Time series recovery and prediction with regression-enhanced nonnegative matrix factorization applied to electricity consumption

Jiali Mei

NNT : 2017SACLS578

Time series recovery and prediction with regression-enhanced nonnegative matrix factorization applied to electricity consumption

Thèse de doctorat de l'**Université Paris-Saclay**
préparée à **Université Paris-Sud**

École doctorale n°574 mathématiques Hadamard (EDMH)
Spécialité de doctorat : **Mathématiques appliquées**

Thèse présentée et soutenue à Orsay, le 20 décembre 2017, par

Jiali Mei

Composition du Jury :

Mathilde Mougeot Professeur, ENSIIE et Université Paris Diderot	Présidente du jury
Pierre Pinson Professeur, Technical University of Denmark	Rapporteur
François Glineur Professeur, Université Catholique de Louvain	Rapporteur
Sylvain Arlot Professeur, Université Paris-Sud	Examinateur
Yohann De Castro Maitre de Conférence, Université Paris-Sud	Directeur de thèse
Georges Hébrail Chercheur sénior, EDF Lab Paris-Saclay	Co-Directeur de thèse
Yannig Goude Chercheur, EDF Lab Paris-Saclay et Université Paris-Sud	Invité co-encadrant
Jean-Marc Azaïs Professeur, Université de Toulouse Paul Sabatier	Invité co-encadrant

Thèse de doctorat

As in manufacture so in science—retooling is an extravagance to be reserved for the occasion that demands it. The significance of crises is the indication they provide that an occasion for retooling has arrived.

Thomas Kunh
The Structure of Scientific Revolutions

Remerciement

Je remercie avant tout mes encadrants : mon directeur de thèse, Yohann De Castro de l'Université Paris-Sud, mes encadrants à EDF, Georges Hébrail et Yannig Goude, et mon co-encadrant de thèse, Jean-Marc Azaïs de l'Université Paul-Sabatier de Toulouse. Je serai toujours reconnaissante de leurs précieux conseils et encouragements. J'ai eu une chance incroyable d'avoir travaillé pendant quatre ans sur un sujet qui est à la fois stimulant et a des potentiels d'applications importants. Merci à Georges de m'avoir proposé un sujet avec tant d'intérêt. Merci à Yohann pour ses visions qui ont permi de définir ce projet. Merci à Yannig et à Jean-Marc pour m'avoir suggéré des directions pragmatiques quand j'étais bloquée, et quand j'étais submergée par des difficultés. Je leur remercie aussi tous les quatre pour la patience qu'ils ont eu pour relire les rapports et les manuscripts à maintes reprises, et pour trouver les coquilles grammaticales, les fautes d'orthographes, mais aussi des trous de logique et des incohérences de définition, et de suggérer des corrections avec une telle gentillesse. Grâce à cette thèse, grâce à eux, j'ai pu devenir plus persévérante que je n'avait jamais pensé être.

Je tiens à remercier Pierre Pinson et François Glineur d'accepter de rapporter la thèse. Merci à Pierre pour sa rigueur, d'avoir donner des remarques et suggestions constructives qui soit aussi détaillées. Je remercie également Mathilde Mougeot et Sylvain Arlot d'accepter d'être membres du jury.

Je pense aussi à tout le monde que j'ai croisé à EDF et au LMO. Merci à tous les collègues de l'équippe E72 d'EDF R&D pour leur amitié : Anne pour les moments précieux d'échange des milles choses qui peuvent mal tourner pendant une thèse doctorale et une session `R`, Manel pour le rire et pour son balcon parisien incroyable, Bérénice pour son bon humeur, Laurent pour son super-pouvoir à identifier le système de projection cartographique (en particulier WGS 84) et son bon humeur, Méryl pour nous avoir poussés à aller à la salle de sport, ainsi qu'Amandine, Irina, Alzennyr, Guillaume, Lou, Benoît, Asma, Marie-Luce, Jean-Pierre Isabelle... Sans eux, sans les bo-buns et les verres partagés au Diamant, ces quatre ans auraient été beaucoup moins agréables. Merci à Maud pour son soutien et son aide depuis notre tout premier entretien. Je doit aussi dire merci à Raphaël de m'avoir toléré quand je monopolise le serveur de calcul. Merci à mes interlocuteurs à Enedis, qui m'ont aidé tant que possible sur les données. Je remercie France, Danièle pour leur patience pour la multitude de mes problèmes administratifs. Je remercie aussi toutes les personnes que j'ai croisées au LMO et à l'Ecole Doctorale. Même si je n'ai pas passé beaucoup de temps à Orsay, j'ai toujours été bien accueillie et aidée quand j'y étais.

Etant étrangère dans un pays, la vie n'aurait pas été simple même sans la charge d'une thèse. Ainsi, je dois un merci à mes amis qui ont toujours été là : à Maike et à Lingyi pour leur messages de compathie à tout moment, à Lei et Lingjie pour leur enthousiasme à chaque fois que j'ai un petit faible pour la cuisine asiatique. Merci à la famille Raison, pour leur gentillesse et leur affection.

我还要特别感谢我的父母。他们不仅辛勤努力，为我创造了优越的生活环境，而且教会了我如何思考，如何工作。

Enfin, merci à Martin, mon ami, pour son soutien et son amour, d'abord à distance, et puis juste là à côté de moi.

Contents

List of Figures

List of Tables

List of publications and main activities

Articles in proceedings of international conferences

[1] Jiali Mei, Yannig Goude, Georges Hebrail, and Nicholas Kong. "Spatial Estimation of Electricity Consumption Using Socio-Demographic Information". In: *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*. Oct. 2016, pp. 753–757. DOI: 10.1109/APPEEC.2016.7779596 *Cited on pages 12, 43.*

[2] Jiali Mei, Yohann De Castro, Yannig Goude, and Georges Hébrail. "Nonnegative Matrix Factorization for Time Series Recovery From a Few Temporal Aggregates". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 6–11 Aug 2017, pp. 2382–2390 *Cited on pages 67, 82, 85, 86, 95.*

Submitted article

[3] Jiali Mei, Yohann De Castro, Yannig Goude, Georges Hébrail, and Jean-Marc Azaïs. "Nonnegative Matrix Factorization with Side Information for Time Series Recovery and Prediction". In: *submitted* (2017) *Cited on page 81.*

Talks

Jiali Mei. In: *Jounées de Méthodologie Statistique de l'INSEE*. Paris, May 2015.

Jiali Mei. In: *Journée Des Doctorants En Proba-Stat de LMO*. Orsay, June 2016.

Jiali Mei. In: *Paris Machine Learning Meetup*. Paris, Dec. 2016.

Jiali Mei. In: *Rencontres Des Jeunes Statisticiens de La SFdS*. Porquerolles, Apr. 2017.

Patent

Georges Hebrail, Jiali Mei, Castro Yohann De, Yannig Goude, and Jean-Marc Azais. "Gestion D'un Approvisionnement Pour Des Consommations Se Produisant En Une Multiplicite De Sites". FR3046280 (A1). CIB: G01D4/00; G06Q50/06. June 2017.

Chapter

1 Résumé en français

Contents

1.1 Motivation industrielle : estimation et prédiction de consommation électrique

Pour assurer la sécurité et la stabilité des réseaux électriques, l'électricité injectée dans les réseaux doit correspondre à la demande à tout moment. C'est la responsabilité des entreprises d'électricité comme Électricité de France (EDF) de gérer cette équilibre offre-demande. La demande d'électricité dépend essentiellement du portefeuille de clients et des conditions extérieures, comme par exemple la météorologie, sur lesquelles les entreprises d'énergie ont peu de marge de manoeuvre. Or, l'électricité est une forme d'énergie qui peut difficilement se stocker. Ainsi les entreprises d'électricité gèrent l'équilibre offre-demande en ajustant la production d'électricité et la configuration des réseaux, selon les estimations et prédictions de demande en temps réel.

On peut donc comprendre pourquoi les données de consommation électrique sont importantes pour une entreprise comme EDF. Pour la planification de production nucléaire et la planification des réseaux, il est important d'avoir des prévisions de demande de moyen ou long terme (à partir de 6 mois). Pour la planification des opérations réseaux, il est important d'avoir des prévisions de moyen ou court terme, par exemple des prévisions horaires ou quotidiennes avec un horizon entre un jour et quelques mois. Pour profiter de la production décentralisée d'énergie en plein développement (par exemple photovoltaïque), il est intéressant d'avoir des données ou prévisions sur une échelle spatiale assez fine.

Outre ces problèmes dans le domaine du management d'énergie, les données de consommation électrique sont utiles pour beaucoup d'autres applications. Dans le contexte de marché ouvert

d'électricité, les données de consommation peuvent être utilisées par les opérateurs de réseau de transport (RTE en France) et de distribution pour mesurer les consommations/productions des portefeuilles des différents responsables d'équilibre tels les producteurs d'électricité, les gros consommateurs, les agrégateurs d'effacement ou les traders.

L'électricité est un aspect important des activités socio-économiques. Ainsi les données de consommation électrique sont une information intéressante en soi. Elles peuvent être mises en parallèle avec d'autres statistiques publiques sur la société. Si la consommation électrique est connue à la même échelle spatio-temporelle que d'autres informations socio-démographiques, croiser ces données peut permettre

- d'une part, aux entreprises d'électricité de fournir un service de manière plus efficace et stable ;
- d'autre part, aux chercheurs en sciences sociales de révéler comment l'utilisation d'électricité influence les divers phénomènes sociaux, et vice versa.

Cependant, les données sont très souvent insuffisantes pour produire des estimations et prévisions à l'échelle spatio-temporelle exigée par les applications mentionnées. Deux sources de données sont souvent disponibles aux entreprises d'électricité :

- des mesures de capteurs installés sur le réseau ;
- des mesures venant des compteurs de clients.

Les deux sources de données ont des limitations importantes. Les données de capteurs sont souvent de fréquence temporelle assez haute (pas de temps de 10 ou 15 minutes), mais elles couvrent un nombre important de clients. En France, le dernier niveau du réseau de distribution équipé de capteurs de manière systématique est celui des départs Haute Tension A (HTA). Chaque départ HTA couvre en moyenne 1500 clients. D'autre part, les compteurs clients ne couvrent en général qu'un seul client. Par contre, la plupart des clients sont pour l'instant équipés de compteurs traditionnels qu'il faut relever de manière manuelle : un technicien doit se déplacer physiquement chez le client pour lire l'index sur le compteur. Ainsi, ces index sont souvent relevés une fois tous les quelques mois. Avec le déploiement de compteurs intelligents (Linky), les index peuvent être stockés localement toutes les 10 minutes. Néanmoins, à cause des coûts de transmission et de traitement, ainsi que des considérations de protection de la vie privée, ils ne seront accessibles aux entreprises d'électricité que sous forme de sommes journalières.

Dans cette thèse, on envisage des méthodes statistiques pour produire des estimations et prévisions sur une échelle spatio-temporelle assez fine. L'objectif est de développer des méthodes qui peuvent utiliser toute information disponible, pour avoir des estimations de qualité supérieure. À part les deux sources de données de consommation, les méthodes développées ici utilisent notamment des "informations auxiliaires", ou *side information* en anglais, pour enrichir les estimations. Ces informations auxiliaires peuvent être globales, comme l'autocorrélation temporelle des courbes de consommation. Elles peuvent aussi être locales : des variables exogènes correspondant à une période et un individu dans le jeu de données, comme la température d'une période ou l'information socio-démographique d'un client.

1.2 Formalisation du problème

Dans cette thèse, on s'intéresse à la valeur d'une variable d'intérêt $v \in \mathbb{R}$ (typiquement la consommation électrique), pendant n_1 périodes consécutives, pour un groupe de n_2 individus[1]. Pour un individu j, $j \in \{1, 2, ..., n_2\}$, $v_{i,j}$ représente la consommation pendant la période i, pour $i \in \{1, 2, ..., n_1\}$. On utilise des lettres majuscules en gras pour représenter des matrices et des lettres minuscules en gras pour représenter des vecteurs. Ainsi, $\mathbf{V} \in \mathbb{R}^{n_1 \times n_2}$ est la matrice regroupant la variable d'intérêt, $(\mathbf{v}^i)^T$ est le i-ième ligne de $\mathbf{V}$ (la consommation de tous les individus à la période i), et $\mathbf{v}_j$ est la j-ième colonne (la consommation de l'individu j à travers les périodes).

On peut aisément formaliser le problème d'estimation mentionné ci-dessus avec cette représentation matricielle. La matrice représentant la variable d'intérêt est le tableau bleu dans figure 1.1. Chaque colonne représente un individu et chaque ligne une période. Les deux sources de données partielles sont des sommes d'entrées de matrice : le rectangle orange représente une agrégation temporelle, et le rectangle vert représente une agrégation d'individus (spatiale par exemple).

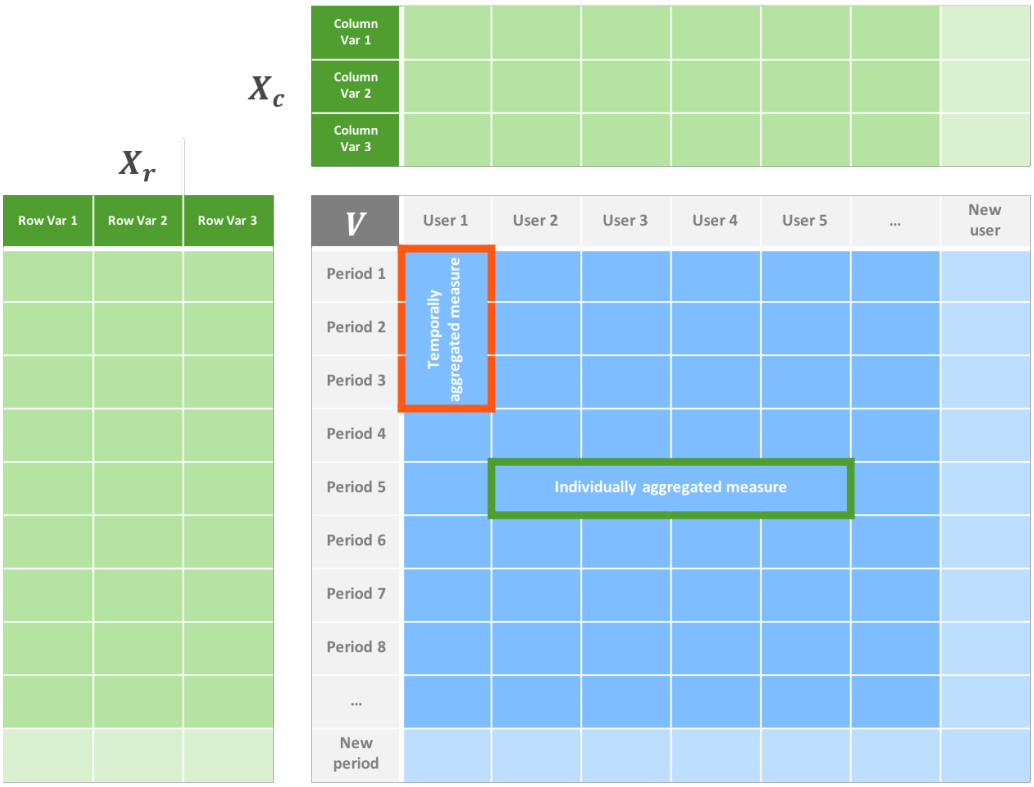


Figure 1.1: *Représentation matricielle de la variable d'intérêt. Dans cette thèse, une entrée de la matrice est souvent la consommation électrique d'un individu (une colonne), durant une période (une ligne). Les mesures de capteurs sur le réseau sont souvent des agrégations d'individus (somme d'entrées dans le rectangle vert). Les relevés de compteurs individuels sont souvent des agrégations temporelles (somme d'entrées dans le rectangle orange).*

Les variables exogènes sont représentées par les deux tableaux verts. Leurs lignes et colonnes sont alignées avec celles du tableau bleu.

Les objectifs d'estimation ont une représentation naturelle dans cette formalisation matricielle. Dans la plupart du temps, on est intéressé par la valeur de la variable d'intérêt pour chaque période de chaque utilisateur. Ainsi, il faut estimer toutes les entrées de $\mathbf{V}$. Selon des

[1] On entend par "un individu", l'unité basique considérée dans l'application. Ceci peut être une personne physique, mais aussi un compteur électrique, ou encore un capteur sur un niveau du réseau électrique.

applications, on peut être intéressé par une agrégation des entrées de $\mathbf{V}$. Par exemple, dans le problème de changement de support spatial, on est amené à estimer la variable d'intérêt sur une agrégation spatiale (entre individus) différente de celle des données. On peut aussi s'intéresser à la valeur maximale de la consommation électrique quotidienne, ce qui correspond à la valeur maximale de chaque colonne pendant un certain nombre de périodes.

La prévision correspond à estimer une ou plusieurs nouvelles lignes (nouvelles périodes) ou colonnes (nouvelles individus) de $\mathbf{V}$ dans laquelle aucune entrée n'a été couverte par les données. Quand aucune information à part les mesures de $\mathbf{V}$ n'est disponible, il est difficile de produire des prédictions. On considère deux cas de variables exogènes dans cette thèse :

- information implicite sur $\mathbf{V}$, telle que les colonnes de $\mathbf{V}$ ont une structure auto-régressive, soit entre périodes temporelles, soit entre des individus qui ont une structure de corrélation spatiale;
- information explicite, comme par exemple la température de chaque période, ou des informations socio-démographiques sur les individus.

Cette thèse est organisée de la manière suivante : dans le reste de ce chapitre, les trois approches envisagées dans la thèse seront présentées brièvement. L'approche basée sur la factorisation de matrice, qui est la proposition principale de la thèse, sera introduite de manière un peu plus détaillée. À part ce chapitre, le reste de cette thèse sera en anglais. Le chapitre 2 reprendra le contenu de ce chapitre et fera une introduction en anglais. Les chapitres 3 et 4 forment la première partie, où l'on considère des données qui sont des agrégations spatiales ou des agrégations d'individus, et dont le but est l'estimation à des niveaux d'agrégation différentes de celles des données. Dans la deuxième partie, les chapitres 5-9, on présente l'approche basée sur la factorisation de matrice nonnégative, qui permet de résoudre le problème de manière plus systématique. Le lien entre les trois approches est résumé dans le chapitre 8 et les perspectives sont présentées dans le chapitre 10.

1.3 Solutions classiques et nouvelles

1.3.1 Statistique spatio-temporelle

Quand les positions géographiques des individus sont disponibles, un choix naturel est d'utiliser des outils de la statistique spatiale. En considérant des valeurs individuelles comme observations d'un processus aléatoire spatial ou spatio-temporel, cette classe de méthodes fournit des modèles qui sont adaptés pour des phénomènes assez divers. Dans le chapitre 3, on présentera la statistique spatiale. En particulier, on discutera du krigeage (*kriging* en anglais), une méthode qui permet d'interpoler et d'extrapoler la variable d'intérêt à des positions géographiques nouvelles. En modélisant la structure de corrélation du processus spatial, cette méthode peut fournir des estimations *optimales*. On appliquera cet outil à deux jeux de données, pour examiner sa capacité à résoudre des problèmes représentés sur la figure 1.1.

1.3.2 Prédiction de consommation électrique multiple

Dans une approche parallèle à celle de la statistique spatiale, on présentera une méthode qui modélise plutôt la structure de la moyenne de la variable d'intérêt dans le chapitre 4. Cette approche permet d'estimer la consommation d'électricité des zones cibles, où l'information socio-démographique est disponible, à partir des zones sources, où la consommation est disponible à une échelle temporelle petite. Cette méthode utilise le clustering pour relier ces deux types

de zones. Le transfert des modes de consommation électrique est réalisé par des modèles de régression utilisant des variables explicatives telles que la température ou le cycle annuel, hebdomadaire et journalier. Cette méthode est évaluée par sa capacité de prédiction sur deux jeux de données synthétiques et réelles.

1.3.3 Méthodes basées sur la factorisation de matrice

Comme suggéré par la figure 1.1, l'objet d'intérêt de cette thèse se prête bien à une représentation matricielle. Du chapitre 5 au chapitre 9, on propose une approche générique basée sur la factorisation de matrice nonnégative [4–6] qui résout le problème d'estimation. Nous allons présenter les intuitions derrière cette approche et la contribution de cette thèse sur ces méthodes dans la section suivante.

1.4 Contributions à la factorisation de matrice nonnégative

1.4.1 Introduction à la factorisation de matrice nonnégative

Récemment, la factorisation de matrice nonnégative (*nonnegative matrix factorization* en anglais, ou NMF) a connu un succès important dans des domaines très divers, de la segmentation des images [4] au système de recommandation [7] en passant par le clustering des documents [8] et la décomposition du son [9].

L'idée principale derrière cette classe de méthodes est la suivante. Étant donné $\mathbf{V} \in \mathbb{R}_+^{n_1 \times n_2}$, une matrice de grande taille, on peut souvent faire l'hypothèse que son rang k est faible devant sa dimension. Les colonnes de $\mathbf{V}$ sont ainsi à l'intérieur d'un sous-espace de $\mathbb{R}^{n_1}$, de dimension k, bien inférieure à n_1. Ceci équivaut à dire que $\mathbf{V} = \mathbf{F}_r \mathbf{F}_c^T$, avec $\mathbf{F}_r \in \mathbb{R}_+^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}_+^{n_2 \times k}$. Les matrices $\mathbf{F}_r$ et $\mathbf{F}_c$ sont appelés les *facteurs*.

