (2005), présenté dans la section II.D, mais que sa performance restait tout de même très en dessous de l'ICC des données ($r^2 = 0.839$), ce qui indique clairement que ce modèle est inexact pour cette tâche, ainsi que tous les autres modèles testés. J'ai alors procédé à une analyse approfondie des réponses humaines, ce qui m'a permis de conclure que l'humain construit ses réponses de généralisation en combinant des approximations linéaires faites à partir de certaines paires de points ("bipoints") où les valeurs de la fonction sont données. Sur cette base, j'ai construit un nouveau modèle d'approximation rapide des fonctions, dénommé ABI (pour "Average of Bipoint Interpolations"), et ce modèle s'est avéré capable de prédire les réponses de généralisation humaines beaucoup plus précisément que tous les autres modèles testés. L'ajustement obtenu est $r^2 = 0.951$, ce qui est légèrement inférieur à la borne

inférieure de l'ICC des données (0.959), indiquant que le modèle est encore inexact, mais cela tient vraisemblablement à peu de choses.

La suite du projet consiste à tenter d'identifier et de corriger ce "peu de choses". L'une des pistes que je vais explorer consiste à tenir compte de certains biais d'estimation qui semblent influencer les réponses des sujets. Il est en effet possible que les approximations à partir de bipoints ne soient linéaires que lorsque les valeurs trouvées sont dans un certain intervalle délimité par les valeurs minimales et maximales de l'ensemble des valeurs fournies dans les données du problème. En dehors de cet intervalle, le sujet pourrait tenter de corriger son estimation de façon à ne pas trop dépasser les valeurs "vraisemblables", ce qui introduirait le biais supposé.

Validation de modèles

L'utilisation de la technique statistique de validation de modèles développée par Courrieu et al. (2011) m'a conduit à observer que les grandes bases de données comportementales contiennent en général une proportion assez élevée de données manquantes (jusqu'à 16%), résultant d'erreurs de réponse, d'incidents techniques, ou de données aberrantes ("outliers"). J'ai également observé que ces données manquantes biaisent de façon importante les statistiques de consistance des données (ICC), aussi bien que les statistiques d'ajustement des modèles (r^2). A la limite, une proportion trop importante de données manquantes empêche les tests utiles de fonctionner correctement. Il était donc urgent de trouver une parade à ce problème, et c'est ce que j'ai récemment fait en collaboration avec Arnaud Rey (Courrieu & Rey, submitted-2). Dans un premier temps, nous avons défini une statistique corrigée qui permet d'estimer très efficacement ce que serait l'ICC d'une base de données si aucune donnée n'était manquante, alors qu'une proportion importante des données manque. Munis de cet outil, nous avons également pu définir une statistique corrigée d'ajustement de modèle (r^2 corrigé) qui estime efficacement l'ajustement réel d'un modèle aux données. Enfin, nous avons défini un algorithme d'imputation des données manquantes, dénommé CRARI (pour "Column and Row Adjusted Random Imputation") qui permet de remplacer toutes les données manquantes par des estimations, de telle manière que les moyennes par item originales sont préservées, et que l'ICC de la base de données sous imputation peut être arbitrairement choisi. Il suffit alors de choisir pour ce dernier la valeur de l'ICC corrigé évoqué ci-dessus, et l'on obtient une base de données sous imputation possédant des

propriétés statistiques importantes qui seraient celles de la base de données originale si aucune donnée n'était manquante. Précisons que l'algorithme CRARI est une extension de l'algorithme d'imputation connu sous le nom de "Adjusted Random Imputation" (Chen, Rao, & Sitter, 2000), ce dernier ne donnant malheureusement pas des résultats satisfaisants en ce qui concerne l'ICC des bases de données sous imputation. La figure 2 présente une des nombreuses expériences que nous avons réalisées avec des données simulées pour tester l'ICC corrigé et l'algorithme CRARI. Dans ces expériences, on part d'une matrice de données initiale complète dont on calcule l'ICC exact, puis on dégrade progressivement la matrice en augmentant la proportion de données manquantes (aléatoirement dans cette expérience). Dans chaque cas, on calcule l'ICC de la matrice dégradée (courbe "missing"), puis l'ICC corrigé (courbe "estimate"), et enfin l'ICC de la matrice sous imputation des données manquantes par l'algorithme CRARI (courbe "imputed"). Ainsi qu'on peut le voir dans la figure 2, l'ICC de la matrice dégradée décroît rapidement lorsque la proportion de données manquantes augmente. En revanche, l'ICC corrigé oscille toujours dans un proche voisinage de l'ICC exact, et l'ICC de la matrice sous imputation des données manquantes est toujours égal à l'ICC corrigé.