Le faible rang du problème a une interprétation très intuitive : dans le clustering de documents, le nombre de thèmes est souvent bien inférieur au nombre de documents et au nombre de mots dans le dictionnaire ; dans un système de recommandation, il y a peu de clients et marchandises "indépendants" parmi un grand nombre de clients et marchandises. Le choix d'imposer les contraintes de non-négativité est souvent justifié par les applications aussi : avec des facteurs, on peut aisément interpréter les documents comme un mélange de thèmes, et une marchandise comme un mélange de plusieurs types de marchandises.

Une interprétation du même type pour les applications dans le domaine d'électricité nous semble aussi naturelle, puisqu'on remarque qu'il y a un nombre réduit de manières d'utiliser l'électricité (appelé les *profils*), bien qu'il y ait un nombre important de clients. Tous les clients sont proches d'un mélange de ces profils. Cet aspect est d'ailleurs pris en compte dans des systèmes actuels de reconstruction de consommation électrique, qui utilisent des profils nationaux de consommation [10]. Ces profils sont actuellement calibrés par des méthodes statistiques qui nécessitent des traitements assez longs. Utiliser des méthodes basées sur la factorisation de matrice nonnégative permettrait de les estimer de manière beaucoup plus automatique, à partir des données partielles.

Pour obtenir une factorisation, on peut résoudre le problème d'optimisation suivant,

$$\begin{aligned} \min_{\mathbf{F}_r \in \mathbb{R}^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}^{n_2 \times k}} \quad & \ell(\mathbf{V}, \mathbf{F}_r \mathbf{F}_c^T) \\ \text{s.t.} \quad & \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}, \end{aligned}$$

où ℓ est une fonction de perte qui mesure la différence entre $\mathbf{F}_r \mathbf{F}_c^T$ et $\mathbf{V}$, et les inégalités sont

élément par élément. La fonction ℓ peut être assez générale, mais on considère principalement la perte quadratique en norme Frobenius où $\ell(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X} - \mathbf{Y}\|_F^2 = \sum_{i,j}(x_{i,j} - y_{i,j})^2$.

La factorisation de matrice n'est pas un nouveau problème. Dans le cas de l'Analyse en Composantes Principales (ACP), une factorisation de matrice est aussi calculée dans laquelle un des facteurs est contraint à être une matrice orthogonale (celui des composantes principales).

Malgré son interprétation très intuitive, la NMF n'est pas facile à résoudre. En fait, dans le cas général, la NMF est NP-difficile [11].

Une ligne de recherche théorique consiste à étudier si une matrice avec des entrées nonnégatives a une NMF "unique". Pour toute paire $(\mathbf{F}_r, \mathbf{F}_c)$, une NMF de $\mathbf{V}$, et pour toute matrice inversible $\mathbf{R} \in \mathbb{R}^{k\times k}$, $(\mathbf{F}_r\mathbf{R}, \mathbf{F}_c(\mathbf{R}^{-1})^T)$ est encore une factorisation de $\mathbf{V}$ (leur produit est égal à $\mathbf{V}$). Mais cette nouvelle factorisation n'est pas nécessairement nonnégative. Par contre, pour certaines matrices, il est possible que les seules matrices inversibles $\mathbf{R}$ telles que $(\mathbf{F}_r\mathbf{R}, \mathbf{F}_c(\mathbf{R}^{-1})^T)$ reste une factorisation nonnégative soient des compositions de dilatation et de permutation. Dans ce cas-là, on dit que la NMF est "unique". Décider si une matrice a une NMF unique est aussi un problème difficile. Deux conditions nécessaires et suffisantes de l'unicité ont été trouvées, mais étant donnée $\mathbf{V}$, il est difficile de décider si elle vérifie ces conditions [5, 12]. Des conditions suffisantes, plus faciles à vérifier, ont été proposées sur des classes particulières de NMF [12, 13].

Des algorithmes avec garantie de convergence ont été proposés pour certaines classes de NMF, en particulier celles qui vérifient les conditions suffisantes d'unicité [14–17]. Ces algorithmes sont souvent assez spécifiques, et la preuve de convergence utilise des outils complexes. Pourtant, en pratique, des algorithmes dits de "premier ordre" sont préférés. Ces algorithmes sont souvent basés sur la descente de gradient ou l'algorithme de descente par bloc. Même s'ils ont peu de garantie théorique, la simplicité d'implémentation et la performance empirique ont fait qu'ils sont devenus très populaires [18].

L'idée principale de ces algorithmes de descente est la méthode Gauss-Seidel : soit une fonction $f(\mathbf{x}_1, ..., \mathbf{x}_d)$ à minimiser sur un produit cartésien de convexe $\mathcal{X}_1 \times ... \times \mathcal{X}_d$. L'algorithme minimise par rapport à chaque bloc de variables $\mathbf{x}_1, ..., \mathbf{x}_d$ de manière itérative. Dans le cas de NMF, les algorithmes sont caractérisés par les blocs de variables : on peut utiliser des blocs scalaires et alterner entres les éléments de $\mathbf{F}_r$ et $\mathbf{F}_c$, des blocs vectoriels (les colonnes et lignes de $\mathbf{F}_r$ et $\mathbf{F}_c$), ou des blocs matriciels ($\mathbf{F}_r$ et $\mathbf{F}_c$).

Grâce aux résultats d'optimisation, les algorithmes du type Gauss-Seidel convergent vers un point stationnaire sous des conditions faibles. Ainsi la plupart des algorithmes de NMF convergent vers un point stationnaire.

1.4.2 Appliquer la NMF pour estimer la matrice V (chapitres 5, 6)

La contribution principale de la thèse est l'application de la NMF au problème introduit dans la section 1.2.

Sous l'hypothèse de faible rang, on peut formaliser le problème d'estimation de la figure 1.1 comme un problème similaire à la NMF. Dans ce problème, la matrice $\mathbf{V}$ est à estimer, à partir de mesures linéaires $\boldsymbol{\alpha}$:

$$\begin{aligned} \min_{\mathbf{F}_r\in\mathbb{R}^{n_1\times k}, \mathbf{F}_c\in\mathbb{R}^{n_2\times k}, \mathbf{V}\in\mathbb{R}^{n_1\times n_2}} \quad & \|\mathbf{V} - \mathbf{F}_r\mathbf{F}_c^T\|_F^2 \\ \text{s.t.} \quad & \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}, \quad \mathbf{V} \geq \mathbf{0}, \\ & \mathcal{A}(\mathbf{V}) = \boldsymbol{\alpha}, \end{aligned} \tag{1.1}$$

où $\mathcal{A}$ est un opérateur linéaire sur l'espace des matrices de taille $n_1 \times n_2$ et $\boldsymbol{\alpha}$ est un vecteur de

données de dimension N. Les opérateurs linéaires sur $\mathbb{R}^{n_1 \times n_2}$ sont caractérisé par N matrices de taille $n_1 \times n_2$, $A_1, ..., A_N$, et

$$\alpha_i = \mathrm{Tr}(\mathbf{V}\mathbf{A}_i^T) = \langle \mathbf{V}, \mathbf{A}_i \rangle.$$

Ces matrices $A_1, ..., A_N$ sont appelés des *masques*.

Dans le chapitre 6, nous proposons une extension de la NMF pour utiliser les agrégations temporelles comme données.

Ceci correspond à expliciter (1.1) dans le cadre où les mesures linéaires sont des agrégations temporelles. Concrètement, ceci veut dire que pour tout $1 \leq i \leq N$,

$$\alpha_i = \langle \mathbf{V}, \mathbf{A}_i \rangle = \sum_{(t,n) \in I_i} v_{t,n},$$

où I_i est l'ensemble des indices des périodes couvertes par la i-ième mesure. Ces contraintes de somme, et la contrainte de nonnégativité, définissent la région admissible de $\mathbf{V}$, qui est un simplex.

Dans le chapitre 6, nous proposons un algorithme qui résout (1.1). Pour ce faire, nous ajoutons, aux algorithmes classiques de NMF, une étape de projection dans ce simplex défini par des contraintes (Algorithm 1). Cet algorithme combine un algorithme efficace de projection dans un simplex, et deux algorithmes classiques de NMF [19–21]. C'est un algorithme convergent vers un point stationnaire comme la plupart d'algorithmes de NMF.

En plus de cet algorithme, nous proposons aussi de prendre en compte deux types d'informations supplémentaires.

1.4.3 Autocorrélation temporelle des individus (chapitre 6)

Le premier type d'information supplémentaire est l'autocorrélation temporelle des individus. En pratique, il est connu que les séries temporelles de consommation électrique sont souvent assez régulières au sens où il existe une certaine dépendance temporelle entre les observations. Nous prenons en compte cet aspect dans chapitre 6.

La régularité des facteurs a été envisagée dans de nombreuses études de factorisation de matrice ou de NMF. Souvent, une pénalisation est ajoutée à la fonction objectif pour obtenir des facteurs plus réguliers et/ou plus sparses [22–26]. Ici, nous avons un objectif différent. Nous supposons avoir de l'information a priori sur l'autocorrélation temporelle des individus, et essayons d'améliorer la reconstitution grâce à cette information supplémentaire. Cette information peut venir de sources différentes : elle peut être le résultat des observations historiques (sur des individus qui ont participé à des études expérimentales, par exemple), ou elle peut être issue de modélisation.

Nous modifions (1.1) en ajoutant un terme de pénalisation lié aux seuils d'autocorrélation. Nous montrons qu'ajouter la pénalisation est dans certains cas équivalent à contraindre que l'autocorrélation de chaque individu soit supérieure à ces seuils (Theorem 1). Nous proposons un algorithme pour ce problème modifié (Algorithm 2).

1.4.4 Prendre en compte des informations exogènes (chapitre 7)

Le deuxième type d'information supplémentaire que nous proposons de prendre en compte concerne les variables exogènes. Dans ce cadre, comme mentionné dans figure 1.1, des variables exogènes liées correspondant aux lignes ou colonnes de la matrice $\mathbf{V}$ sont disponibles.

Dans la littérature de la NMF, les variables exogènes sont un sujet qui attire beaucoup d'attention. Ce principe est applicable aux diverses applications, notamment en système de recommandation, où la préférence d'un client pour une marchandise dépend naturellement des attributs du client (âge, sexe, profession, *etc.*) et de la marchandise (prix, utilité, gamme, *etc.*). En ce qui concerne l'application en électricité, ce principe est également pertinent, car la consommation électrique dépend d'attributs liés à la période (température, nébulosité, heure de la journée, jour de la semaine), et au client (résidentiel ou industriel, composition du ménage, type de logement).

Nous résumons l'état de l'art de la littérature sur ce sujet dans la table 7.1, sur la page 85. Remarquons que pour les applications qu'on envisage, il est intéressant de pouvoir traiter des mesures linéaires comme observations et des modèles de régression non-paramétriques pour gérer les variables exogènes. Comme les méthodes existantes ne sont pas assez flexibles sur ces deux points, nous proposons un cadre général de NMF avec variables exogènes (cf. section 7.1.1).

Connaissant des variables exogènes liées aux colonnes $\mathbf{X}_c$ et aux lignes $\mathbf{X}_r$, en plus des mesures linéaires $\boldsymbol{\alpha}$, nous proposons de résoudre

$$\begin{aligned} \min_{\mathbf{V}, f_r \in F_r^k, f_c \in F_c^k} \quad & \|\mathbf{V} - (f_r(\mathbf{X}_r))_+ (f_c(\mathbf{X}_c))_+^T\|_F^2 \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{V}) = \boldsymbol{\alpha}, \quad \mathbf{V} \geq \mathbf{0}, \end{aligned} \tag{1.2}$$

où $(\cdot)_+$ est la fonction qui remplace toute valeur négative par 0, et F_r et F_c sont des espaces fonctionnels dans lesquels nous cherchons des fonctions de régression.

En variant l'opérateur de mesure $\mathcal{A}$, l'espace fonctionnel de régression, et les variables exogènes, cette modélisation peut être adaptée à des problèmes assez divers. En particulier, pour modéliser les relations non-linéaires entre les variables exogènes et les facteurs, nous utiliserons des splines de régression et des méthodes à noyaux dans l'application de ce modèle.

Sur l'aspect théorique de ce modèle, nous déduisons une condition suffisante d'unicité de NMF dans le cadre de NMF avec variables exogènes (Theorem 5). En pratique, garantir que la NMF est unique permet de mieux interpréter les facteurs obtenus par le modèle.

Algorithme convergent vers un point stationnaire

Nous proposons un algorithme pour résoudre (1.2) (Algorithm 4). Similaire aux algorithmes 1 et 2, dans cet algorithme appelé HALSX (Hierarchical Alternating Least Squares with eXogeneous variables), nous ajoutons une étape supplémentaire dans HALS, qui est un algorithme classique de NMF [27]. Cette étape consiste à estimer des modèles de régression avec comme variables explicatives les variables exogènes.

Nous démontrons que cet algorithme converge vers un point stationnaire pour une grande classe de modèles de régression (Theorem 7).

1.4.5 Applications aux jeux de données synthétiques et réelles

Les méthodes proposées sont évaluées sur plusieurs jeux de données synthétiques et réelles.

Les jeux de données synthétiques sont créés pour tester la pertinence des méthodes proposées dans un environnement contrôlé. Cela consiste à simuler des matrices de rang faible de manière exacte. Les détails de simulation sont présentés dans les sections 6.3 et 7.4.

Jeux de données de consommation électrique Les jeux de données réelles de consommation électrique sont les suivants :

- **Consommation électrique française** (Données fournies par Enedis) la consommation journalière des 636 départs HTA (haute tension A) dans la région autour de Lyon. Chaque individu dans ce jeu de données correspond à la consommation d'un groupe d'environ 1500 clients. Les données brutes sont disponibles au pas de 10 minutes, pendant 3 ans. Pour les expériences des chapitres 6 et 7, la consommation est agrégée au pas journalier.

- **Consommation électrique portugaise** (Jeu de données publiques[28]) la consommation journalière de 370 clients individuels (anonymisés) portugais. Issu d'un jeu de données publique, la consommation est disponible au pas de 15 minutes pendant 4 ans. Pour les expériences aux chapitres 6 et 7, la consommation est agrégée au pas journalier.

- **Consommation électrique irlandaise** (Jeu de données publiques[29, 30]) la consommation journalière de 426 petites et moyennes entreprises (PME) irlandaises. Issu d'une expérimentation européenne, la consommation est disponible au pas horaire pendant l'an 2010. L'agrégation journalière de ce jeu est utilisée au chapitre 6.

Jeux de données de système de recommandation Pour évaluer la performance de la méthode sur un opérateur de mesure linéaire autre que l'agrégation temporelle, on utilise aussi un jeu de données standard en système de recommandation.

- **MovieLens** C'est un jeu de données publique de scores de films. En tout, 100,000 scores sont disponibles pour 943 utilisateurs et 1682 films.

Comme schématisé dans la figure 1.1, deux applications sont considérées :

Reconstitution de consommation passée/estimation de score de films (chapitre 6 et 7) Nous mesurons les agrégations temporelles sur un jeu de données de consommation en tirant au hasard un certain nombre de dates d'observation. Nous utilisons une méthode pour estimer la matrice de consommation, et mesurons la performance en comparant les estimations avec la consommation réelle. Pour le jeu de données de score de films, nous tirons au hasard un sous-ensemble de score comme observations, et complétons la matrice des scores. Ensuite nous comparons les estimations avec les scores qui n'ont pas été observés.

Prédiction pour les nouveaux clients et/ou les nouvelles périodes (chapitre 7) Nous cachons toutes les observations sur certains individus et certaines dates (ou certains films dans le cas de MovieLens), et appliquons la NMF avec variables exogènes sur les observations restantes. Nous produisons ensuite des prédictions pour les dates et les individus cachés. Nous les comparons avec la réalité pour mesurer la performance de la méthode.

1.4.6 Implémentation

Les algorithmes NMF proposés qui ont été développés, sont regroupés dans un package `R` appelé `meterModels`. Pour faciliter les tests, des fonctions d'expérimentation, d'évaluation des méthodes, ainsi que des méthodes de référence sont aussi incluses dans ce package.

La plupart du package est en `R`. Certaines fonctions intensives en calcul sont écrites en `C++` avec une interface en `R`, pour accélérer l'exécution des programmes. Quand la bibliothèque est disponible, le package peut utiliser `openMP` pour distribuer certaines calculs. Quand le package est utilisé sur une machine multi-cœurs, l'estimation de plusieurs modèles peut aussi se faire en parallèle en utilisant le package `parallel` de `R`.

Représentation des observations Les mesures d'agrégation temporelle et les mesures linéaires générales sont gérées par le package. Pour les agrégations temporelles, les observations sont représentées comme des dates d'observation et des index de compteurs. Pour les mesures linéaires générales, l'opérateur de mesure est représenté comme une grande matrice sparse, et les observations sont le produit de l'opérateur et le vecteur des entrées de la matrice.

Méthodes disponibles Les méthodes de NMF avec agrégations temporelles ou mesures linéaires sont disponibles, ainsi que les variantes proposées par la thèse : la prise-en-compte de l'autocorrélation et des variables exogènes. Les méthodes proposées ainsi que les méthodes de référence sont détaillées dans la table 9.1 sur la page 118. L'estimation de modèle se fait à travers une interface unifiée, où le choix des méthodes se fait par un paramètre. Il est aussi possible de comparer plusieurs méthodes sur un jeu de données avec des paramètres variables.

1.5 Conclusions et perspectives

Dans cette thèse, nous avons étudié un problème pratique en industrie : la modélisation de la consommation électrique à partir des données partielles issues de nombreux individus. Nous avons envisagé ce problème sous différents angles, avec des sources de données agrégées temporellement ou individuellement, avec comme objectif l'estimation de la consommation passée, et la prévision de la consommation future. Nous avons investigué des solutions dans plusieurs cadres différents : la statistique spatio-temporelle, le clustering des individus, ainsi que la factorisation de matrice nonnégative. Particulièrement pour la NMF, nous avons adapté la méthode pour la problématique, et proposé des améliorations méthodologiques par rapport à l'état de l'art. La méthodologie a été évaluée par des expériences nombreuses sur des jeux de données synthétiques et réelles, et des résultats prometteurs ont été obtenus.

Des perspectives de recherche sont envisagées. Outre les applications directes de la méthodologie étudiées dans cette thèse, il existe de nombreuses applications industrielles. Nous pouvons utiliser la NMF pour estimer des statistiques importantes du réseau électrique. Par exemple, par rapport à la consommation électrique de chaque moment, la consommation maximale instantanée est cruciale pour la maintenance du réseau. Il serait intéressant d'évaluer la méthode de NMF dans ce contexte, voire même changer la fonction objectif du problème d'optimisation pour s'adapter à la statistique qui nous intéresse. Dans la thèse, nous avons fait l'hypothèse que le dispositif d'acquisition des données est fixe. Or, il peut être envisageable de modifier la manière dont les données sont collectées, et donc de trouver une manière optimale de collecter les données, sachant que la méthodologie de matrice de rang faible est disponible pour le traitement de ces données. Nous avons étudié des méthodes adaptées séparément pour les agrégations temporelles et individuelles. Il serait intéressant de voir si combiner les deux types de données peut encore améliorer la performance de la méthode. Sur l'aspect théorique de la NMF, des questions importantes restent également ouvertes, comme l'analyse de la convergence globale des algorithmes de premier ordre pour certaines classes de matrices, et l'approfondissement des conditions d'unicité de la NMF.

Chapter

2 Introduction

Contents

2.1 Motivation for electricity consumption estimation and prediction

To ensure the safety and stability of electric power systems, the electricity injected into a network has to be met with the load, namely the consumption. This is called load balancing. Electricity consumption is determined by the client mix connected to the network and external conditions, on which utility companies have very little control. As a core characteristic of electricity is that it can not be stored efficiently, utilities mostly achieve load balancing by adjusting power generation and operating the electric network according to real-time estimation and predictions of the load[1].

Given these conditions, it is easy to see how electricity consumption data can be useful for a utility company, such as Electricité de France (EDF). In order to plan power generation and network development, it is important to have mid-term or long-term consumption prediction, starting from six-month ahead. To plan network operations, it is useful to have temporally fine-grained short or mid-term prediction and estimation, for example hourly or daily prediction until several months ahead. To take fully advantage of the burgeoning decentralized renewable energy generation in load balancing, it can be interesting to have consumption data and prediction on a fine spatial grid.

Apart from its value for power planning and network operations, electricity consumption data can be valuable for other reasons. In the context of open electricity market, regulations often stipulate that each market participant, *i.e.* suppliers, utility traders, large consumers, be responsible for its own load balancing at all time. Any imbalance caused by a participant is typically billed by transmission system operators (TSO). To calculate the imbalance, TSOs have to produce an hourly or half-hourly estimation of the consumption of each participant.

Electricity is an important part of socio-economic activities, and thus can be put into connection with many other public datasets concerning the society. Should electricity consumption

[1] Historically, this is mostly done by hydroelectric energy storage, namely pumping water into a reservoir in higher locations during low-demand periods, and using that water to generate electricity during peaks. Nowadays, battery storage is extensively researched, but remains expensive and experimental for the moment.

data be available at the same spatio-temporal scale as other socio-demographic information, utility suppliers can provide energy service more efficiently and stably, and researchers in social science can shed new lights on various social phenomena by crossing electricity data with data from other domain.

However, there is rarely enough data to produce estimations and predictions that are spatio-temporally fine-grained enough for these applications. Two sources of consumption data are often available to utility companies:

- measures taken by sensors installed on the network;
- measures read on end users' electric meters.

Both data sources are somewhat limited. Sensor data on the electric network are of sufficiently high frequency (at an interval of ten or fifteen minutes), but covers a rather big area. In France, for example, the lowest level in the electric grid in which sensor data is available is at the medium-voltage (MV) feeder level, where each MV feeder covers approximately 1500 clients. Individual electric meter, on the other hand, only covers one client. However, traditional meters typically can only be read manually by a worker who has to physically visit every client. Therefore, such meter data can only be at an interval of several months. With the development of smart meters, consumption can be recorded locally up to every minute. Nevertheless, due to transmission and processing costs and/or privacy issues, utility companies often have limited access to these data, for example daily aggregates of individual consumption.

In this thesis, we investigate statistical methods to produce estimations and predictions of spatio-temporally fine-grained electricity consumption. The objective is to use any information that is available, in order to produce high-quality estimation and prediction. Apart from the two sources of electricity consumption data considered above, the methods developed in this thesis use *side information* to enrich the estimations. Such side information could be global, such as the autocorrelation of the time series of electricity consumption. Or they could be local: exogenous variables corresponding to the periods and individuals in the dataset, for example, the temperature of a given period, or the socio-demographic information, or geographic location of the clients.

Given the importance of load information for the electric network, a plethora of authors have studied load modeling and forecasting. Traditionally, spatial load forecasting [31], focused on forecasting long-term trend in load over spatial regions, and mid-to-short term forecasting [32], focused on the national level, are two separate domains with differnt methodologies. This has changed in recent years, since the demands for load forecasting have changed due to the evolution of the electricity market as well as the availability of large-scale datasets.

On one hand, the spatial scale of short and mid-term load forecasting has drastly decreased [33–36]. A by-product of this change is that more attention is paid to seeing electric load as multivariate time series which needs to be processed as a whole. In this area, particular problems to be addressed are the similarity between load series [1, 35], and the coherent structure between series from multiple components connected through the network [37].

On the other hand, this change is also accompanied by the development of new forecasting methods, both from statistics and the machine learning community. As a testimony to this trend, winners for recent load forecasting competitions often used a variety of differnt methods. In Global Energy Forecasting Competition 2014 [38], the winner of the load forecasting track used a quantile regression version of generalized additive model (GAM) to tackle probabilistic load forecasting [39]. In a 2017 multivariate load forecasting contest organized by Réseau de Transport d'Electricité, the main TSO in France [40], the winner used an ensemble of Xgboost

models to achieve an adaptive model with high performance.[2]

We believe the methods presented in this thesis are a useful complement to these recent developments, as they allow heterogenous data to be preprocessed, before state-of-art load modeling methods can be applied.

2.2 Problem formalization

In this thesis, we are interested in the value of a variable $v \in \mathbb{R}$ (typically electricity consumption), over n_1 consecutive periods, for a group of n_2 individuals. For an individual j, with $j \in \{1, 2, ..., n_2\}$, $v_{i,j}$ represents its consumption at period i, with $i \in \{1, 2, ...n_1\}$[3]. We will use lower-case boldface letters to denote vectors and upper-case boldface letters to denote matrices, so that $\mathbf{V} \in \mathbb{R}^{n_1 \times n_2}$ is the matrix of the variable of interest, and $(\mathbf{v}^i)^T$ is the i-th row vector of this matrix, representing the i-th period, and $\mathbf{v}_j$ is the j-th column vector, representing the j-th individual.

With this notation, we can easily represent both the data and the estimation target of this thesis. In Figure 2.1, a scheme of the variable of interest (blue table) is given. We can also represent the two types of data sources previously mentioned: temporally aggregated measure from a user meter (orange box), and measures taken on from a point on the electricity network, thus an aggregation of several individuals (green box).

The exogenous variables can be represented by matrices that can be put in parallel to the variables of interest (green tables in Figure 2.1).

The estimation targets can also be represented in this way. Most of the time, we aim at estimating the electricity consumption of all individuals for all periods. This is simply every entry of the matrix $\mathbf{V}$. Depending on the applications, one can also be interested in some aggregations of the matrix $\mathbf{V}$. These estimation tasks can be achieved by performing the aggregation on the estimated matrix $\mathbf{V}$. For example, in a change-of-support problem, we would want to estimate the variable of interest on a spatial aggregation grid, that is different from that of the data. One can also be interested in the maximal value of electricity consumption throughout a day, which is the maximum of a number of periods for each user.

As for prediction, it corresponds to estimating the value of the variable for new rows (new periods) and new columns (new users) of $\mathbf{V}$. When no information other than measures on the data matrix is available, it can be difficult to produce predictions. In this thesis, prediction is often achieved by considering

- either implicit information about $\mathbf{V}$, such as the temporal auto-regressive structure of the rows of $\mathbf{V}$, or the spatial auto-regressive structure of columns of $\mathbf{V}$;
- or exogenous variables *per se*, such as the temperature, or socio-demographic information of clients.

This thesis is organized as following: In Chapters 3-4, we present two methods that use individually aggregated data to estimate and predict electricity consumption for aggregations that is different from that of the data. In Chapters 5-9, we present nonnegative matrix factorization, a class of methods that can solve the task in a more general way. The three approaches are compared in Chapter 8, and perspectives are discussed in Chapter 10.

[2] Jérémy Lesuffleur, personnal communication, 2017.

[3] We will call a basic unit an "individual", if it is the smallest unit considered in a given context, whether it is actually a physical person, a meter, a household or a group of clients.

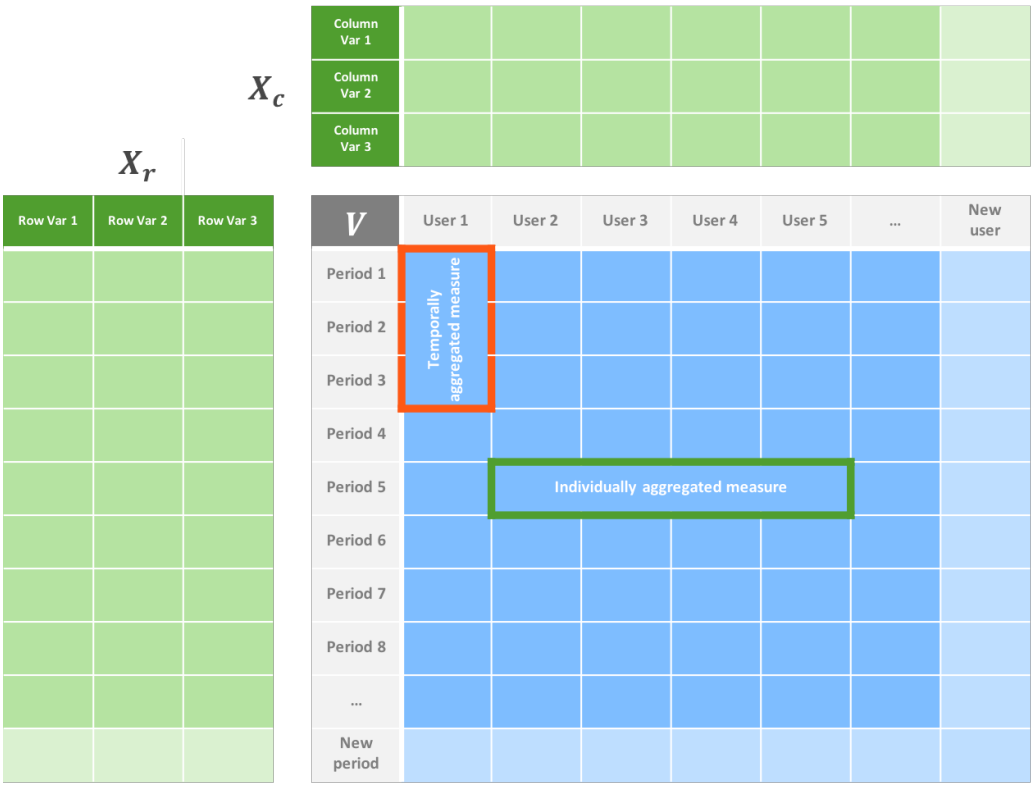


Figure 2.1: *The matrix representation of the variable of interest. Typically, an entry of the matrix is the electricity consumption of an individual (a column), during a period (a row). Measures taken by sensors on the network are often aggregations of individuals (sum of the entries in the green box). Measures read on user meters are often temporally aggregated (sum of the entries in the orange box).*

2.3 Classical and novel solutions

2.3.1 Spatio-temporal kriging

When geographic positions of the individuals are available, one natural way to model the data is to use spatial statistics. By considering the individual values of the dataset as observations of a spatial or spatio-temporal random process, this class of methods provide a toolbox for modeling a wide range of social and natural phenomena. In Chapter 3, we will briefly present the basics of spatial statistics. In particular, we will present *kriging*, a method to interpolate and extrapolate the value of variables of interest to new positions. Modeling the correlation structure of the spatial random process, this method can produce estimations that are *optimal* in a sense. By applying kriging to two datasets, we will examine the capacity of this model for solving the estimation problems of Figure 2.1.

2.3.2 Prediction of multiple electricity consumption series

In an approach parallel to spatial statistics, we will present a method to estimate the mean of electricity consumption of unobserved zones, in Chapter 4. This generic approach can estimate electric consumption from source zones, where fine-grained consumption data is available, to target zones where exogenous socio-demographic information is available. By using additional socio-demographic variables available both at source and target zones, this method creates a link between the two by clustering. The electricity consumption is then estimated by regression models. This method is validated on synthetic and real electricity consumption datasets.

2.3.3 Matrix-factorization based methods

As suggested by Figure 2.1, the object of interest in this thesis can easily be represented in a matrix form. In Chapters 5 to 9, we propose a general framework based on matrix factorization, which solves the estimation problem of Figure 2.1.

To do this, we first extend nonnegative matrix factorization (NMF) to the matrix recovery context where the observations are general linear measurements of the matrix, instead of matrix entries. In Chapter 5, we describe this extension, and discuss several applications that can be fit into this framework. In addition, we also discuss the recent literature on the global convergence of matrix factorization in the nonnegative case.

In Chapters 6 and 7, we propose two ways to include additional information in NMF. In Chapter 6, we pay especially attention to temporal aggregate data. We propose an efficient algorithm for this data representation, which is guaranteed to converge to a stationary point. We also provide another version of this algorithm, which takes into account the autocorrelation of the consumption of each individual. The utility of these two algorithms are illustrated on synthetic and real electricity consumption datasets, in comparison to reference methods.

In Chapter 7, we propose an extension that takes into account side information: additional time and individual-dependent features. By adding a regression layer into the NMF framework, the algorithm proposed in this chapter can be applied to many application problems. In particular, it can be used to produce electricity consumption predictions for new individuals, from only partially observed consumption data. The experimental study on several datasets shows that this method has very good empirical performance, compared to reference methods.

Part I

State-of-art methods

Chapter

3 Spatio-temporal change of support

Contents

In this chapter, we are interested in the following setting: electricity consumption data are available as individual aggregations, at a fine temporal scale. We want to estimate aggregations that are different from that of the data. We use tools from spatio-temporal statistics, and formulate this problem as a change-of-support problems . That is, we model electricity consumption as a spatially-indexed random process, and deduce estimations in this framework.

3.1 Introduction to spatial statistics

In spatial statistics, we study the properties of a social or natural phenomenon indexed by its location. In formal terms, the subject is a possibly multivariate random process $\{Z(s)\}$ with its index in a domain of interest $\mathcal{D}$:

$$\{Z(s) : s \in \mathcal{D}\},$$

where $\mathcal{D}$ is often a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$.

In general, both the position s and the variable of interest $Z(s)$ can be random. Here is of primary interest the case where the locations are fixed. According to the categorization adopted

by [41], with a fixed set of positions $\{s\}$, the data $Z(s)$ emitted by a stochastic process is called geostatistical data.

When the value of Z on a partition of the domain $\mathcal{D}$ is of interest, the aggregation or average of Z on small areas is often studied. In this case, the stochastic process is studied on a partition of $\mathcal{D}$, namely a collection of pairwise disjoint subsets $\{D_i\}_{i=1,2,...,n}$. This is called areal data analysis.

In this chapter, a particular class of questions addressed in spatial statistics, change-of-support problem (COSP), is considered. Areal datasets from different sources are often defined on overlapping spatial partitions. Therefore, in order to merge these datasets and study the mutual influence of different phenomena, it is necessary to transfer a dataset from one spatial support to another.

In this section, the basic assumptions of spatial statistics is presented. This section is brief, mostly following [41], where a thorough presentation of spatial statistics can be found.

In Section 3.2, methods of univariate COSP are presented. The most important of these is a method called kriging, derived from the geostatistics literature. A few areal interpolation methods other than kriging are presented in Section 3.2.3. In Section 3.3 and Section 3.4, these methods are generalized in the multivariate and spatio-temporal context. Finally, Section 3.5 shows a small empirical study of the methods presented.

3.1.1 Basics in geostatistics

As mentioned above, geostatistics is focused on a random process $\{Z(s)\}$ indexed by its fixed location s. The domain of interest $\mathcal{D}$, of permissible values for s, is a subset of $\mathbb{R}^d$ (in practice, $d = 2$ or 3). A realization of this random process is the observed value of Z, $z_1, z_2, ..., z_n$, at a number of locations: $s_1, s_2, ..., s_n$.

The random process model works well for applications in geology, mining, and environmental science, where a natural phenomenon, for example the ore reserve or air pollution, is measured by sensors at several locations, and something is to be said about the whole area.

The random distribution of a stochastic process is defined by a probability measure on the function space $\{f : \mathcal{D} \to \mathbb{R}\}$, which can be very general. We will focus on Gaussian processes, namely, for any number of positions $\{s_1, ..., s_n\}$, the joint distribution of $\{Z(s_1), ..., Z(s_n)\}$ is a multivariate Gaussian. A Gaussian process is completely characterized by its first two moments:

- a mean function $m : \mathcal{D} \to \mathbb{R}$, such that $m(s) = \mathbb{E}(Z(s)), \forall s \in \mathcal{D}$,
- and a covariance function $C : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$, such that $c(s_1, s_2) = \text{cov}(Z(s_1), Z(s_2)), \forall s_1, s_2 \in \mathcal{D}$.

Similar to time series analysis, which is a statistical analysis based on a stochastic process index by an interval of $\mathbb{R}$, reasonable assumptions of stationarity have to be made while analyzing geostatistical data. Several kinds of stationarity is defined in spatial process.

One useful assumption to consider is second-order stationarity. $\{Z(s)\}$ is *second-order stationary* if

- $\mathbb{E}(Z(s)) = \mu$, which is a constant for all $s \in \mathcal{D}$;
- there is a *covariance* function $C(\cdot)$ such that $\text{cov}(Z(s_1), Z(s_2)) = C(s_1 - s_2)$.

Additionally, if $C(s_1 - s_2)$ is in fact only a function of $\|s_1 - s_2\|$, then this process is *isotropic*.

A less strict assumption is intrinsic stationarity. $\{Z(s)\}$ is *intrinsically stationary* if

- $\mathbb{E}(Z(s)) = \mu$, which is a constant for all $s \in \mathcal{D}$;
- there is a *variogram* function $\gamma(\cdot)$ such that $\mathrm{Var}(Z(s_1) - Z(s_2)) = 2\gamma(s_1 - s_2)$.

Additionally, if $2\gamma(s_1 - s_2)$ is in fact only a function of $\|s_1 - s_2\|$, then this process is *isotropic*.

All second-order stationary process is intrinsically stationary because if the process has a covariance function $C(\cdot)$, then

$$\begin{aligned}
\mathrm{Var}(Z(s_1) - Z(s_2)) &= \mathbb{E}[(Z(s_1) - Z(s_2))^2] \\
&= \mathbb{E}(Z(s_1)^2) + \mathbb{E}(Z(s_2)^2) - 2\mathbb{E}(Z(s_1)Z(s_2)) \\
&= C(0) + \mu^2 + C(0) + \mu^2 - 2(C(s_1 - s_2) + \mu^2) \\
&= 2C(0) - 2C(s_1 - s_2) \\
&\equiv 2\gamma(s_1 - s_2).
\end{aligned}$$

Conversely, not all intrinsically stationary processes are second-order stationary. Consider the Lévy Brownian motion with d parameters $\{B(s)\}$. The variogram is well defined: $\mathrm{Var}(B(s_1) - B(s_2)) = \|s_1 - s_2\|$. However, $\mathrm{cov}(B(s_1), B(s_2)) = \|s_1\| + \|s_2\| - \|s_1 - s_2\|$, which is not a function of $\|s_1 - s_2\|$ ([41, p. 68]). In univariate kriging, we mostly consider intrinsically stationary processes, and thus studying variograms and not covariances, while in multivariate and spatio-temporal context, covariance function is preferred.

3.1.2 Variogram

For a function C or γ defined on $\mathcal{D}$ with values in $\mathbb{R}$ to be a covariance function or variogram, it must satisfy that for any n points in $\mathcal{D}$: $s_1, ..., s_n$, the variance-covariance matrix $(\mathrm{cov}(s_i, s_j))_{1 \le i,j \le n}$ is positive semi-definite. For stationary processes, this is equivalent to conditional negative-definiteness on variograms:

$$\forall a_1, ..., a_n \in \mathbb{R} \text{ not identically zero}, \quad \text{such that } \sum_{i=1}^{n} a_i = 0, \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \gamma(s_i - s_j) \le 0.$$

We should notice that a function that is only *semi-definite positive* but not *definite positive* can still be a valid covariance function for a stationary Gaussian process. This means that, in general, the variance-covariance matrix $(\mathrm{cov}(s_i, s_j))_{1 \le i,j \le n}$ for n distinct points $s_1, ..., s_n$ can be singular. However, it would be convenient (for kriging, for example) if we could restrict covariance matrix to be non-degenerate. Fortunately, this is the case for all covariance functions considered below ([42, p. 64] shows a sufficient condition for this to be satisfied).

Covariance functions or variograms are important in the specification of a spatial model, because their regularity implies the regularity of the random process (see [43] for more thorough treatment). For instance, a stochastic process is continuous in mean square at a position $s_\star$, if and only if $c(s, s')$ is continuous at $s = s' = s_\star$. As a reminder, mean-square continuity means that $\mathbb{E}(|Z(s_n) - Z(s_\star)|^2) \to 0$ for all sequence $(s_n)_n$ convergent to $s_\star$. For a stationary process, this is equivalent to the covariance function C being continuous at 0.

In a similar fashion, the k-th order partial derivative $\partial^k Z(s)/\partial s_{i_1} ... \partial s_{i_k}$ exists as a mean square limit, if and only if $\partial^{2k} C(s)/\partial^2 s_{i_1} ... \partial^2 s_{i_k}$ exists at $s = 0$.

Several classes of covariance functions are extensively studied and commonly used in the Gaussian process and spatial statistics literature. When studying an isotropic covariance function, two parameters are often considered

- the *sill*, which is $C(0)$, the value of the covariance function at 0;
- and a scale parameter, sometimes called the *range*, which is the minimal distance at which we can consider that the covariance is 0.

For a semivariogram, the sill is the maximal semivariance, and the range is the distance at which the semivariance stabilizes to the sill.

Below, a number of common isotropic stationary variograms are presented (see, for example, [44, Chp. 5] for other variograms). Note that all of these variograms are isotropic regardless of the dimension of the index space, which is not true for general covariance functions.

- Exponential variogram: $\gamma(h) = c(1 - \exp(-\frac{h}{a}))$, where c is the sill, a is a scale parameter, controlling how quickly $\gamma(h)$ approaches the sill. The range of exponential variogram is $+\infty$ in theory. Since its derivative at 0 is not zero, this variogram is appropriate for processes with high variation at a small scale.
- Gaussian variogram: $\gamma(h) = c(1 - \exp(-\frac{h^2}{a^2}))$, where c and a are the sill and the scale parameter. The major difference with exponential variogram is that, seen as a function defined on the real line, Gaussian variogram is infinitely differentiable at 0. Therefore, the associated process is also infinitely differentiable in mean square.
- Matérn variogram: $\gamma(h) = c(1 - \frac{2^{1-2\nu}}{\Gamma(\nu)}(\frac{\sqrt{2\nu}h}{a})^\nu K_\nu(\frac{\sqrt{2\nu}h}{a}))$ (see [45] for more detail). Additionally to the sill and the scale parameter, Matérn class of variograms has a regularity parameter ν ($\Gamma(\cdot)$ and $K_\nu(\cdot)$ are the gamma function and the modified Bessel function of order ν). An interesting property of Matérn variogram is that the associated process is k-times mean square differentiable for all $k < \nu$. Therefore, by setting the value of ν, the regularity of the stochastic process is controlled. Another feature is that when $\nu = p + \frac{1}{2}$ where p is an integer, the general formula becomes the product of an exponential function with a p-order polynomial.
- Nugget effect: $\gamma(h) = b$ if $h \neq 0$ and $\gamma(0) = 0$. The nugget effect is often added to other variograms. If it is used alone, the random Gaussian process is a white noise, with no spatial correlation. The nugget effect is introduced into spatial modeling because empirically, an estimated variogram rarely has 0 value at $h = 0$. It is often interpreted as resulting from a measurement error on the data.