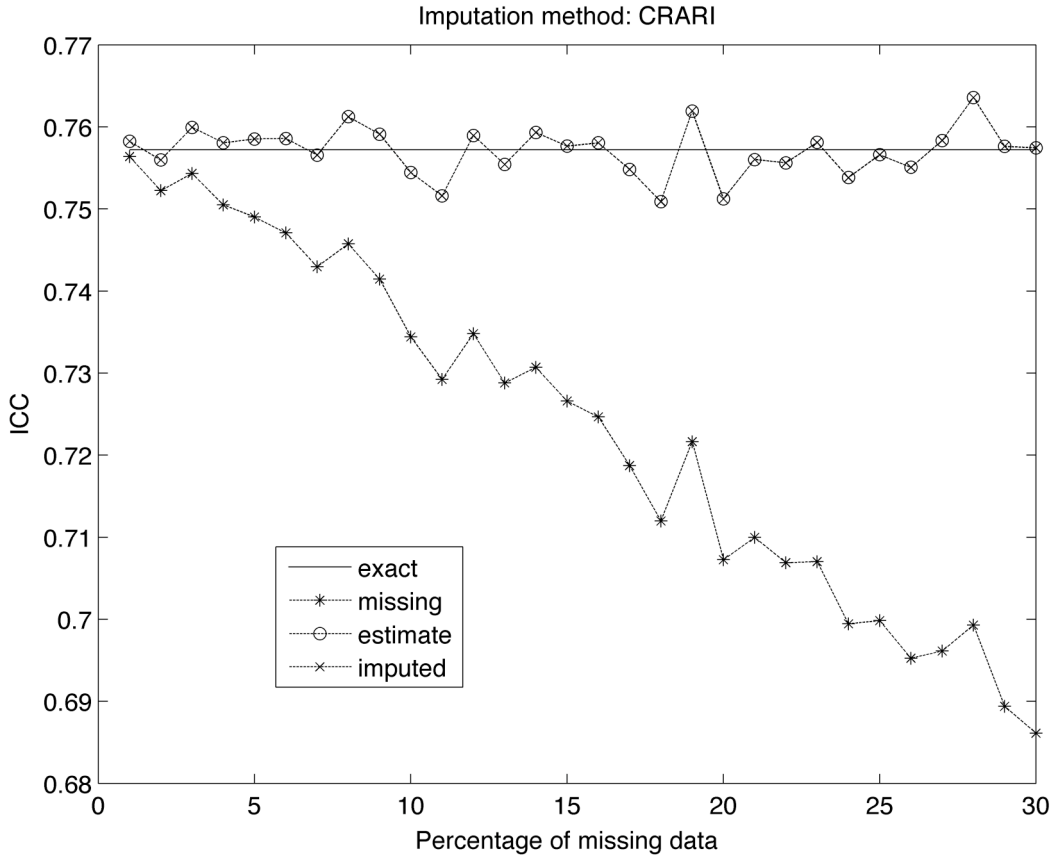


Figure 2. Test de l'ICC corrigé et de l'ICC sous imputation des données manquantes par l'algorithme CRARI sur des données simulées (Courrieu & Rey, submitted-2).