The discussion above established that variogram is a useful tool in geostatistics. However, in most empirical problems, the variogram is unknown. It has to be estimated from the data, before being used in an application.

Without imposing a functional form for the covariance, one can begin by estimating an empirical variogram. The semivariance at different distances is estimated on the data, while the analytical formula of the variogram is not explicit. In the pioneer work of Matheron [46], the following estimator is proposed:

$$2\hat{\gamma}(h) = \frac{1}{|N(h)|}\sum_{N(h)}(Z(s_i) - Z(s_j))^2, \quad N(h) = \{(i,j)|s_i - s_j = h\}.$$

For irregularly spaced data, we can either be interested in the scatter-plot of semivariance with respect to distance, or group pairs of observations into bins of distance.

To conduct a spatial statistics study, the empirical variogram is estimated and visually examined first. This helps to decide on the family of covariance function, and whether or not to

adjust the anisotropy. In a second place, a chosen parametric covariance function model fitted on the data. This establishes an estimation of covariance between all potential points and can be used later in kriging.

3.2 Kriging in univariate change of support problem

3.2.1 Kriging

An important task in spatial statistics is the prediction of the variable of interest at locations not present in the data. Under different contexts, the interpolation of spatially-index data is called kriging, Gaussian process regression, or Wiener–Kolmogorov prediction. This and the next section present the basic techniques of kriging. More detailed description of these methods can be found in many reference books or lecture notes ([41], [47]).

As in time series, the kriging predictor falls under the category of *best linear unbiased predictor* (BLUP). Namely, it is a weighted average of the data, with expected value equal to that of the process at this location, of minimal mean squared residuals.

Consider a Gaussian process $\{Z(s) : s \in \mathcal{D}\}$, with mean function m and stationary covariance function C. This actually means that the residual component of the process, $Z(s)-m(s) \equiv R(s)$ is a second-order stationary process of mean 0 and of covariance function C. Suppose that this process is observed at n different locations $\{s_1, ..., s_n\}$. An unbiased linear predictor $\hat{Z}$ of Z at location s has the form:

$$\hat{Z}(s) - m(s) = \sum_{i=1}^{n} \lambda_i (Z(s_i) - m(s_i)),$$

where λ_i is the weight of the i-th observation. Define the residual of the estimator by $\hat{R}(s) \equiv \hat{Z}(s) - m(s)$. The weights are then to be determined in order to minimize the quadratic risk, $\mathbb{E}[(\hat{Z}(s) - Z(s))^2] = \mathrm{Var}(\hat{Z}(s) - Z(s)) = \mathrm{Var}(\hat{R}(s) - R(s))$.

The three kriging methods presented in this section differ in their treatment of the mean function $m(\cdot)$. An empirical comparison of these methods on the log-density of population can be found in Section 3.5.

Simple kriging The simplest case of kriging is called simple kriging. In this scenario, the intrinsically stationary process $\{Z(s) : s \in \mathcal{D}\}$ considered has a constant known expected value, $m(s) = \mu$, for all $s \in \mathcal{D}$. The quadratic risk can be written as:

$$\begin{aligned}
\mathrm{Var}(\hat{R}(s) - R(s)) &= \mathrm{Var}(\sum_{i=1}^{n} \lambda_i R(s_i) - R(s)) \\
&= \mathrm{Var}(\sum_{i=1}^{n} \lambda_i R(s_i)) + \mathrm{Var}(R(s)) - 2\mathrm{cov}(\sum_{i=1}^{n} \lambda_i R(s_i), R(s)) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j C(s_i - s_j) + C(0) - 2\sum_{i=1}^{n} \lambda_i C(s_i - s).
\end{aligned}$$

The minimization of the quadratic risk is an unconstrained quadratic optimization that can be solved analytically. Note by $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_n)^T$ the column vector of weights, $\mathbf{K} = (C(s_i - s_j))_{1 \le i,j \le n}$ the matrix of mutual covariance between locations, and $\mathbf{k} = (C(s_1 - s), ..., C(s_n - s))^T$ the column vector of covariance between s and the observations. Because of the positive-definite

property of C, the matrix $\mathbf{K}$ is also positive definite, and therefore invertible. Then it becomes clear that minimizing the quadratic risk is equivalent to minimizing

$$\boldsymbol{\lambda}^T\mathbf{K}\boldsymbol{\lambda} - 2\boldsymbol{\lambda}^T\mathbf{k} = (\boldsymbol{\lambda} - \mathbf{K}^{-1}\mathbf{k})^T\mathbf{K}(\boldsymbol{\lambda} - \mathbf{K}^{-1}\mathbf{k}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}.$$

The optimal weights is therefore $\boldsymbol{\lambda} = \mathbf{K}^{-1}\mathbf{k}$.

It can be noticed that the optimal weights do not depend on the value of observations $Z(s_i)$, only the covariances.

In practice, if a large number of observations are available, the inversion of matrix $\mathbf{K}$ can be cumbersome. There might also be numerical problems with covariances calculated on an estimated variogram. In that case, a form of *local kriging* can be considered, where only the closest neighboring observations are used to provide prediction of a given location. The procedure remains essentially the same.

Ordinary kriging In the ordinary kriging case, the mean function of the Gaussian process, $m(s)$ is unknown. An additional assumption of constant local mean is added: for all neighboring locations involved in kriging, the expected value is unknown but constant. Namely, for all $1 \leq i \leq n$, $m(s_i) = \mu = m(s)$, where μ is an unknown parameter. One way to specify a linear predictor

$$\hat{Z}(s) - \mu = \sum_{i=1}^{n} \lambda_i (Z(s_i) - \mu)$$

without actually knowing μ is to impose that $\sum_{i=1}^{n} \lambda_i = 1$, so that $\hat{Z}(s) = \sum_{i=1}^{n} \lambda_i Z(s_i)$.

In this case, the quadratic risk is minimized under the linear equality constraint $\mathbf{1}^T\boldsymbol{\lambda} = 1$. The minimizer of this problem is the solution of the linear system

$$\begin{pmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \alpha \end{pmatrix} = \begin{pmatrix} \mathbf{k} \\ 1 \end{pmatrix},$$

where α is a Lagrange multiplier to impose the constraint.

Universal kriging Now generalize the model to include a p-dimensional explanatory variable $\mathbf{x}(\cdot)$ available everywhere in $\mathcal{D}$, especially at observation locations and the kriging target. The mean function has the form

$$m(s) = \mathbf{x}(s)^T\boldsymbol{\beta}, \forall s \in \mathcal{D},$$

where in $\mathbf{x}(\cdot)$ there is potentially a constant component, and $\boldsymbol{\beta}$ is an unknown regression coefficient vector.

Similarly to ordinary kriging, we need to specify

$$\hat{Z}(s) - \mathbf{x}(s)^T\boldsymbol{\beta} = \sum_{i=1}^{n} \lambda_i (Z(s_i) - \mathbf{x}(s_i)^T\boldsymbol{\beta}),$$

without knowing $\boldsymbol{\beta}$. This is equivalent to specify $\hat{Z}(s) = \sum_{i=1}^{n} \lambda_i Z(s_i)$ under the constraint

$$\mathbf{x}(s) = \sum_{i=1}^{n} \lambda_i \mathbf{x}(s_i) \equiv \mathbf{X}\boldsymbol{\lambda},$$

where $\mathbf{X}$ is a p-by-n matrix combining explanatory variable at all observations.

This constraint leads to a p-dimensional Lagrange multiplier $\boldsymbol{\alpha}$. The minimizer of quadratic risk under the constraint is the solution of the linear system

$$\begin{pmatrix} \mathbf{K} & \mathbf{X}^T \\ \mathbf{X} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{k} \\ \mathbf{x}(s) \end{pmatrix}.$$

Variance of kriging predictors Because the processes are specified as Gaussian, the variance (which is equal to the quadratic risk in the case of unbiased predictor) of a linear interpolation method can be calculated with ease. In the case of kriging methods presented above, it only depends on the position of the observations and predictions, and not on the value. This provides a measure of uncertainty in the predictions obtained by kriging methods.

3.2.2 Block kriging

Kriging methods are the most elementary way to extrapolate observations to unobserved positions. However, to solve the change of support problem, one often needs to deal with average value of a process on areal support. To deal with this scenario, block kriging can be used.

Consider n disjoint areas in the domain, $B_1, ..., B_1 \subset \mathcal{D}$. The observations are the average of the process $Z(\cdot)$ on these are:

$$Z(B_i) = \int_{s \in B_i} Z(s) ds, \quad \forall 1 \leq i \leq n.$$

These integrals are defined with minimal regularity (for example $Z(\cdot)$ being continue on path and B_i compact. This is the case for all covariance functions considered in this document). The target of kriging, $Z(B)$, is the average on a new area B.

Because of the linearity of the averaging operation, the calculation in the previous section applies naturally on the case of block kriging, with minimal assumptions of regularity. Instead of covariance between points, integrals of the covariance function on areas are now used:

$$C(B_i, B_j) = \int_{B_i} \int_{B_j} C(s_i - s_j) ds_i ds_j.$$

In universal block kriging (the most general case), the explanatory variables also need to be integrated. In that case, the optimal linear weights λ are the solution of the linear system

$$\begin{pmatrix} \mathbf{K} & \mathbf{X}^T \\ \mathbf{X} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{k} \\ \mathbf{x}(B) \end{pmatrix},$$

where the elements of $\mathbf{K}$, $\mathbf{X}$, and $\mathbf{k}$ are all integrated.

In order to solve this kriging equation, it is necessary to insure that the block covariance matrix, $\mathbf{K}$, is invertible. This can be difficult for general covariance models.

In practice, integrals of the covariance function can be approximated numerically by dividing the areas on a regular grid. To limit the time spent on these integrals, it is often interesting to use local kriging.

Bayesian estimation and prediction Although the geostatistical methods are presented here in a frequentist point of view, Bayesian estimation is actually rather prevalent in the spatial statistics literature. Because of the hierarchical nature of spatial and spatio-temporal models, Bayesian estimation methods such Markov chain Monte Carlo (MCMC) algorithms are

very popular in applied studies (see [48], [49] for a detailed description of these methods and numerous examples). However, such methods can be intricate to implement, and by definition MCMC is often expensive in computation time. This makes MCMC methods difficult to scale.

3.2.3 Areal interpolation beyond kriging

Areal interpolation Historically, the area-to-area change of support problem is solved by areal interpolation [50], without explicit generative modeling assumptions. In this procedure, the interpolation weights are obtained proportionally to the area of the blocks considered.

This line of methods is further extended [51] into areal regression models, where the relationship between a co-variable (known everywhere) and the variable of interest is estimated in a regression model. This relation is then used to produce estimation of the variable on an overlapped area. Areal regression models are generalized in a Bayesian context by [52]. For more historic context in general change of support problems, see [53].

Spatial statistical model on areal data A more model-based approach to solve areal interpolation is by considering the observations as the average value of the variable of interest, $Z(D_1), ..., Z(D_n)$, on a collection of subsets, $D_1, ..., D_n$, of the domain $\mathcal{D}$. ([41, Chap. 6], [54, Chap. 9], [48, Chap. 4]).

Instead of deriving a block correlation matrix by integrating on an underlying Gaussian process, as in block kriging, in this approach, the relationship between area is derived directly from the spatial relationships of the blocks. The blocks are viewed as vertices of a graph, whose edges can be defined on either contiguity or by setting a distance threshold. The edges can also be weighted according to relevant criteria of the variable of interest [54].

Once the covariance structure of blocks is specified, the joint distribution of $Z(D_1)$, ..., $Z(D_n)$ can be estimated from the data. Often, the Markov property is assumed for the process, so that the theory of Markov random field can be used in the model specification and estimation (see [48] and references within for a more detailed exposition). Moreover, it is often possible to infer the distribution of the value on a target block, given the estimated model. This provides an interesting way to solve area-to-area change of support problem, especially when combined with Monte Carlo simulations ([55, Chap. 7]).

Multi-scale models Additional structure can be added into a spatial model on areal data to fit multi-scale data. Tree models are particular adapted for integrating data from several resolutions ([56], [57]). The connection between different grid levels is introduced to link multiple nested spatial areal models, under the same graphical model representation. Because of the complexity of the model, this line of research mostly uses Bayesian methods for estimation and prediction.

3.3 Multivariate kriging

In universal kriging, exogenous variables are used to explain the mean of the interpolated variable. One way to generalize this model is to include the spatial covariance between different variables. This technique, called cokriging, has a wider range of application as we will see in this section.

3.3.1 Cross-covariance

Formally, cokriging studies a Gaussian process $\{\mathbf{Z}(s) : s \in \mathcal{D}\}$ indexed s in the domain $\mathcal{D}$, and with values in $\mathbb{R}^d$, where d is the number of variables in question. Similarly as in Section 3.1, we define a multivariate Gaussian process as second-order stationary if it has constant mean and a stationary cross-covariance function:

- $\mathbb{E}(\mathbf{Z}(s)) = \boldsymbol{\mu}$, for all $s \in \mathcal{D}$;
- there is a *cross-covariance* function $\mathbf{C}(\cdot)$ such that $\mathrm{cov}(\mathbf{Z}(s_1), \mathbf{Z}(s_2)) = \mathbf{C}(s_1 - s_2)$, for all $s_1, s_2 \in \mathcal{D}$.

In this definition $\boldsymbol{\mu}$ is a d-dimensional vector, and $\mathbf{C}(s_1 - s_2)$ is a d-by-d matrix. Note that the covariance function $\mathbf{C}(\cdot)$ is not restricted to be symmetric: in general, $\mathrm{cov}(Z_i(s+h), Z_j(s)) \neq \mathrm{cov}(Z_j(s+h), Z_i(s))$, for $1 \leq i < j \leq d$.

Similar to the univariate case, not all matrix-valued function can be a valid cross-covariance. The following properties need to be verified:

- the limit of $\mathbf{C}(h)$ at $h = 0$ has to be positive definite;
- $\mathrm{cov}(Z_i(s+h), Z_j(s)) = \mathrm{cov}(Z_j(s), Z_i(s+h))$, in other words $\mathbf{C}(h) = \mathbf{C}^T(-h)$;
- generally, for any n points in $\mathcal{D}$, $s_1, ..., s_n$,

$$\forall \mathbf{a}_1, ..., \mathbf{a}_n \in \mathbb{R}^d, \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{a}_i^T \mathbf{C}(s_i - s_j) \mathbf{a}_j > 0.$$

Cross-covariance functions are a more complicated object than covariance. In particular, it is difficult to build matrix-valued functions satisfying the three properties mentioned above. Two methods are presented below. The rest of approaches are readily available in references (for example in [49, Chap 9]).

Separable model $\mathbf{C}(h) = \rho(h)\mathbf{T}$, where $\rho(\cdot)$ is a univariate correlation function ($\rho(0) = 1$), and $\mathbf{T}$ is a symmetric positive-definite matrix. This model is easy to understand and to estimate, but it can be too strong in practice: the cross-covariance is necessarily symmetric, and all variables have a single spatial-correlation range (that of ρ).

Linear model of coregionalization We make the assumption that $\mathbf{Z}(s) = \mathbf{A}\mathbf{w}(s)$, with $\mathbf{A}$ a d-by-d full rank matrix, and $\mathbf{w}(s)$ a d-dimensional Gaussian process with independent components and unit variance. The cross-covariance of $\mathbf{Z}(\cdot)$ is then

$$\mathbf{C}(h) = \sum_{i=1}^{d} \rho_i(h) A_i A_i^T,$$

with $\rho_i(\cdot)$ the correlation functions of w_i. The cross-covariance matrix is still symmetric, but each component in $\mathbf{Z}(s)$ has a distinct covariance function.

The estimation of linear coregionalization model can be achieved either by an iterative least square method (alternatively vary the parameters in individual correlation functions, and in the matrix $\mathbf{A}$, see [58]), or integrated into a global Bayesian context [59].

In the multivariate case, intrinsically stationary processes are less considered. In fact, the natural generalization of variogram is the cross-variogram $\mathrm{Var}(\mathbf{Z}(s_1) - \mathbf{Z}(s_2))$. This matrix is

always symmetric whether the cross-covariance is symmetric. In order to have an object that contains as much information as the cross-covariance, a pseudo cross-variogram is defined as $(\mathrm{Var}(Z_j(s_1) - Z_i(s_2)))_{1\leq i,j\leq d}$ [60]. This quantity arises naturally in the variance of the cokriging predictor as the variogram in univariate kriging.

3.3.2 Cokriging

Once specification and estimation of cross-covariance is pinpointed, the cokriging equation itself can be derived in a similar fashion as in universal kriging.

Under constant mean, a linear predictor of one component Z_1 of $\mathbf{Z}$ at s, with data available at $s_1, ..., s_n$, is of the form

$$\hat{Z}_1(s) - \mu_1 = \sum_{i=1}^{n}\sum_{j=1}^{d} \lambda_{ij}(Z_j(s_i) - \mu_j).$$

In order to eliminate the unknown means and guarantee unbiasedness, the linear constraints,

$$\sum_{i=1}^{n} \lambda_{i1} = 1, \quad \sum_{i=1}^{n} \lambda_{ij} = 0, \quad \forall 2 \leq j \leq d,$$

are imposed.

The quadratic risk minimization leads to a similar matrix equation as in ordinary and universal kriging with bigger matrices. Note by $\mathbb{K}$ the nd-by-nd cross-covariance matrix,

$$\mathbb{K} = \begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{21} & \cdots & \mathbf{K}_{d1} \\ \mathbf{K}_{12} & \mathbf{K}_{22} & \cdots & \mathbf{K}_{d2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}_{1d} & \mathbf{K}_{2d} & \cdots & \mathbf{K}_{dd} \end{pmatrix},$$

where $\mathbf{K}_{ij}$ is a n-by-n cross-covariance matrix between the i-th and j-th component of the process, on the n positions of the data. The kriging coefficients and covariance between target position and available positions are stacked into nd-dimensional vectors in a corresponding fashion, namely,

$$\boldsymbol{\lambda} = \begin{pmatrix} \lambda_{11} \\ \lambda_{21} \\ \vdots \\ \lambda_{n1} \\ \lambda_{21} \\ \vdots \\ \lambda_{nd} \end{pmatrix}, \quad \mathbf{k} = \begin{pmatrix} \mathrm{cov}(Z_1(s), Z_1(s_1)) \\ \mathrm{cov}(Z_1(s), Z_1(s_2)) \\ \vdots \\ \mathrm{cov}(Z_1(s), Z_1(s_n)) \\ \mathrm{cov}(Z_1(s), Z_2(s_1)) \\ \vdots \\ \mathrm{cov}(Z_1(s), Z_d(s_n)) \end{pmatrix}.$$

By defining a d-dimensional Lagrange multiplier $\boldsymbol{\alpha}$, the kriging equation is

$$\begin{pmatrix} \mathbb{K} & \mathbb{I}_d \otimes \mathbf{1} \\ \mathbb{I}_d \otimes \mathbf{1}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{k} \\ \boldsymbol{e}_1 \end{pmatrix},$$

where $\mathbb{I}_d \otimes \mathbf{1}$ is the Kronecker product between the d-by-d identity matrix and a n-dimensional unit vector, and $\boldsymbol{e}_1$ is the first vector of canonical basis in $\mathbb{R}^d$.

3.4 Change of support in spatio-temporal model

3.4.1 Spatio-temporal covariance and kriging

As in the spatial case, the key to spatio-temporal kriging is the specification of covariance function model. [48, Chap 6] is a recent review of methods in this vein. Here we will focus on the two principal ways to integrate the temporal component into the spatial Gaussian process.

Continuous-time spatio-temporal model Continuously indexed time can be viewed as an additional dimension in the index of the Gaussian process. For an interval I of $\mathbb{R}$, the spatio-temporal process is

$$\{Z(s,t) : s \in \mathcal{D}, t \in I\}.$$

A second-order stationary spatio-temporal process has a constant mean, and a stationary covariance function $C(h,\tau) = \mathrm{cov}(Z(s,t), Z(s+h, t+\tau))$, for all $s \in \mathcal{D}, t \in I$.

As in the purely spatial case, the covariance function has to be positive-definite. The most direct way to achieve this is to treat time as an additional dimension of the index space, and use one of the spatial covariances mentioned in Section 3.1. The underlying assumption would be that time is a dilated dimension, which is often not the case.

A separable spatio-temporal covariance is one that verifies $C(h,\tau) = C_1(h)C_2(\tau)$. This can be insufficient. Typically, if we want to obtain the value of the variable of interest at a moment t at a location s, and if this variable is observed at every other location except s, observations at other moments can simply be tossed away, since just doing spatial kriging already gives the optimal estimation. This condition is often too restrictive, for example, to model the propagation of a wave across the spatial domain.

To generalize a separable spatio-temporal covariance, we can notice that the sum or product of valid covariance functions are still valid covariance function ([48, Theorem 6.1]). This procedure allows us to produce non-separable but fully symmetric covariances easily. These covariance functions are called product-sum models. They are usually defined as $C(h,\tau) = C_1(h) + C_2(\tau) + kC_1(h)C_2(\tau)$, with a joint parameter k, given spatial covariance $C_1(h)$ and temporal covariance $C_2(\tau)$.

The most general way to specify spatio-temporal covariances is to start from a positive spectral measure. As seen in Section 3.1, there is a correspondence between positive-definite functions and positive measures in the frequency space. Therefore, covariance functions can be specified as inverse Fourier transform of positive measures. Moreover, [61] showed that for a function $\alpha : (\omega,\tau) \mapsto \alpha(\omega,\tau)$ defined on $\mathbb{R}^d \times \mathbb{R}$, that is *positive-definite* in τ, the function

$$C(h,\tau) = \int \mathrm{e}^{i<h,\omega>}\alpha(\omega,\tau)d\omega, \quad h \in \mathbb{R}^d, \quad \tau \in \mathbb{R},$$

is a positive-definite function in $\mathbb{R}^d \times \mathbb{R}$. This is an easy way to construct very general covariance functions.

Fitting spatio-temporal functions to data can be done in a similar way as in cokriging, although given the complexity of the model, Bayesian methods are preferred in this context.

Discrete-time spatio-temporal model Time can also be seen as discrete. For each position s_i in the data, a discrete time series $\{Z_t(s_i), t = 1, 2, ..., T\}$ is observed. Often, such situation arises when the data is on a regular temporal rhythm.

Without assumption on temporal stationarity, the covariance structure can be treated in the same way as cross-covariance in cokriging. If temporal stationarity is assumed, then lagged auto-covariance can be estimated on averaging pairs of observations of the same lag.

More sophisticated dynamic models can be considered with an appropriate physical phenomenon, but that would be beyond the scope of the present document (see[Chap 7] [48] and references there within for details).

Computational issues in spatio-temporal kriging Once the covariance function is estimated and applied on the data, the kriging equations is exactly the same as in cokriging. The most important issue in this approach is scalability. Indeed, solving for the kriging weights needs the inversion of the matrix $\mathbb{K}$. In a multivariate dataset, $\mathbb{K}$ is of dimension nd-by-nd, whereas in a spatio-temporal dataset, it is nT-by-nT.

3.5 Application

In order to test the geostatistical methods presented before, two illustrative examples (one purely spatial kriging, and one of spatio-temporal kriging) are presented next. The kriging methods used are implemented in the `gstat` package in `R`. The research article [62] is a presentation of this package. [63] is a general presentation of spatial data analysis in `R`.

3.5.1 Variogram of the density of population in Lyon

In the following, the logarithm of population density (population divided by area) of IRIS in Lyon is used as an illustrating example (Figure 3.1). An IRIS (*Ilots Regroupes pour l'Information Statistique*) is an administration zone division of approximately 2,000 habitants, used by the French official statistics department (INSEE). The logarithm of population density of an IRIS is calculated as

$$\log_{10} \frac{\text{Population in the IRIS}}{\text{Area in the IRIS (km}^2)}.$$

The domain of interest is included in an approximately 10km-by-10km square. Eache IRIS is colored with a color scale that correponds to its population density. As we can see, this data has some obvious discontinuity on the frontier of the IRISes. In the rest of this section, we suppose this density is attached to the geometric center of each IRIS, and try to find an estimation for each point in the area.

Suppose that the model is isotropic (the case of anisotropy is often resolved by applying a re-scale on some axes). Figures 3.2 and 3.3 shows the boxplots of semivariance of distance smaller than 5000 meters, grouped in 15 bins, and the empirical variogram, calculated for the 15 bins.

As can be seen in Figures 3.2 and 3.3, an isotropic variogram is often an increasing function on the positive real line. Actually, $\gamma(\cdot)$ is a symmetric function with respect to $h = 0$ defined on the real line. Therefore, when γ is differentiable at 0, its derivative at 0 has to be 0 too. It can also be noticed that the semivariance somewhat stabilizes after a period of rapid increase. This maximal semivariance is the sill in the geostatics literature. The distance h at which the semivariance stabilizes is the range. On the corresponding covariance function, this is the minimal distance at which the spatial correlation practically disappears.

Figure 3.4 shows the values of exponential, Gaussian, and a 1.5-order Matérn variogram with the same range and sill. The graphics are generated by the `gstat` Package in `R` [62].

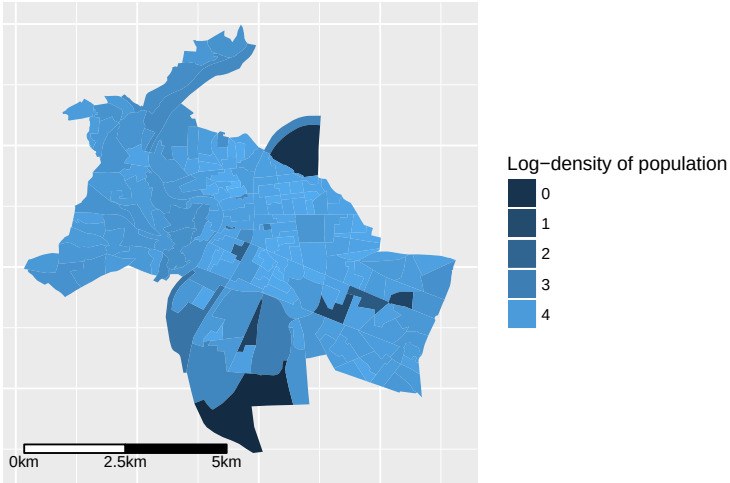


Figure 3.1: *Log-density of population in Lyon.*

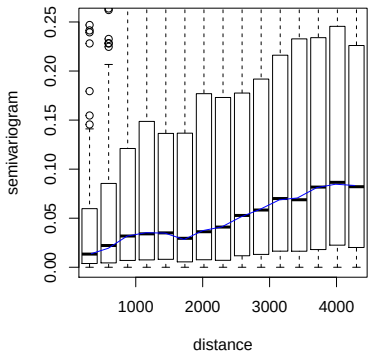


Figure 3.2: *Cloud of empirical semivariance log-density of population*

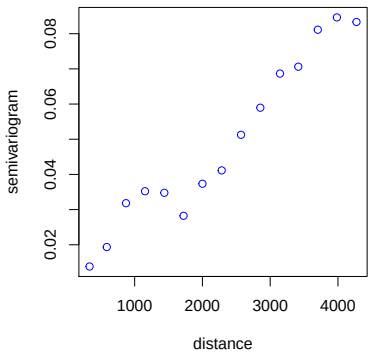


Figure 3.3: *Empirical variogram log-density of population*

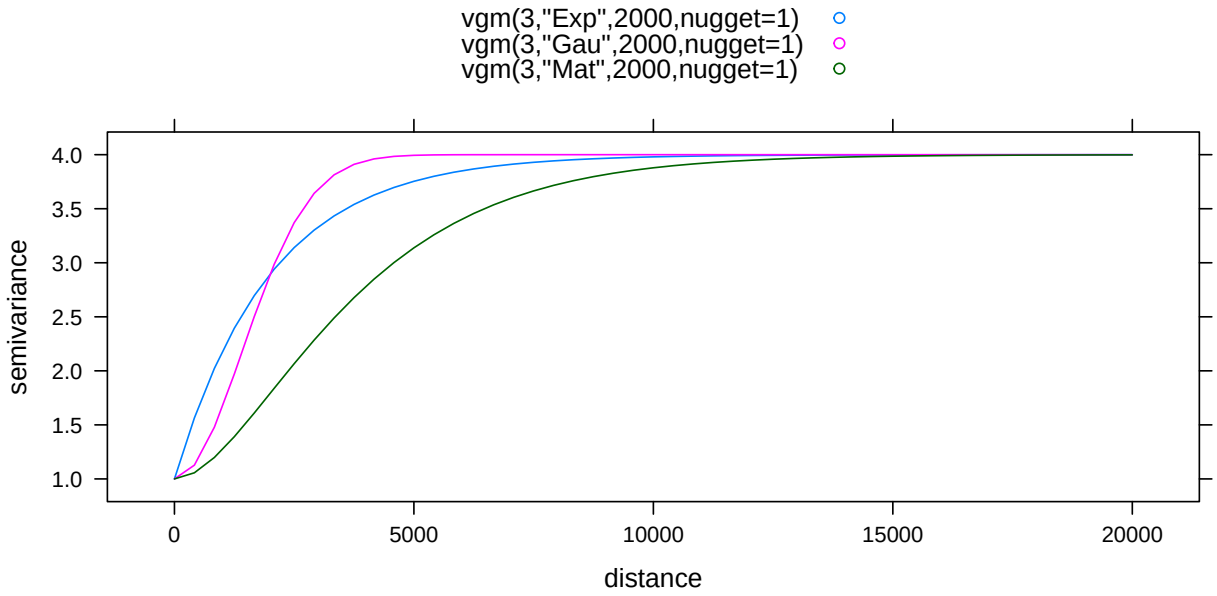


Figure 3.4: *Variogram models included in 'gstat'*

Table 3.1: *Estimated parameters of variogram models*

	Nugget (b)	Sill (c)	Scale (a)
Exponential	0.015	4.469	236729.503
Gaussian	0.019	0.075	3238.211
Matern	0.020	0.215	3731.366

Once the geostatistician made a decision on which variogram to use, a fitting method is used to estimate the parameters in the variogram model. Several methods are studied and implemented in statistical softwares (for example `gstats`). These methods are mostly maximum likelihood methods, based on the Gaussian assumption of the data, or some weighted version of least square estimation, by minimizing a squared empirical risk. Figure 3.5 shows these variograms models on the log of population density, estimated by the default fitting method implemented in `gstat` (least square weighted by the number of observation points divided by squares of the distance). The estimated sill, nugget, and scale parameters are reported in Table 3.1.

3.5.2 Kriging of a univariate variable

A 100m-by-100m grid is created on the original spatial domain. Figure 3.6 shows the log-density of population assigned to the geometric center of each IRIS. The population density of other squares are estimated by universal kriging described in Section 3.2: the mean of the Gaussian process is assumed to be constant but unknown.

Figure 3.7 shows the kriging predictions for each little squares on the grid. The three estimated variograms of Figure 3.5 are used. In comparison, a linear interpolation based on triangulation of the data points is also shown. Note that in linear interpolation, only points within the convex hull of the data points are evaluated.

The kriged value surface is smooth for all three variogram models, much smoother than linear interpolation. Although the exponential covariance is not differentiable at zero (therefore the kriged value surface is not differentiable either), it is only slightly less smooth than Gaussian and Matérn covariance models. At points where observations are available, there is sometimes

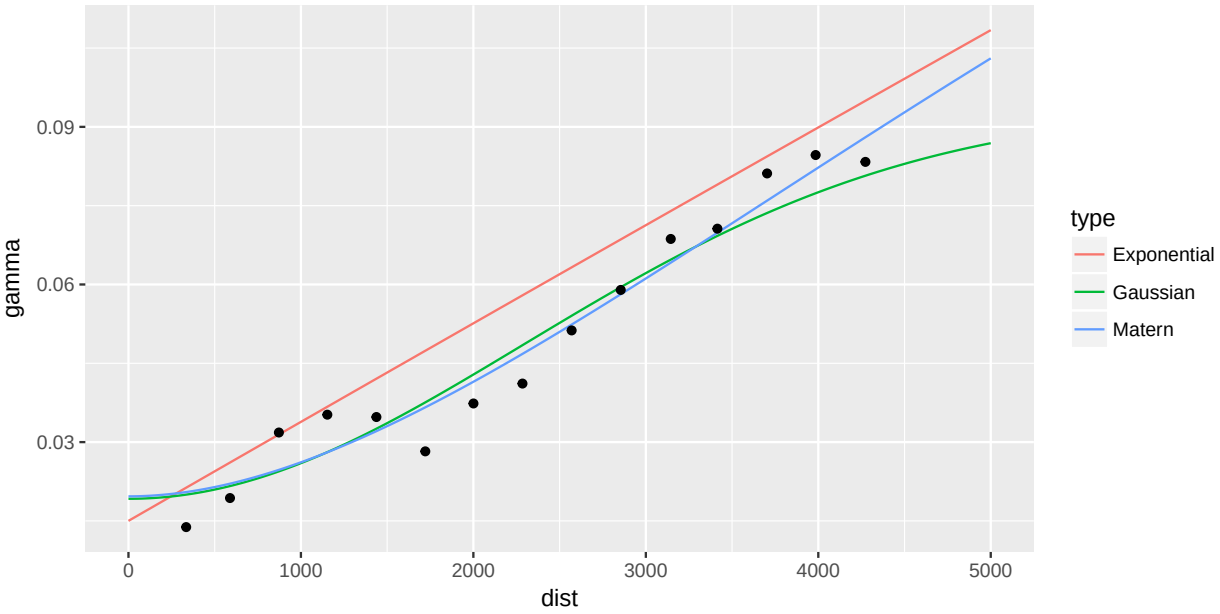


Figure 3.5: *Fitted variogram models*

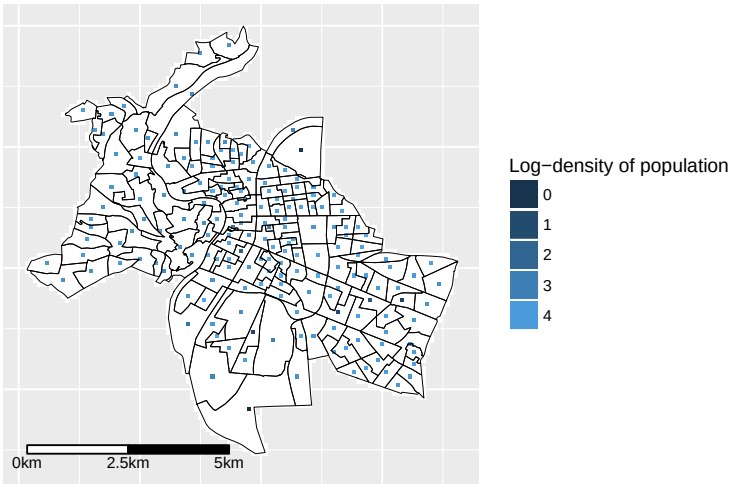


Figure 3.6: *Grid on the domain of interest. The log-density of population is assigned to the square containing the geometric center of each IRIS.*

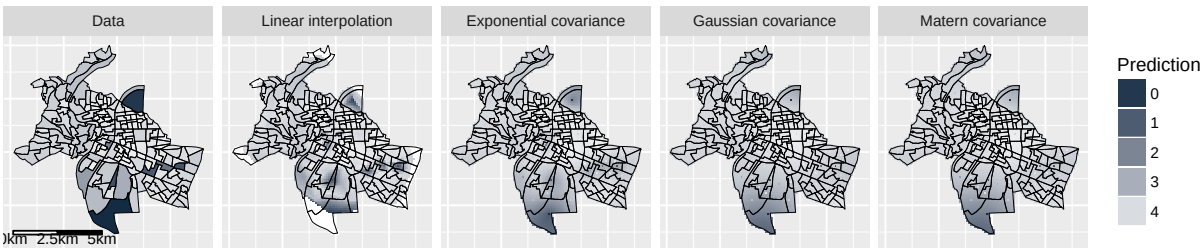


Figure 3.7: *Kriging prediction value of log-density of population.*

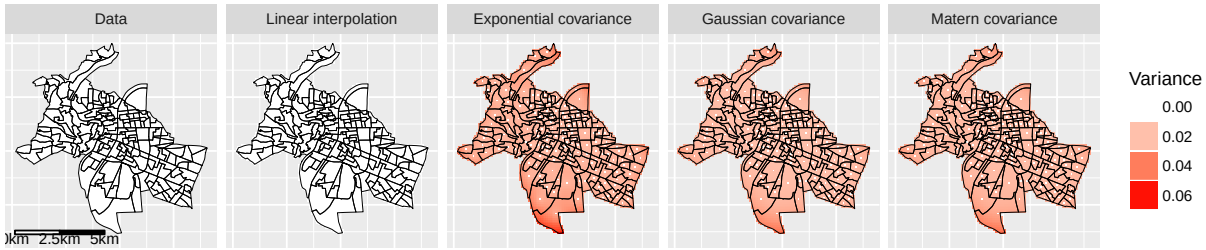


Figure 3.8: *Kriging variance of log-density of population.*

a discrepancy even with its nearest neighbors, for example the geometric center of the most southern IRIS.

Figure 3.8 shows the kriging variance for the grid. The variance is evidently larger at the edge of the domain of interest: the distance with any observations being larger, little information is provided for these points.

3.5.3 Block kriging

The data used in this example is area-based. The reported log-density of population is the mean of the IRIS. Therefore, the point kriging method used in the previous section is a simplification of the actual data. In this section, block kriging and areal interpolation is compared to the last previous. The kriging target is the collection of blocks obtained by triangulation of the position of local substations.

As presented in Section 3.2, the major difference between block kriging and classical point-to-point kriging, is that the covariance function is evaluated and integrated on the source and/or target blocks. In `gstat` package, a numerical approximation is introduced to calculate point-to-area covariance, by dividing target blocks on a grid. For the moment, block-to-block covariance is not implemented in the kriging feature of the package.

To be the most accurate, with area-based data, the covariance model should be fitted considering the empirical covariance as block-based as well. Here the point-based fitted covariance models are used.

In Figure 3.9, the kriged mean value of target blocks are shown, so are the IRIS frontier (the source data are assumed to be point-based). Similar to the previous section, areal interpolation produces a value surface that is much less smooth than the kriged value surface. Exponential covariance is the least smooth of the three covariance functions, but the difference is rather small.

In Figure 3.10, the variance of kriged value is shown. Block kriging methods have smaller variance than point kriging, which is a natural, since it is the block averages that is estimated.

3.5.4 Kriging of spatio-temporal data

In this section, the simulated weekly electricity consumption of MV feeders with local substations overcrossing IRIS of Lyon for the 2010-2012 period is studied.

The simulations are produced in the following manner. We have the geographical positions of the low-voltage (LV) distribution transformers connected to the feeders. These transformers are therefore positioned inside the boundaries of each IRIS. For each transformer, we choose a client type according to socio-demographic information of the IRIS: the ratio between the number of apartments and contractual power (CP), and the ratio between the number of warehouses and

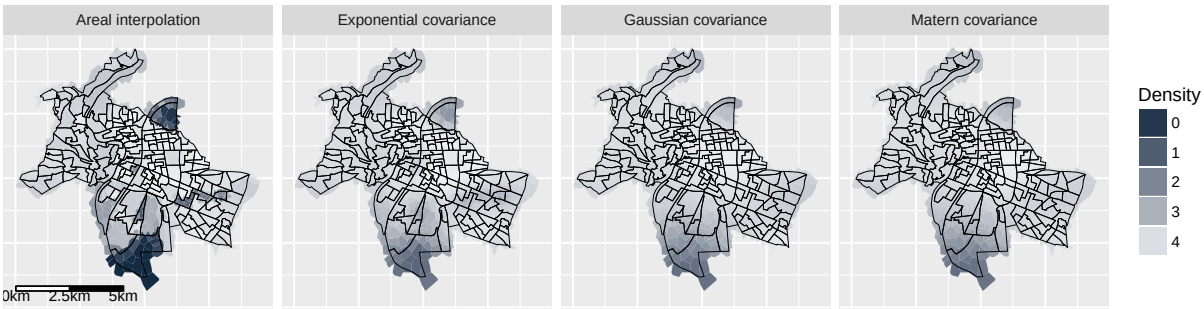


Figure 3.9: *Block-kriging value of log-density of population.*

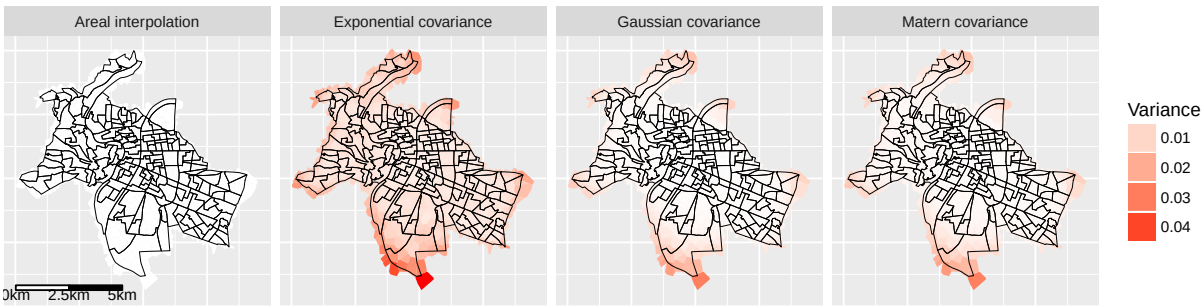


Figure 3.10: *Block-kriging variance of log-density of population.*

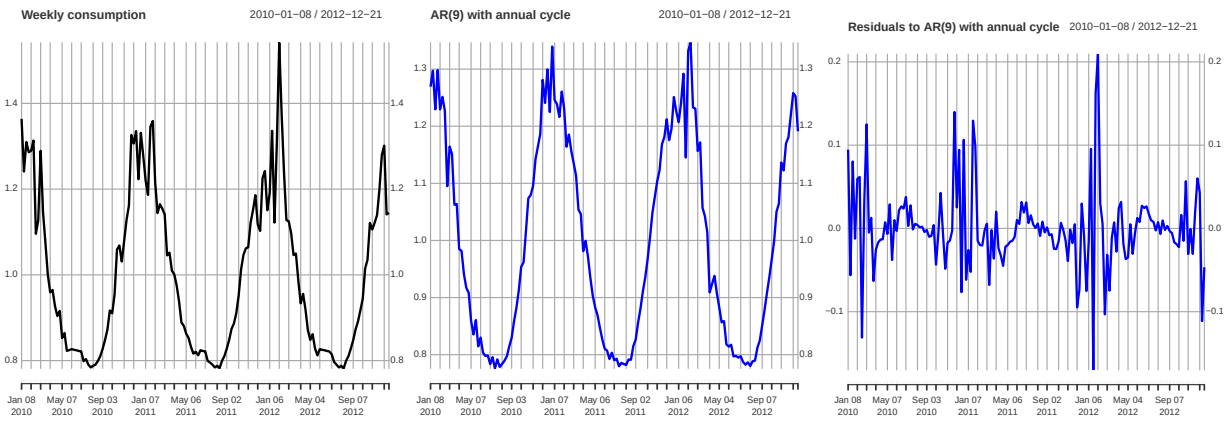
CP. Using these two ratios, the transformer is assigned a client type whose hourly consumption time series is then simulated with regulated French profiles of typical consumers [64]. We plug real local temperature and calendar variables for the 2010-2012 period into the profiles to simulate hourly consumption shape of each LV transformer, and multiply it by the total contractual power of the transformer. The IRIS aggregation is calculated by summing over transformers insider its boundary, while the feeder aggregation is obtained by summing over transformers connected to it. The weekly consumption is obtained by taking the averages over hourly simulations. This dataset is also used in the next chapter. See Section 4.3.1 for more details on the simulation process.