L'utilisation de cette méthodologie nous a permis de valider l'approche ICC de Courrieu et al. (2011) pour la base de données DLP (Keuleers, Diependaele, & Brysbaert, 2010) qui fournit des temps de décision lexicale pour plus de 14000 mots hollandais. La suite du projet consistera à appliquer cette méthodologie à d'autres grandes bases de données comme ELP et FLP, en résolvant les problèmes particuliers résultant du caractère incomplet du plan de recueil des données propre à ces bases de données. Nous envisageons également de développer un site internet permettant aux utilisateurs de calculer les statistiques de leurs propres bases de données et de tester leurs modèles en ligne, mais ce projet est subordonné à l'obtention de ressources informatiques appropriées.

Modèle de lecture

Il s'agit d'un projet de longue haleine sur lequel je travaille depuis plusieurs années, et qui a déjà fourni un certain nombre de résultats intermédiaires, lesquels ont leur intérêt propre dans la mesure où ils peuvent s'appliquer dans d'autres cadres que celui de ce projet, et ont donc donné lieu à un certain nombre de publications. On sait depuis longtemps que la lecture de mots imprimés passe par l'identification des lettres qu'ils contiennent (McClelland, 1976), mais ce point de vue strictement analytique doit être nuancé car certaines observations indiquent qu'une forme de traitement global ("holistique") des mots pourrait également contribuer à leur reconnaissance (Allen & Emerson, 1991; Lété & Pynte, 2003). Ceci est encore plus vraisemblable lorsqu'on considère la lecture de mots manuscrits, dans lesquels il n'est pas rare qu'un certain nombre de lettres ne soient tout simplement pas identifiables individuellement. Cependant, même dans le cas du manuscrit, nous restons capables de lire des mots inconnus ou mal orthographiés pour lesquels il ne peut s'agir de reconnaissance à proprement parler. Mon projet était donc d'essayer de rendre compte de ces capacités, aux apparences un peu contradictoires, par l'hypothèse suivante. Supposons que l'espace d'entrée est un espace de formes convenablement codées (ex. Courrieu, 2006, 2007), et que nous disposons d'un "dictionnaire" de formes correspondant à des mots (manuscrits ou imprimés) dont nous connaissons par ailleurs l'orthographe. Nous pouvons alors comparer toute nouvelle forme donnée en entrée aux formes contenues dans notre dictionnaire de formes, et calculer une mesure de similitude de la nouvelle entrée à chaque forme répertoriée. Suivant un principe similaire à celui des réseaux à fonctions bases radiales, et surtout de la généralisation que j'en ai proposée dans ce but (Courrieu, 2005), on peut construire, par apprentissage

supervisé sur l'espace de ces mesures de similitude, une fonction dont l'espace de sortie est un espace de codes orthographiques. Une fois construite, cette fonction permettrait d'associer à toute nouvelle forme d'entrée un code orthographique de sortie par généralisation. Il ne s'agit pas de "reconnaître" des mots, mais plutôt d'inférer une forme orthographique (éventuellement inconnue) à partir d'une forme visuelle, par référence à des formes connues d'orthographe connue. Les théorèmes d'approximation des fonctions disponibles garantissent la possibilité d'obtenir une solution convenable dans tous les cas, mais le véritable problème est la complexité potentielle de cette solution, et donc la possibilité de la réaliser en pratique.