The data spans 156 weeks (three years), and 114 MV feeders are included. Figure 3.16 shows the first four periods of data.

To understand the temporal profile of the data, the spatial average of the 122 time series is studied. This mean series has a strong periodic pattern (Figure 3.11a). Given the sample autocorrelation and partial autocorrelation (Figure 3.12), an AR(9) model with an annual cycle (Figure 3.11b) is chosen to estimate the spatial averages. A period of 9 weeks is used since it is large enough to cover seasonal dependency. We can see that the annual cycle explains quite well the time series of spatial averages.

We first apply purely spatial block kriging to each of the four periods (with the temporal trend removed), before comparing spatio-temporal kriging to this benchmark. Figure 3.13 shows the empirical variogram and a fitted order-1.5 Matérn variogram model for each of the four weeks. The parameters of the fitted variogram model are reported in Table 3.2. It can be observed that the variograms are different for the four weeks, which suggests a joint spatio-temporal modeling of such a covariance structure can be difficult.

An empirical spatio-temporal variogram is estimated on the data, minus the fitted time series model (the leftmost panel in Figures 3.14, 3.15). The semi-variance increases quickly and plateaus out once the spatial distance exceeds a threshold of less than 1000 meters, suggesting a rather small spatial covariance. On the contrary, increase temporal lag does not increase much

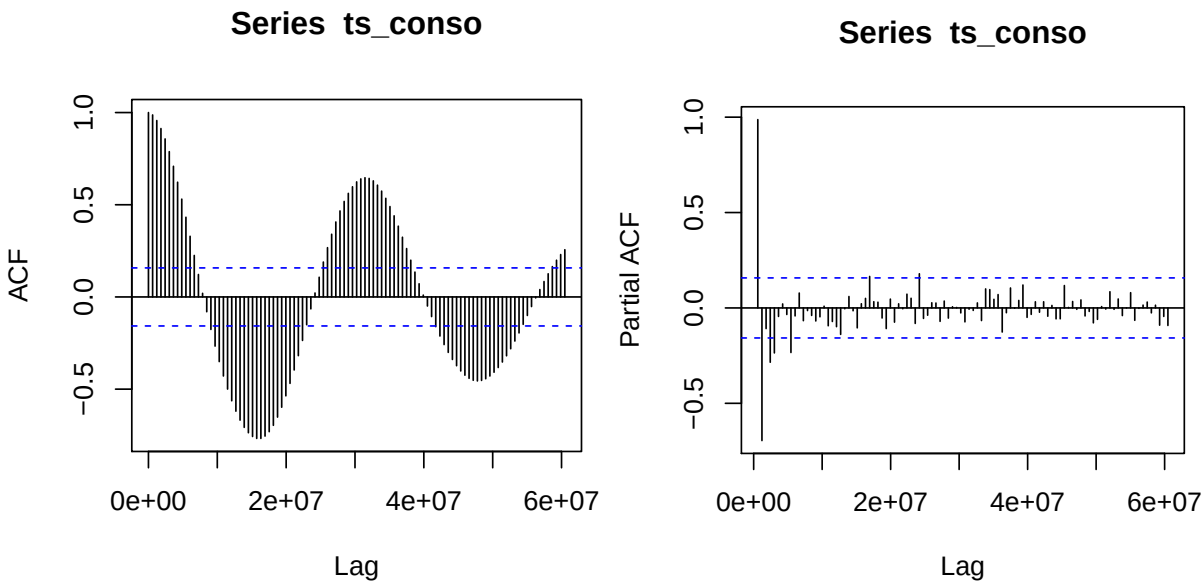


(a) Spatial average of 122 weekly electricity consumption series.

(b) Fitted AR(9) model with annual cycle on the average electricity consumption.

(c) Residuals of the AR(9) model with annual cycle

Figure 3.11: *Spatial average of the weekly consumption and an autoregressive model fitted on it.*



(a) Sample autocorrelation function.

(b) Partial autocorrelation function.

Figure 3.12: *Autocorrelation and partial autocorrelation function.*

Table 3.2: *Estimated parameters of Matern variogram models for the first four weeks of consumption data*

Week	Nugget (b)	Sill (c)	Scale (a)
1	0.244	0.081	633.972
2	0.252	0.143	631.750
3	0.283	0.111	455.967
4	0.258	0.137	440.876

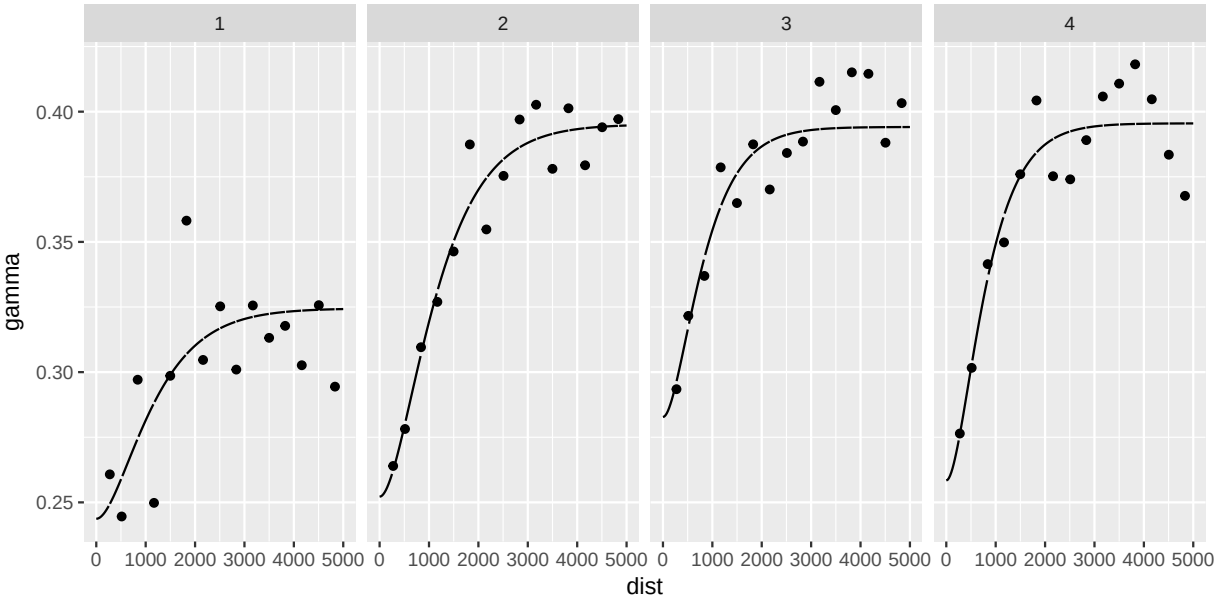


Figure 3.13: *Spatial variograms fitted on the four weeks separately.*

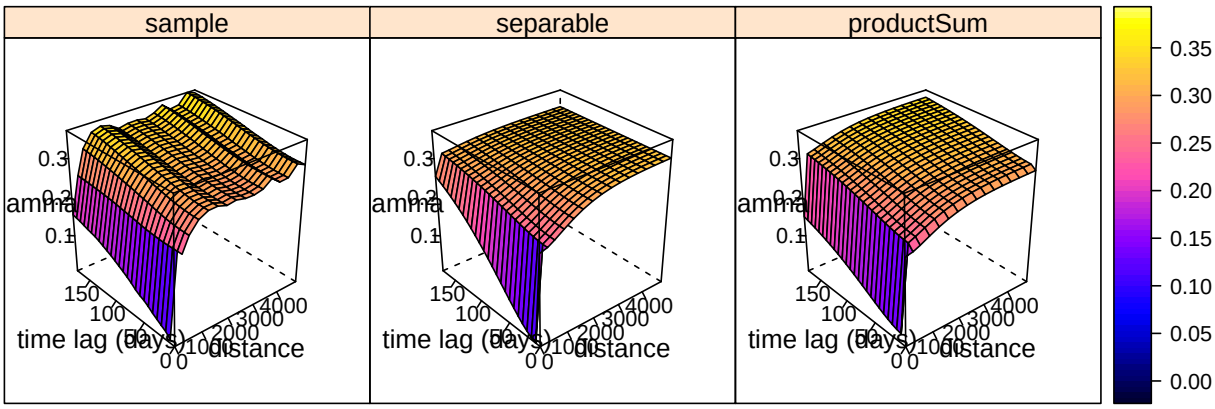


Figure 3.14: *A wireframe plot of the empirical and fitted spatio-temporal variogram.*

the semi-variance, suggesting high temporal covariance.

Two spatio-temporal variogram models are fitted by `fit.StVariogram` function in `gstat`, one separable model and one sum product model (Table 3.3). Both model constructions are provided in `gstat` tool box. The separable model has a joint sill, while the spatial and temporal components have unit sills (as the sum of nugget effect and a Matérn covariance). The parameter k in sum-product model is the weight of the product term. In `gstat`, parameters are optimized using generic optimization function `optim` of `R`. Here the optimization routine is L-BFGS-B, a quasi-Newton optimization method for box domains.

The fitted variograms can be visually compared in Figure 3.14 and 3.15. It can be noticed that it is difficult to fit the decrease in semi-variance around 500m, either by separable or sum-product model. The sum-product model seems more appropriate than the separable model, in that for large spatial distances, the increase of semi-variance when temporal lag increases is kept in fitted model.

```
##         min       max
## x  402208.4  412869.6
## y 4879405.6 4889993.4
```

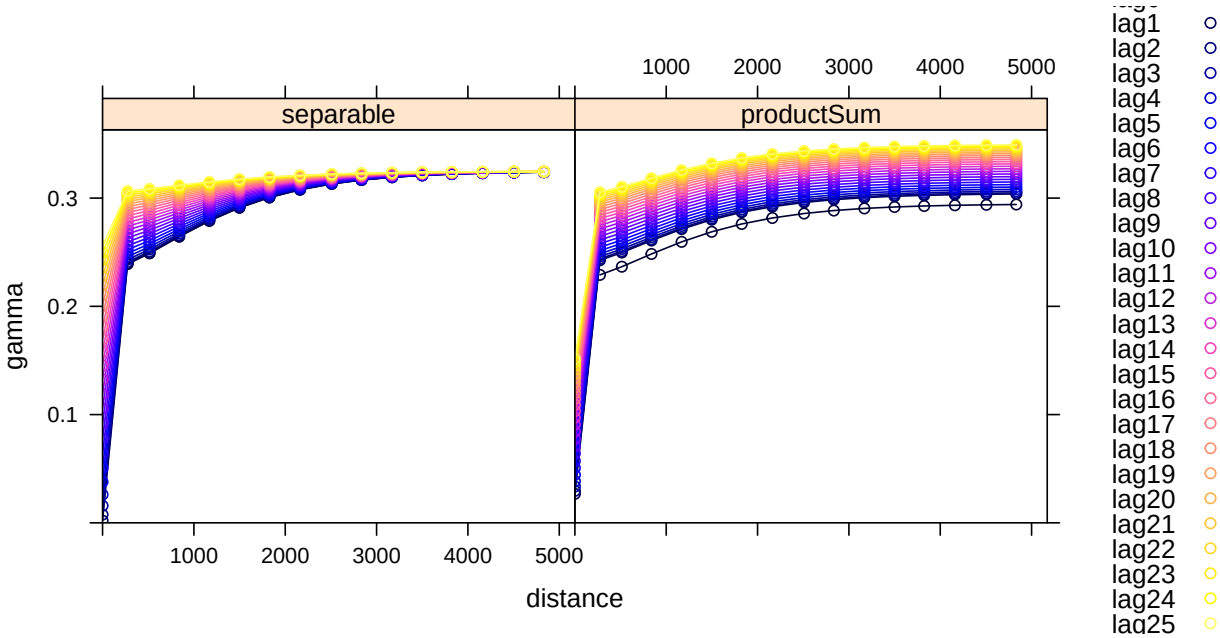


Figure 3.15: *The empirical and fitted temporal variogram by temporal lag.*

Table 3.3: *Estimated parameters of spatio-temporal variogram models*

	Model component	Nugget (b)	Sill (c)	Scale (a)
Separable	Space	0.72	0.28	700
Separable	Time	0	1	60
Separable	Joint	sill	0.32	
Sum-product	Space	0.13	0.04	700
Sum-product	Time	0.01	0.06	60
Sum-product	Joint	k	9.96	

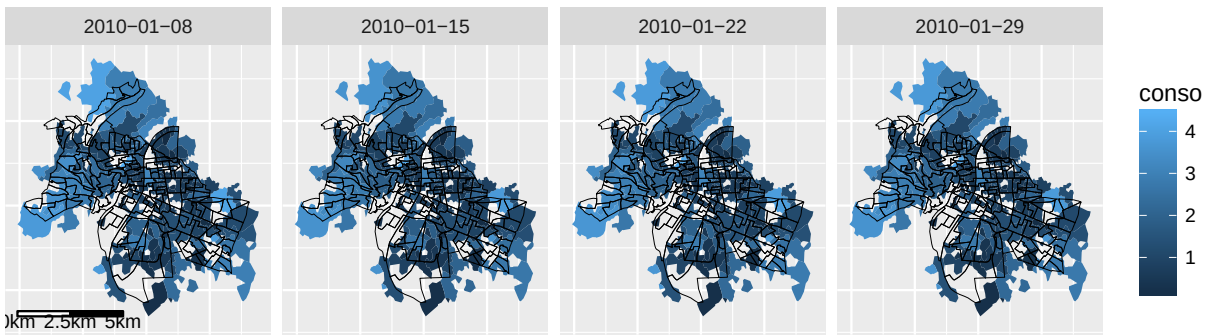


Figure 3.16: *Weekly electricity consumption of the MV feeders of the first four weeks.*

Table 3.4: *Error rates of the kriging estimation compared to the simulation consumption of IRIS (ground truth)*

	Separable	sumProduct	spatialKrige
MAPE	1.614	1.643	1.513
RRMSE	4.479	4.549	4.229

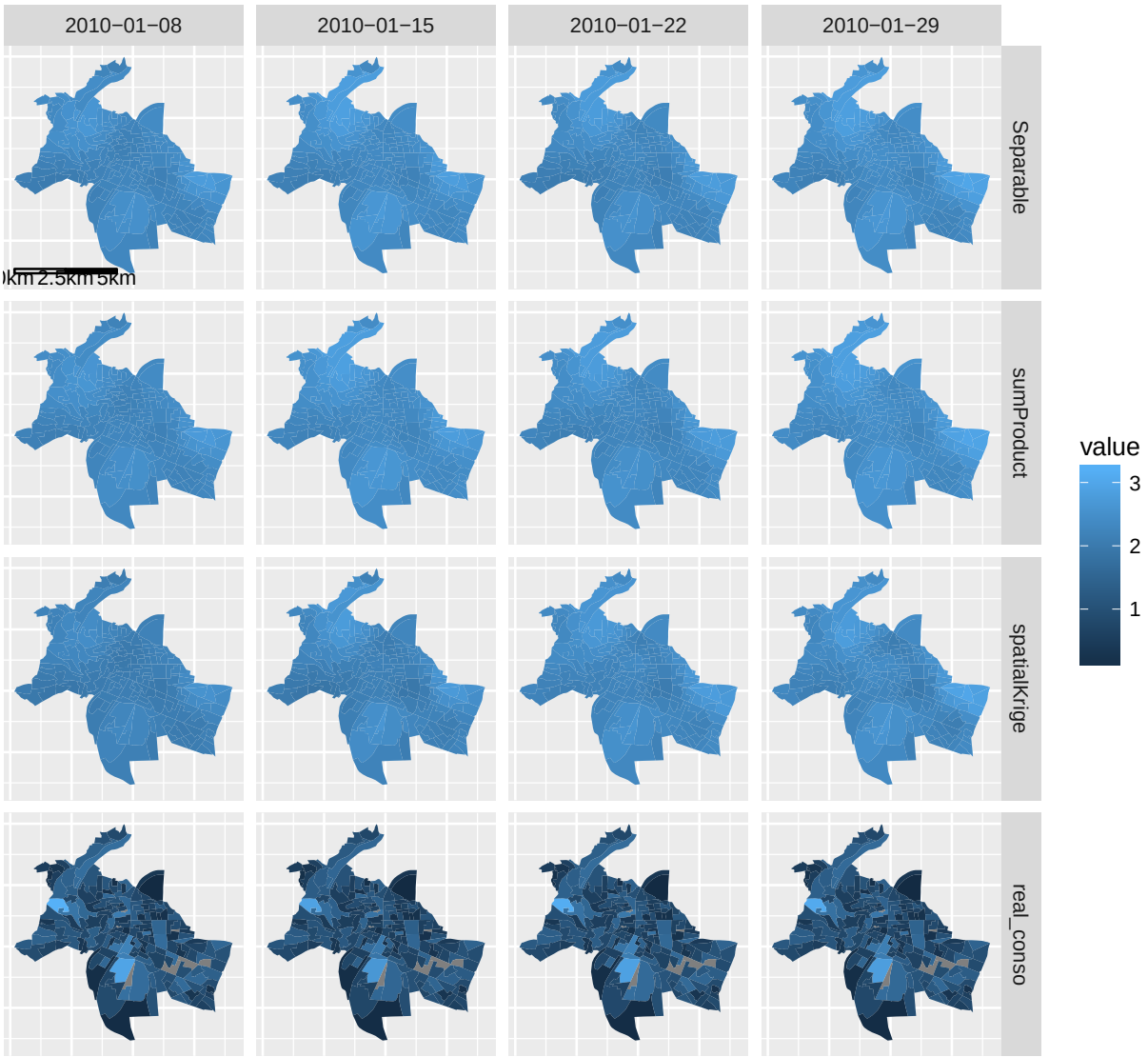


Figure 3.17: *Kriged weekly IRIS electricity consumption of the first four weeks*

Spatio-temporal (point) kriging is applied using the fitted variograms. In Figure 3.17, the spatio-temporally krigged weekly IRIS electricity consumption is represented. The first and the third rows of Figure 3.17 are respectively the separable and product-sum model. Because the kriging computation becomes very expensive for non-separable variograms, only four weeks of data are used in the product-sum model. The second row shows the purely spatial kriged value of the same data. In the fourth row, the actual simulated consumption of the IRIS is plotted (the ground truth).

We observe that compared to the data (Figure 3.16 and the fourth row of Figure 3.17), the kriging values are much more smooth spatially. The temporal variation between the first week and next three weeks is also decreased in the kriging results. The separable covariance does the most spatial and temporal smoothing: the product-sum model seems more smooth spatially on the third date: there is the least contrast between different dates and areas. The product-sum model has the least smooth spatial contrast, while purely spatial kriging is somewhere in the middle.

Error rates of the kriging estimation are shown in Table 3.4. Two error metrics are used here

- the relative root- mean-squared error: $\mathrm{RRMSE}(\mathbf{x}, \mathbf{x}^*) = \frac{\|\mathbf{x}-\mathbf{x}^*\|_2}{\|\mathbf{2}^*\|_F}$,
- and the mean absolute percentage error: $\mathrm{MAPE}(\mathbf{x}, \mathbf{x}^*) = \frac{1}{n}\sum_{t=1}^{n} \frac{|x_t - x_t^*|}{x_t^*}$.

where $\mathbf{x}^*$ is the true vector of dimension n to be estimated, and $\mathbf{x}$ is the estimated value. Indeed, the error rates of the kriging methods are all very high. By examining Figure 3.17, we realise that the kriging estimations are much too smooth compared to the true value, hence the high error rates.

In fact, by examining both the data (Figure 3.16) and the instant spatial variogram (Figure 3.13) that the stationary hypothesis, which we adopted from the beginning of this chapter, is not verified by this dataset. This indicates that although we can obtain an estimation by kriging, if we only model the variation by the covariance in a stationary process, the quality of the estimation can not be very convincing.