Afin d'évaluer la faisabilité de ce projet, j'ai dans un premier temps téléchargé la base de données IAM (Marti, & Bunke. 2002), qui fournit 82227 images numérisées correspondant à 10841 mots anglais distincts, manuscrits par environ 400 participants. J'ai défini l'espace d'entrée comme l'espace des codes de densité des images (Courrieu, 2006, 2007), muni d'une fonction de similitude des codes capable de réduire des transformation régulières de complexité choisie. J'ai défini l'espace de sortie comme un espace de codes orthographiques numériques possédant un certain nombre de propriétés nécessaires pour l'application (vecteur de dimension fixe, décodable, et résistant aux erreurs), muni d'une fonction de similitude des codes appropriée. Ces codes orthographiques sont une variante simple du "codage spatial" de Davis (2010). Préalablement à l'application de la procédure d'apprentissage supervisé, qui risquait d'être lourde, j'ai préféré tester la compatibilité des topologies des espaces d'entrée et de sortie pour différentes fonctions de similitude des formes. Pour ce faire, j'ai échantillonné dans la base IAM un millier de paires d'images distinctes correspondant pour un tiers à des mots identiques, pour un tiers à des mots différents mais orthographiquement proches, et pour un tiers à des mots bien différents. J'ai ensuite calculé la corrélation entre les similitudes des formes et les similitudes des codes orthographiques correspondants, ce qui est une façon d'estimer la compatibilité des topologies d'entrée et de sortie. J'ai obtenu une corrélation de 0.50 dans le meilleur des cas, ce qui est loin d'être négligeable, mais n'est cependant pas suffisant pour générer un réseau efficace de complexité raisonnable. J'ai de plus fait l'observation quelque peu déconcertante suivante: la corrélation diminue lorsqu'on augmente la complexité des transformations que la fonction de similitude réduit, et il en va de même si l'on utilise des codes de densité plus "flexibles" (que j'ai élaborés au passage). La raison est que la flexibilité des codes ou des fonctions de similitude profite en moyenne plus aux paires de mots différents, en réduisant leur écart de forme, qu'aux paires de mots semblables. La seule transformation qui améliore la corrélation est un simple centrage des codes de densité

sur les médianes des distributions marginales. Il est donc clair qu'il faut rechercher une variante plus efficace du modèle, ce qui est l'objet du projet suivant.

Ayant noté qu'il n'est pas pertinent de réduire des transformations régulières dans l'évaluation des similitudes de formes, on peut envisager de ne pas réduire les transformations et d'intégrer la variabilité des formes dans les données d'apprentissage. Toutefois, si l'on utilise le principe du dictionnaire de formes tel qu'il est décrit plus haut, on va évidemment obtenir un réseau de taille considérable car il faudra intégrer au dictionnaire de nombreuses variantes de chaque forme de référence. Une solution à ce problème consiste à associer une unité du dictionnaire à chaque mot appris, et non à chaque forme visuelle, puis à entraîner chacune de ces unités à répondre à chaque forme présentée en entrée par une approximation de la "distance" séparant les codes orthographiques du mot correspondant à l'entrée courante et du mot représenté par l'unité. Ceci est possible car les codes de densité représentant des formes sont des codes numériques de grande dimension, et sont de plus arbitrairement extensibles en des polynômes qui possèdent la capacité d'approcher uniformément toute fonction continue. Les polynômes étant des formes linéaires en leurs coefficients, on peut tout simplement utiliser des "neurones linéaires", avec les techniques d'apprentissage élémentaires qui leurs sont associées, pour approcher la fonction "distance orthographique" au mot de référence de l'unité considérée, pour chaque forme d'entrée. A noter que même si la distance orthographique utilisée lors de l'apprentissage est une métrique sur l'espace des codes orthographiques, les réponses de généralisation obtenues ensuite (pour des formes inconnues) ne seront pas forcément des valeurs exactes de cette métrique. Comme certaines valeurs de généralisation peuvent être négatives, les neurones linéaires doivent comporter une fonction de sortie qui annule les valeurs négatives (i.e. un seuil de réponse à zéro), mais ceci n'intervient pas dans l'apprentissage. Fort heureusement, le modèle neurocomputationnel utilisé pour l'étage suivant (Courrieu, 2005) admet, comme entrées, des mesures de (dis)similarité sans exiger qu'elles aient les propriétés d'une métrique, de sorte que ce modèle peut approcher les codes orthographiques de sortie sur la base des approximations de distances orthographiques fournies par les neurones linéaires. A partir de là, tout se passe comme dans le modèle initial, à ceci près que la topologie induite sur l'espace des codes de densité par les sorties de la couche de neurones linéaires est une approximation de la topologie de l'espace de sortie, ce qui permet d'espérer de bonnes performances de généralisation avec une complexité raisonnable du réseau. La figure 3 présente un résumé de l'architecture du modèle que je propose.