3.6 Conclusions

Open questions From the above examination of the spatio-temporal statistics methods, it is seen that:

- An extensive literature in geostatistics and related domain studies the problem of kriging, and the theoretical foundation of this technique has been greatly enriched in the past 20 years.
- Although there is a developed theory, not all of the methods have been implemented in a satisfactory way to solve every instance of kriging problems. In particular, block kriging in a spatio-temporal context, or block-to-block kriging has not really been implemented. There are two reasons for this lack: the first is the relatively high cost of doing block kriging, the other is the lack of application of such methods in geostatistics.
- The methods are general quite slow, especially when the models are complex and MCMC estimation has to be used for estimation.

Inspirations drawn from the spatial statistics literature for our problem In the rest of this thesis, the main objective to be achieved is a computationally feasible method for spatio-temporal interpolation estimating the consumption time series at the IRIS level, from data collected at another spatial division, probably of a similar level or below. From this preliminary examination of the literature, spatio-temporal kriging is not well suited for the problem introduced in the introduction, as tempting as this option sounds. Unlike in physical phenomena studied in geostatistics, electricity consumption is a phenomenon is mainly driven by human activities. Thus using appropriate exogenous variables to correctly model the mean function, probably has more impact to the performance of the methodology, than optimizing over the choice of spatial covariance structure. Moreover, the computational complexity of spatio-temporal kriging is generally too high for the applications considered in this thesis. In the previous section, we were only able to produce spatio-temporal kriging during four periods, whereas in the following chapters, experiments are generally conducted for data involving hundreds or thousands of periods.

However, a number of important lessons are to be kept in mind for the rest of the methods developed:

- Given an appropriate modeling of the trend, studying the spatio-temporal covariance structure of the residuals can probably enhance the model, to deliver better performance.
- The exogenous variables used in the trend modeling can be kriged to be estimated at places different than the data source. This is especially true for weather processes which, by their nature, has a spatio-temporal covariance structure.
- Even if not applied directly, understanding spatio-temporal kriging techniques is a plus for fully comprehending spatial data provided by external sources such as Météo France.

Chapter

4 Spatial estimation of electricity consumption using socio-demographic information

Contents

In this chapter, the electricity consumption data is available in the same way as the previous chapter: individual aggregations different from those that interest us. We'd like to obtain an estimation of electricity consumption in target zones, not with the auto-regressive structure of the consumption process, but with additional information of the source and target zones.

4.1 Introduction

Electric consumption at local level - a city, a village or a block - is an important concern for utilities and grid operators. A grid planner needs load data at a sufficiently small scale to perform spatial load forecasting to optimize maintenance and investments on the grid. With increasingly decentralized generation of renewable energy (for instance, wind or solar power), local consumption is becoming more important for managing the supply-demand balance at the distribution level. Crossed with telecommunication or other utility data at the same level, local electricity consumption can help authorities understand local human activities at a fine-grained temporal level.

In this chapter, we propose a method to estimate past and future electricity consumption at a small temporal scale for target zones with little or no historical consumption data. Two cases are of our interest here: 1. for new substations or feeders, historical data is insufficient for forecasting; 2. for a block or a village, measurements in substations or feeders are not

This chapter is a joint work with Yannig Goude, Georges Hébrail and Nicolas Kong. It is based on the conference paper [1].

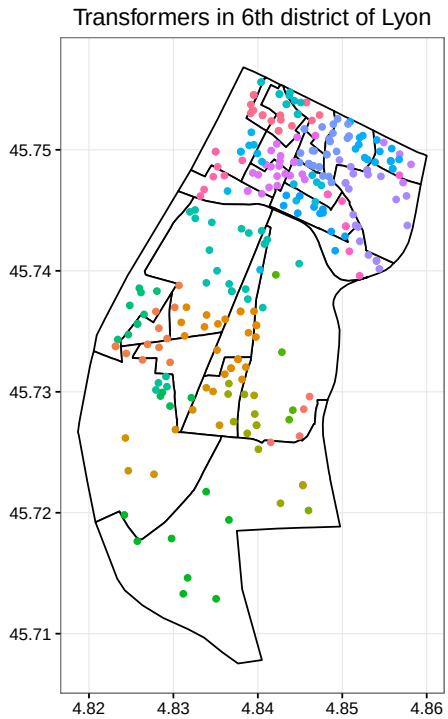


Figure 4.1: *Overlapping of IRIS and area supplied by MV feeders in the 7th municipal district of Lyon. IRIS are delimited by black borders. Points of different colors represent transformers connected to different MV feeders - each color corresponds to one MV feeder. In this example, source zones are areas served by transformers of the same color, and target zones, the IRIS, are the areas delimited by the black lines.*

spatially fine-grained enough to supply a direct estimate. To solve these two problems, we can use consumption data in source zones which are similar to the target zones in terms of socio-demographics. An illustration of this target/source zone distinction is given in Figure 4.1

Traditionally, spatial load forecasting is focused on the long term, with power system planning as primary objective [31]. In recent years, thanks to the increasing availability of data, various methods are proposed to model and to forecast electric load at various temporal and spatial scales [33]. For aggregated load forecasting from national down to substation level, state-of-art regression methods achieve satisfactory prediction error, with exogenous variables such as temperature and calendar variables [32, 65, 66]. However, there is generally a decrease in accuracy of these methods when the aggregation level becomes smaller [67, 68], even when historical data of the target zones are available. Consequently, more complex strategies are proposed to tackle small-area load forecasting. Reference [34] uses the hierarchy structure in electric grid to uncover patterns in the distribution network.

We propose to use socio-demographic information to transport consumption data from source zones to target zones with no or little historical consumption data. The idea is to form clusters of both source and target zones based on client information or socio-demographical information from publicly available datasets. We then estimate semi-parametric consumption models for each cluster, using consumption data only from source zones. These models are then transported to target zones to estimate consumption for past periods, or to forecast future periods. Clustering approaches [35, 69–71] are employed on smart meter time series to form consumer profiles to improve modeling accuracy. Contrary to these methods, we form clusters using socio-demographics or client information instead of consumption data. This allows us to produce estimates for target zones without historical data. The number of public datasets has been exploding for a few years. Many governments now provide detailed demographic, economic, and

sociological statistics collected at a local scale [72–74]. Our method takes advantage of these new data to produce new small-area load estimates.

The estimation procedure is explained in Section 4.2. The dataset, the validation procedure, and an application using real and simulated consumption is presented in Section 4.3.

4.2 Methodology

Our method has two main components: clustering of the geographical zones and consumption models. In the clustering part, we form clusters of both source and target zones using socio-demographic and client information. In the consumption model part, for each of the estimated clusters, we estimate regression models to explain the consumption behavior of the source zones included in this cluster.

4.2.1 Consumption models

The basic building block of our method is a regression model which explains electricity consumption using temperature and calendar variables. The general model is the following:

$$\mathbf{E}(Y_t) = f(\text{Temperature}_t) + g(\text{TOY}_t) + \alpha_{\text{SR}}\mathbf{I}_{\text{SR}_t} + \alpha_{\text{Xmas}}\mathbf{I}_{\text{Xmas}_t} + \sum_{h=1}^{9} \alpha_h \mathbf{I}_{\text{DayType}_t=h}. \tag{4.1}$$

In this regression model, the expected value of electric consumption at time t, $\mathbf{E}(Y_t)$, is the sum of several terms:

- $f(\text{Temperature}_t)$: a non-linear function of temperature at time t;
- $g(\text{TOY}_t)$: a non-linear function of time of the year (TOY), which is the proportion of the year between January 1 and the day in question (0 on January 1st and 1 on December 31st);
- $\alpha_{\text{SR}}\mathbf{I}_{\text{SR}_t}, \alpha_{\text{Xmas}}\mathbf{I}_{\text{Xmas}_t}, \alpha_h\mathbf{I}_{\text{DayType}_t=h}$: three additional dummy variables for periods with special electricity rates (SR), periods included in Christmas holidays (Xmas), and a 9-category day type (seven days of the week, plus two types of public holiday). The boldface $\mathbf{I}$ is a variable that is equal to 1 when the index is true, and 0 otherwise.

To fit the consumption model on data, we need to specify how the non-linear functions $f(\text{Temperature}_t)$ and $g(\text{TOY}_t)$ are estimated. We consider two specifications:

- Parametric specification:
  - $f(\text{Temperature}_t) = a_1 h_1(\text{Temperature}_t) + a_2 h_2(\text{Temperature}_t)$ where h_1 and h_2 are two functions of temperature specific to heating and air conditioning, previously estimated on the French national data [75]. Parameters a_1 and a_2 are to be estimated.
  - $g(\text{TOY}_t) = \sum_{m=1}^{m=4}(b_m \cos(2\pi m \text{TOY}_t) + c_m \sin(2\pi m \text{TOY}_t))$ is the first four terms of a Fourier series to model the yearly pattern of the consumption. Parameters b_m and c_m, for $1 \leq m \leq 4$, are to be estimated.
- Generalized additive model (GAM, [76]) specification: we estimate $f(\text{Temperature}_t)$ and $g(\text{TOY}_t)$ as linear combinations of spline functions. We use thin plate regression splines to estimate f, the temperature function. For g, the time-of-year function, we use cyclic cubic regression splines, which have identical values for up to the second derivative at both ends of a year. We estimate the time-of-year functions for weekdays and weekends separately.

In our tests, hourly consumption data is used. To account for intra-day variations, we model the 24 hours separately, resulting in 24 consumption models.

Computationally, the parametric specification is fitted using classical least square square method (function `lm` in the statistical programming language `R`). The estimation of a GAM model is handled in the `mgcv` package in `R` [76].

4.2.2 Cluster layer

We use the standard k-means algorithm in data mining to form clusters ([77, Chap 14]) of geographical zones. Given the fixed number of clusters, and characteristic variables for individuals, this algorithm alternatively iterate between:

- calculating the centroids of the variables of given clusters;
- placing individuals into clusters with the closest centroid.

Starting from a random partition, k-means results in a partition of the individuals into clusters of similar characteristics.

We form clusters of geographical zones. Therefore, the "individuals" are geographical zones. Because source zones and target zones have fundamentally different status in our methodology, we use a two-step procedure: the k-means algorithm is run on the source zones to form clusters; the target zones are then distributed into the cluster with the closest centroid. This assures that in each cluster, there are always some source zones. We consider the following two types of characteristics to form clusters:

- socio-demographic variables (IRIS): the percentages of population in each age group, the number of office buildings, shops, and factories in the zone, obtained from public census data;
- client information (CP): the percentages of residential, business, corporate, or industrial clients in contractual power (the subscribed maximal power), obtained from grid operators' knowledge of their clients.

The two groups of variables result in two clustering, called IRIS and CP clustering respectively.

For each cluster, we estimate one common consumption model using data of all included source zones. This model is then transported to the target zones of the same cluster: we use the regression model to estimate their load, by plugging in temperature and calendar information of the periods of interest.

Since consumption models are sensitive to the relative scaling between source zones of one cluster, the estimation procedure must be fed with normalized load time series. When calendar information and temperature is plugged into the model for prediction and estimation, the level of the consumption has to be supplied separately. In the application, we either predict the level by a linear regression model with CP variables as explanatory variables, or simply use the observed level of target zones.

4.3 Application on real and synthetic load data

In the application, we consider a region around Lyon in France. The source zones are geographical zones served by medium-voltage (MV) feeders. Two kinds of target zones are considered:

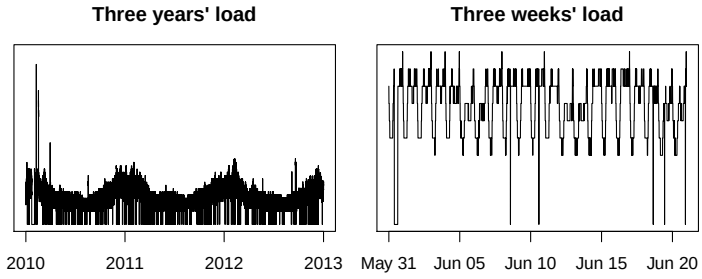


Figure 4.2: *Electric load of an MV feeder during three years (left) and three weeks (right).*

Table 4.1: *Rules of association of client type and distribution transformers. The + (or -) sign for either variable signifies that the value of the variable is larger (or smaller) than the median of all transformers.*

	Apartments/CP -	Apartments/CP +
Warehouses/CP -	rural homes	apartment block
Warehouses/CP +	industrial client	professional

MV feeder distribution zones with insufficient historical data and IRIS (*Ilots Regroupés pour l'Information Statistique*), an administration zone of approximately 2,000 habitants, used by the French National Institute of Statistics for releasing official socio-economic data. Note that feeder distribution zones and IRIS overlap. In this section, we present the datasets, the validation procedure, and finally the test results of our estimation procedure.

4.3.1 Datasets used

Real electricity consumption We use the hourly consumption during three years (2010-2012) of 473 MV feeders located in the region of interest. Fig. 4.2 shows an example of the time series. For confidentiality reasons, the y-axis is hidden in these figures. The hourly average load of all feeders included is around 1,500 kW over the three years. There is a strong presence of annual, weekly, and daily cycles on these time series, as we can see in the example.

Simulated electricity consumption To apply our method to IRIS consumption estimation, we generate consumption simulations for the 1,094 IRIS, and the feeders in the region of interest. The simulations are produced in the following manner. We have the geographical positions of the low-voltage (LV) distribution transformers connected to the feeders. These transformers are therefore positioned inside the boundaries of each IRIS. For each transformer, we choose a client type according to two ratios: the ratio between the number of apartments and contractual power (CP), and the ratio between the number of warehouses and CP. Crossing these two ratios, the transformer is assigned one of the 4 client types (Table 4.1). In addition, we assign a daily peak/off-peak period pattern randomly to each IRIS, from 4 realistic patterns. For each transformer, hourly consumption time series is simulated with regulated French profiles of typical consumers [64]. These profiles are a set of coefficients specifying, for each typical consumer group, the relative hourly consumption throughout a year, and adjustments linked to the temperature. We plug real local temperature and calendar variables for the 2010-2012 period into the profiles to simulate hourly consumption shape of each LV transformer, and multiply it by the total contractual power of the transformer. The IRIS aggregation is calculated by summing over transformers insider its boundary, while the feeder aggregation is obtained by summing over transformers connected to it.

Socio-demographic characteristics A number of variables are available at IRIS level from the French census [74] and from the SIRENE registry[1]. We use the age structure of the population and the number of shops, offices, factories, apartments and warehouses in each IRIS. We distribute the value of each variable for an IRIS in equal parts to the transformers located in it, and then re-aggregate to MV feeders.

Contractual power We use the contractual power (CP) at each transformer by client class as a proxy to client information. Four types of clients are considered: residential, small business, corporate clients supplied on low voltage, and industrial clients on medium voltage. For each transformer, the sum of CP for each of these four categories is known. The CP at an MV feeder (or an IRIS) is obtained by aggregating that of connected transformers (or within its boundaries).

Temperature Hourly historical temperature is obtained from www.climate.gov for weather stations in the region of interest. Each MV feeder and IRIS is associated to the weather station closest to its served zone.

4.3.2 Validation procedure

We test the estimation procedure by its prediction performance on target zones. The prediction is done in a "batch learning" fashion: the models are estimated using the first 30 months of data at once (training periods) on source zones for each of the clusters. Once the models are estimated, we produce forecasts for the last 3 months (test periods) on target zones, by using the temperature and calendar variables corresponding to test periods. For each target zone the mean absolute percentage error is calculated, according to the definition $MAPE_j = \frac{1}{n}\sum_{t=1}^{n}\frac{|Y_{j,t}-\hat{Y}_{j,t}|}{Y_{j,t}}$, where $Y_{j,t}$ is the consumption of Zone j at Period t, and $\hat{Y}_{j,t}$ the prediction of the procedure. Since the forecasting horizon varies from one hour to three months ahead, the $MAPE$ measures the average error of prediction of the procedure across forecasting horizons. As a comparison, we also predict consumption on test periods for the source zones using the same estimated models. The difference of average $MAPE$ between target and source zones is a measure of the transportability of the models, hence the performance of the procedure.

4.3.3 Application on real MV feeder consumption

On the real consumption dataset, which consists of 473 feeders, we randomly select 100 feeders as source zones, and use the rest as target zones. We conduct 5 independent random selections, and the errors reported are averaged over these 5 runs. On source zones, we run the k-means algorithm on two sets of variables:

- the age structure and number of offices, shops and factors;
- percentage of contractual power in client types.

This results in two sets of clusters (IRIS clusters and CP clusters). For each cluster, we use the 30 months of source zone consumption data to estimate one consumption model. Predictions are made for the last three months on source zones and target zones alike.

[1]SIRENE, *Systeme Informatique pour le Repertoire des Entreprises et de leurs Etablissements*, the INSEE database of all companies, public and private organization.

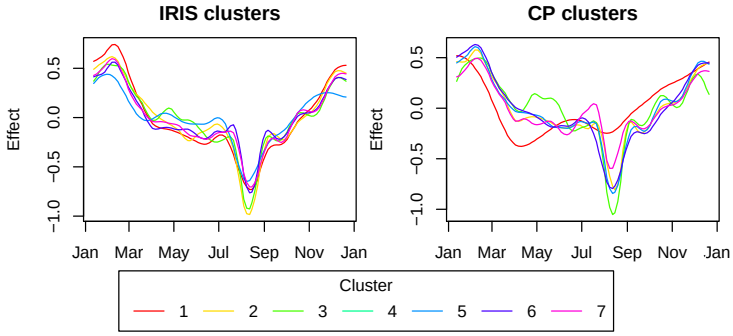


Figure 4.3: *Cluster model: GAM estimation of annual cycle for the consumption at 10 a.m. on weekdays.*

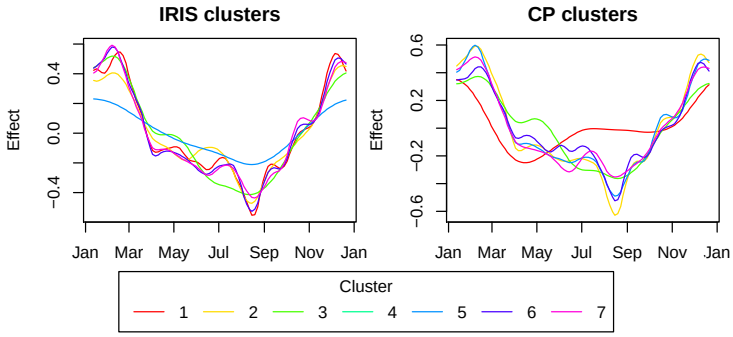


Figure 4.4: *Cluster model: GAM estimation of annual cycle for the consumption at 10 a.m. on weekend.*

Figs. 4.3 and 4.4 show the GAM estimation of the annual cycle for consumption at 10 *a.m.* for each cluster of both sets of clusters, respectively for weekdays and weekends[2,3]. Whether on weekdays or weekends, and across the clusters, the annual cycle is that consumption is higher in winter than in summer, with a large decrease in August due to vacations, and a smaller decrease in May due to the bank holidays. In both sets of clusters, several clusters have a noticeably smaller decrease in August (Clusters 3 and 5 in IRIS data clusters, Clusters 1 and 7 in CP clusters). This is linked to the fact that these clusters are generally located in rural area (*i.e.* they have a lower than average percentage of apartments), where less people go on vacations in summer. These differences in clusters are also visible in functions estimated in the parametric specification (not shown here).

Fig. 4.5 shows the effect of special rate (SR) days estimated by the GAM specification. The higher price starts at 6 *a.m.* on the SR day, and ends at midnight. This is why these effects start at 6 *a.m.*. Often clients reduce their consumption during high rate hours and increase consumption just after midnight, when the rates come back to normal, as can be seen in the figures. Clusters 1 and 5 in the IRIS clusters and Cluster 1 in CP clusters have much less drastic SR effect. This shows that the clusters we obtained are able to capture some differences in electricity consumption of the clients, although no consumption data is used in the clustering step. The parametric specification has qualitatively similar results.

Prediction error rates for the different model specifications are shown in Table 4.2. In forecasts with observed levels (first two rows in each half of the table), the difference of error rate between source and target zones is small. This suggests that the consumption models

[2]The scales are relative on Figs.4.3 and 4.4, since the estimation is applied to normalized time series.
[3]Results for other hours are qualitatively similar.

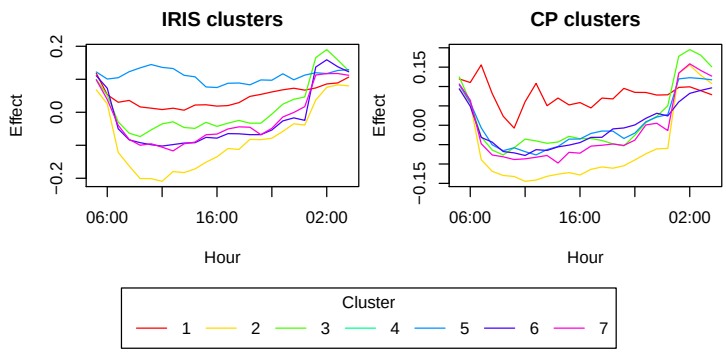


Figure 4.5: *Cluster model: GAM estimation of the SR effect throughout the day.*

Table 4.2: *Mean prediction error rate on the real dataset.*

Specification	Model type	IRIS clusters	CP clusters
Parametric	Observed level (source)	18.62	17.90
Parametric	Observed level (target)	19.13	18.54
Parametric	Estimated level (target)	38.20	37.76
GAM	Observed level (source)	18.90	17.59
GAM	Observed level (target)	19.38	18.28
GAM	Estimated level (target)	38.41	37.53

obtained by the estimation procedure are transportable to target zones. In both cases, the error rate is less than 20%, comparable to the state of art at similar aggregation level with full observations [67].