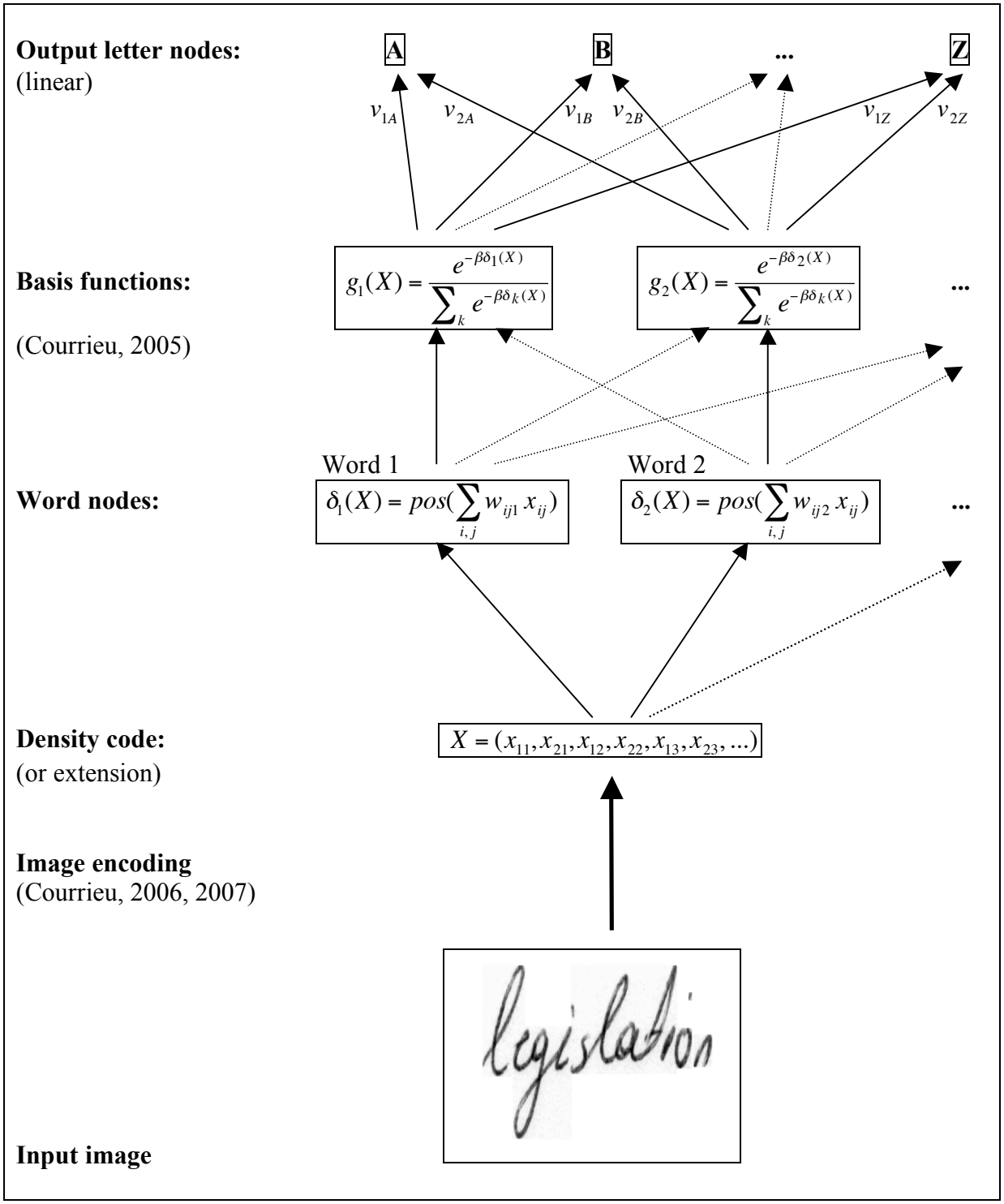


Figure 3. Architecture d'un réseau d'approximation de codes orthographiques (variante du "codage spatial" de Davis, 2010) à partir d'images de mots (ou pseudo-mots) manuscrits. Les poids synaptiques des noeuds "mots" (w_{ijk}), ainsi que ceux des noeuds "lettres" (v_{kL}) sont appris suivant une méthode de moindres carrés simple.

Il reste cependant à vérifier que cette approche donne effectivement des résultats satisfaisants en pratique, ce qui nécessite des calculs assez lourds qui sont en cours de programmation. En cas de succès, la faisabilité de ce principe sera démontrée, et je pourrai envisager de tester les prédictions du modèle par comparaison aux performances humaines.

Pour conclure

En résumé, le projet ci-dessus comporte trois axes thématiques principaux:
1- "L'approximation rapide des fonctions multivariées par l'humain", projet qui est déjà bien avancé et a produit un article actuellement soumis pour publication.
2- "Techniques statistiques de validation de modèles", projet aussi bien avancé qui a produit trois publications (section II.F), ainsi qu'un article actuellement soumis pour publication.
3- "Modèle de lecture", projet de longue haleine qui a déjà produit trois publications présentées dans les sections II.C et II.D, ainsi que l'une des publications techniques présentées dans la section II.E (Courrieu, 2009). Remarquons que le modèle de lecture proposé peut aisément être étendu à d'autres familles de formes, telles que celles qui sont présentes dans des scènes visuelles naturelles, ce qui est un autre développement à envisager.

Pour chacun des trois axes thématiques, le projet comporte une suite assez précisément définie, mais on n'évitera probablement pas l'irruption de question imprévues (notamment techniques) qui se posent habituellement en cours de route et sont également sources de résultats pouvant présenter un intérêt propre.

Ce projet, comme le reste de mon travail, est principalement centré sur la modélisation numérique et sur la validation des modèles. L'expérimentation humaine en est le complément naturel dès l'instant que les modèles sont prêts à produire des prédictions vérifiables, et cela fait aussi partie de la suite de mon programme.

Références

Allen, P.A., & Emerson, P.L. (1991). Holism revisited: evidence for parallel independent word-level and letter-level processors during word recognition. *Journal of Experimental Psychology: Human Perception and Performance, 17*, 489-511.

Chen, J., Rao, J.N.K., & Sitter, R.R. (2000). Efficient random imputation for missing data in complex surveys. *Statistica Sinica, 10*, 1153-1169.

Courrieu, P. (submitted-1). *Quick Approximation of Bivariate Functions*.

Courrieu, P. (2005). Function approximation on non-Euclidean spaces. *Neural Networks, 18*, 91-102.

Courrieu, P. (2006). Density codes and shape spaces. *Neural Networks, 19*, 429-445.

Courrieu, P. (2007). Fast density codes for image data. *Neural Information Processing - Letters and Reviews, 11(12)*, 247-255.

Courrieu, P., Brand-D'Abrescia, M., Peereman, R., Spieler, D., & Rey, A. (2011). Validated intraclass correlation statistics to test item performance models. *Behavior Research Methods*, DOI: 10.1007/s13428-010-0002-7.

Courrieu, P., & Rey, A. (submitted-2). *Missing Data Imputation and Corrected Statistics for Large-Scale Behavioral Databases*.

Davis, C.J. (2010). The spatial coding model of visual word identification. *Psychological Review, 117(3)*, 713-758.

Keuleers, E., Diependaele, K., & Brysbaert, M. (2010). Practice effects in large-scale visual word recognition studies: A lexical decision study on 14,000 Dutch mono- and disyllabic words and nonwords.. *Frontiers in Psychology* 1:174. doi:10.3389/fpsyg.2010.00174

Lété, B., & Pynte, J. (2003). Word-shape and word-lexical-frequency effects in lexical-decision and naming tasks. *Visual Cognition, 10*, 913-948.

Marti, U., & Bunke. H. (2002). The IAM-database: an English sentence database for off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, 5, 39 - 46.

McClelland, J.L. (1976). Preliminary letter identification in the perception of words and nonwords. *Journal of Experimental Psychology: Human Perception and Performance, 2(1)*, 80-91.