As a reference, a preliminary study is done to evaluate of predictions of individual feeder GAM models with the similar specification. The error rates Table 4.2 are higher than these individual models (11.7% for the 473 feeders on average, compared to the around 18% for source zones). This is not surprising, since the cluster models are much less specialized than individual feeder models. This decrease in model accuracy is a small price to pay for the increased model transportability, which enables us to estimate hourly consumption for target zones without using its historical data.

The CP clustering model has a lower error rate, indicating that direct client information has greater importance than indirect socio-demographic variables. Parametric models have similar performances in prediction as GAMs.

We also tried to estimate consumption levels by linear regression using CP variables. The error rates are in the third row of each half of Table 4.2. These predictions have an average error rate between 35% and 40%. This is because the estimation of consumption level by CP is not accurate. If yearly aggregated consumption of target zones is available, this error rate can be reduced.

Table 4.3: *Mean prediction error rate on simulated consumption.*

	IRIS clusters	CP clusters
MV feeders (source)	13.20	11.22
MV feeders (target)	13.87	11.45
IRIS estimation (target)	16.93	14.09

4.3.4 Application on synthetic consumption of feeders and IRIS

On the synthetic consumption dataset, we keep the same five sets of randomly selected 100 feeders as source zones. As target zones, we use the rest of the feeders, and the 1,094 IRIS. As in the real consumption application, models are estimated on source zone load of the first 30 months. Predictions are made for the last 3 months on both source and target zones. We only report the results on GAM specification with observed levels, since the relative difference is similar to the real consumption case. To produce clusters, we use either the apartment, warehouse variables used in simulation, or the percentage of CP in client type. The two sets of clusters are similarly called IRIS clusters and CP clusters.

Prediction error rates are shown in Table 4.3. The difference between source and target zones of feeder service area is small, as in the real consumption case. This confirms that consumptions models obtained by the process are indeed transportable to the target zones. The prediction error is generally lower in this case, as consumption is smoother in simulations than in real data, and the clusters are more homogeneous. We observe a similar slight decrease in prediction error when the percentage of CP in client type is used. This is also reassuring, for it suggests a similarity between real data and simulations.

The third row in Table 4.3 shows the prediction error on IRIS consumption. We observe an average error rate of 16.93% and 14.09% for IRIS clusters and CP clusters. This is higher, but of the same order as short-term forecasting at this aggregation level with full data [67]. Such precision is reasonably good, given that the estimation requires only the average consumption level for each IRIS. As a benchmark, we also used an interpolation proportional to the population from the 100 source zones to produce an estimation for aggregated consumption of the IRIS. This method does very poorly on aggregated IRIS consumption, with 49.57% of mean error. Our method is thus significantly better than the interpolation approach.

4.4 Conclusion

In this chapter, we propose a method to estimate hourly electricity consumption for target zones with insufficient or no historical consumption data using measurements from source zones and public socio-demographic information. The procedure provides state-of-art prediction performance on real and simulated datasets for target zones with very little data, and significantly outperforms the benchmark interpolation method in simulations.

Part II

Nonnegative matrix factorization with general linear measurements and its applications in time series recovery and prediction

Chapter

5 Introduction to nonnegative matrix factorization

Contents

5.1 Introduction

As is suggested by Figure 2.1 on page 14, the object of interest in this thesis can easily be represented in a matrix form. Based on this data representation, we will use nonnegative matrix factorization (NMF, [5]) methods to tackle the estimation problems introduced in Chapter 2. Matrix factorization methods leverage the fact that big matrices observed in real datasets are often of rank much smaller than their dimension, to facilitate the estimation, compression, and processing of such datasets.

In this chapter, after a short introduction to NMF (Section 5.2.1), we will reformulate the estimation problems discussed in Chapter 2 in terms of matrix factorization (Section 5.2.2). In Section 5.3, we will examine some recent progress in the non-convex optimization literature, particularly, how non-convex problems such as matrix factorization can have global convergence with an iterative algorithm, and why it is difficult to achieve the same in NMF.

In Chapters 6 and 7, we will study two specific cases of the general model to better adapt it to our estimation problems. In Chapter 8, we will compare the NMF framework with kriging and socio-demographic clustering introduced previously in Part I (Chapters 3 and 4). We will see that these two methods can be reformulated as special cases of the matrix factorization framework. In Chapter 9, we will review some particularities in the implementation of the

algorithms developed in this part, and present `meterModels`, the `R` package developed in this thesis.

5.2 Multivariate time series estimation and prediction as nonnegative matrix recovery

5.2.1 Nonnegative matrix factorization

Low-rank approximation and factorization methods have been extensively studied in recent years. This class of methods uses the low-rank hypothesis: that a matrix $\mathbf{V} \in \mathbb{R}^{n_1 \times n_2}$ is of rank k much smaller than its dimensions, with $k \ll n_1, n_2$. This means that $\mathbf{V}$ can be rewritten in a factorized form:

$$\mathbf{V} = \mathbf{F}_r \mathbf{F}_c^T, \tag{5.1}$$

where $\mathbf{F}_r \in \mathbb{R}^{n_1 \times k}$ and $\mathbf{F}_c \in \mathbb{R}^{n_2 \times k}$ are called *factors*. In particular, by doing a singular value decomposition (SVD) of $\mathbf{V}$, one can easily find such a low-rank factorization.

It is in general interesting to use $\mathbf{F}_r$ and $\mathbf{F}_c$ instead of $\mathbf{V}$. Since they have lower dimensions, they can be useful for a wide range of applications such as estimation or compression. Even if $\mathbf{V}$ is not exactly of a small rank k, it can still be interesting to find a rank-k approximation for it, by minimizing

$$\min_{\mathbf{F}_r \in \mathbb{R}^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}^{n_2 \times k}} \|\mathbf{V} - \mathbf{F}_r \mathbf{F}_c^T\|_F^2, \tag{5.2}$$

for example, where for any matrix $\mathbf{X}$, $\|\mathbf{X}\|_F$ is its Frobenius norm (Euclidean norm of its entries). In this case, the minimization problem (5.2) is called low-rank *approximation*.

In this thesis, we are especially interested in nonnegative matrix factorization, where the matrix to be factorized and the factors only have nonnegative entries. Note that a matrix of rank k does not necessarily have a nonnegative matrix factorization with k columns in the factor matrices. The smallest integer k for which there exist $\mathbf{F}_r \in \mathbb{R}_+^{n_1 \times k}$ and $\mathbf{F}_c \in \mathbb{R}_+^{n_2 \times k}$ so that $\mathbf{V} = \mathbf{F}_r \mathbf{F}_c^T$ is called the *nonnegative rank* of $\mathbf{V}$. The nonnegative rank is always equal or larger than the rank.

For a given integer k, the nonnegative factorization/approximation problem then becomes

$$\begin{aligned} \min_{\mathbf{F}_r \in \mathbb{R}^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}^{n_2 \times k}} \quad & \|\mathbf{V} - \mathbf{F}_r \mathbf{F}_c^T\|_F^2 \\ \text{s.t.} \quad & \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}, \end{aligned} \tag{5.3}$$

where $\mathbf{X} \geq \mathbf{0}$ for a matrix $\mathbf{X}$ is an entry-wise inequality.

Nonnegative matrix factorization is generally more interpretable, when the underlying quantites are indeed nonnegative. In our application case, the entries of the matrix to be recovered are the quantity of electricity consumption, which is positive. In the convention adopted in this thesis, the rows are periods, and columns are individuals (Figure 2.1). The factor matrices have a naturel interpretation in this convention: the left factor $\mathbf{F}_r$ has columns which are interpreted as typical consumption profiles, and the entries of the right factor matrix $\mathbf{F}_c$ are weights of the individuals in each of the profiles (Figure 5.1).

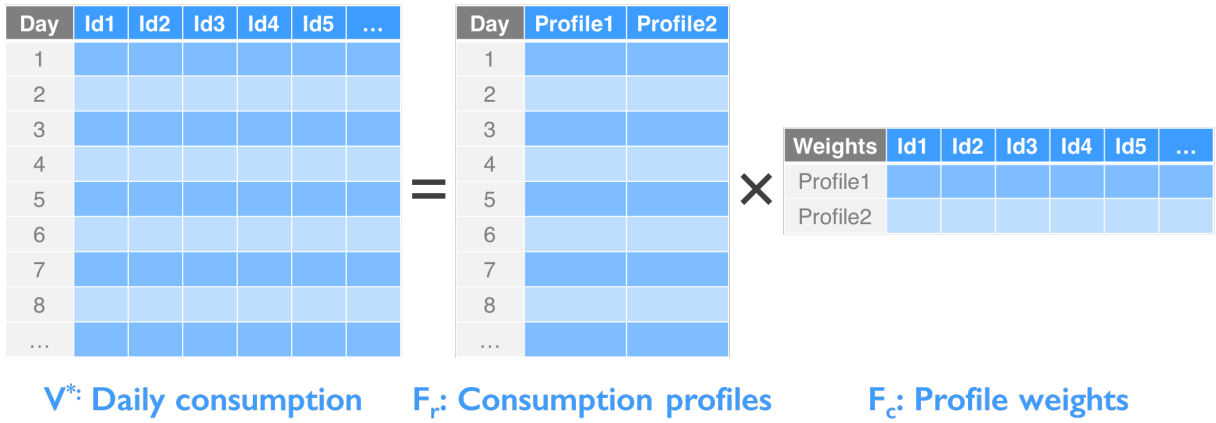


Figure 5.1: *An illustration of nonnegative matrix factorizaiton for a matrix of daily electricity consumption series.*

5.2.2 General model definition

To solve the estimation problems considered in Figure 2.1, we are interested in reconstructing a nonnegative matrix $\mathbf{V}^* \in \mathbb{R}_+^{n_1 \times n_2}$, from N linear measurements,

$$\boldsymbol{\alpha} = \mathcal{A}(\mathbf{V}^*) \in \mathbb{R}^N, \tag{5.4}$$

where $\mathcal{A} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^N$ is a linear operator. Formally, $\mathcal{A}$ can be represented by $\mathbf{A}_1$, ..., $\mathbf{A}_N$, N design matrices of dimension $n_1 \times n_2$, and each linear measurement can be represented by

$$\alpha_i = \mathrm{Tr}(\mathbf{V}^* \mathbf{A}_i^T) = \langle \mathbf{V}^*, \mathbf{A}_i \rangle. \tag{5.5}$$

The design matrices $\mathbf{A}_1$, ..., $\mathbf{A}_N$ are called *masks*. The matrices on the right part of Figure 5.2 show a subsection of the masks correponding to the indices on individual electricity meters shown on the left of Figure 5.2.

We suppose that the matrix of interest, $\mathbf{V}^*$, is of nonnegative rank k: we can find two nonnegative matrices $\mathbf{F}_r \in \mathbb{R}_+^{n_1 \times k}$ and $\mathbf{F}_c \in \mathbb{R}_+^{n_2 \times k}$ so that

$$\mathbf{V}^* = \mathbf{F}_r \mathbf{F}_c^T,$$

with $k \ll n_1, n_2$. Note that this implies that $\mathbf{V}^*$ is of rank at most k, and therefore is of low rank.

In the nonnegative matrix recovery problem, we try to recover the true matrix $\mathbf{V}^*$ as well as the factor matrices $\mathbf{F}_r$ and $\mathbf{F}_c$, given $\mathcal{A}$, the measurement operator, and $\boldsymbol{\alpha}$, the measurements.

To obtain an estimation, we minimize the quadratic error of the matrix factorization. In the two following chapters, we will propose algorithms for optimization problems of the following form:

$$\begin{aligned} \min_{\mathbf{F}_r \in \mathbb{R}^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}^{n_2 \times k}, \mathbf{V} \in \mathbb{R}^{n_1 \times n_2}} \quad & \|\mathbf{V} - \mathbf{F}_r \mathbf{F}_c^T\|_F^2 \\ \text{s.t.} \quad & \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}, \quad \mathbf{V} \geq \mathbf{0}, \\ & \mathcal{A}(\mathbf{V}) = \boldsymbol{\alpha}. \end{aligned} \tag{5.6}$$

Note that in this thesis, $\boldsymbol{\alpha}$, the data used in matrix recovery are generated by a measurement operator, hence are "coherent". Since there is actually a ground truth matrix $\mathbf{V}^*$ that verifies $\mathcal{A}(\mathbf{V}^*) = \boldsymbol{\alpha}$, the feasible set is obviously non-empty.

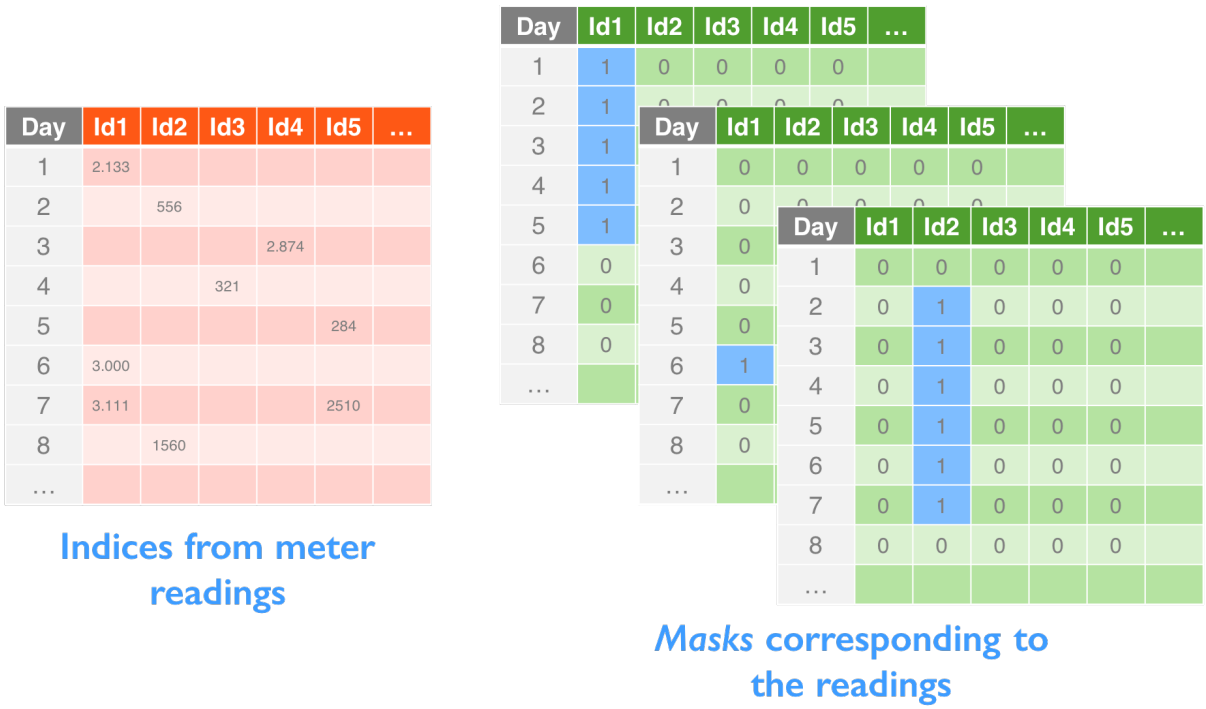


Figure 5.2: *An illustration of masks which correpond to the meter readings on individual meters.*

In order to recover a low-rank matrix, the following alternative optimization problem can also be considered:

$$\begin{aligned} \min_{\mathbf{F}_r \in \mathbb{R}^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}^{n_2 \times k}} \quad & \|\boldsymbol{\alpha} - \mathcal{A}(\mathbf{F}_r \mathbf{F}_c^T)\|_2^2 \\ \text{s.t.} \quad & \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}. \end{aligned} \tag{5.7}$$

Instead of minimizing the low-rank approximation error for a matrix that verifies the data constraint in (5.6), (5.7) minimizes the sampling error of an exactly low-rank matrix. Both have been studied in the literature. For example, objective functions similar to (5.6) have been considered in [78], and ones similar to (5.7) in [79]. We will compare the two in Chapter 7 to argue that (5.6) is a more efficient strategy than (5.7).

5.2.3 Applications of matrix recovery

By specifying different masks, matrix recovery has a number of interesting applications, old and new.

- **Complete observation**: $N = n_1 n_2$, $\mathbf{A}_{i_1,i_2} = \mathbf{e}_{i_1}\mathbf{e}_{i_2}^T$, where $\mathbf{e}_i$ is the i-th canonical vector. This means every entry of $\mathbf{V}^*$ is observed. Obviously, in this case, there is no need to estimate $\mathbf{V}^*$. The main reason to do this is for dimension reduction or denoising. With nonnegative factors, this is used in image segmentation [5], topic extraction [17], and clustering [80], for example.
- **Matrix completion - recommender systems**: $N < n_1 n_2$, the set of design matrix is a subset of complete observation masks. The objective is then to infer the unobserved entries. This is the standard task for recommender systems [81], whether with nonnegative factors or real factors.
- **Matrix sensing**: the design matrices $\mathbf{A}_i$ are *i.i.d.* (independent and identically distributed) random variables from a certain random matrix probability distribution. Typi-

cally, the probability distribution needs to verify certain conditions, so that with a large probability, $\mathcal{A}$ verifies the Restricted Isometry Property [78]. This class of masks are very well studied theoretically. For a more recent review, see [82]. An application of this class of masks with nonnegative factors in imaging can be found in [83].

- **Rank-one projections**: the design matrices are random rank-one matrices, that is $\mathbf{A}_i = \boldsymbol{\alpha}_i \boldsymbol{\beta}_i^T$, where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are respectively random vectors of dimension n_1 and n_2. The main advantage to this setting is that much less memory is needed to store the masks, since we can store $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ (dimension-$(n_1 + n_2)$) instead of $\mathbf{A}_i$ (dimension-$(n_1 \times n_2)$). In [84, 85], theoretical properties are proved for the case where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are vectors with independent Gaussian entries and/or drawn uniformly from the vectors of the canonical basis.

- **Temporal aggregate measurements**: in this case, the matrix is composed of n_1 time series concerning n_2 individuals, and each measure is a disjoint temporal aggregate of the time series of an individual. The design matrices are defined as $\mathbf{A}_i = \sum_{t=t_0(i)+1}^{t_0(i)+h(i)} \mathbf{e}_t \mathbf{e}_{s_i}^T$, where s_i is the individual concerned by the i-th measure, $t_0(i)+1$ the first period covered by the measure, and $h(i)$ the number of periods covered by the measure. This class of masks is the main topic of Chapter 6, since it models the meter readings of resident clients with traditional electricity meters. A similar type of masks has also been studied in network traffic matrix estimation [79].

5.3 Convergence of nonnegative matrix factorization algorithms

5.3.1 Global convergence of alternating minimization algorithms in non-convex problems

The general matrix factorization problem (5.2) has a simple form, but finding its solutions is not simple. Methods based on nuclear norm minimization (by minimizing the trace of $\mathbf{V}$) for matrix recovery and matrix completion problems often have theoretical guarantees [78, 86]. However, first-order algorithms based on the factorized form (alternating minimization, gradient descent, etc.) are much more efficient, and more preferred by practitioners, because they work on a smaller parameter space. This class of methods try to solve a non-convex problem, and therefore rarely had global convergence guarantees.

It is only recently that theoretical guarantees for general matrix factorization algorithms began appearing [87–93]. This progress is connected to the literature on global convergence results for other non-convex problems [16, 94–97]. Motivated by the superior empirical performances, this trend of analysis shows that although a specific class of problems is not convex, nice qualities about the landscape can still guarantee the convergence of descent algorithms.

Of particular interest is the reference [90], in which the authors showed that the general matrix factorization/recovery problem has no spurious local minima. Combined with results on escaping saddle points, stochastic gradient descent algorithms therefore can converge to the global optimum. Contrary to a number of previous works, this result does not rely on a particular initialization of the algorithm.

In general, NMF is an NP-hard problem [11]. Therefore, there cannot be similar results to the general NMF problem. However, in [91] and [98], guarantees of the NMF have been given for alternating direction algorithms, for cases where one of the factors is initialized at a point close to the ground truth.

In the rest of this section, we will go through the steps by [90] to show why this analysis cannot be directly applied to NMF.

5.3.2 What would global convergence mean?

In order to simplify the theoretical analysis, we focus temporarily on the factorization case (5.3), where the data matrix $\mathbf{V}^*$ is completely observed. We note by $\ell(\mathbf{F}_r, \mathbf{F}_c) = \|\mathbf{V}^* - \mathbf{F}_r(\mathbf{F}_c)^T\|_F^2$, the loss function of (5.3).

A pair of matrices $(\mathbf{F}_r^*, \mathbf{F}_c^*)$ is called a *local minimizer* of (5.3), if there exists $\epsilon > 0$, so that $\forall(\mathbf{Z}_1, \mathbf{Z}_2) \in \mathbb{R}_+^{n_1 \times k} \times \mathbb{R}_+^{n_2 \times k}$ with $\|\mathbf{Z}_1 - \mathbf{F}_r^*\|_F^2 + \|\mathbf{Z}_2 - \mathbf{F}_c^*\|_F^2 \leq \epsilon$, we have

$$\|\mathbf{V}^* - \mathbf{F}_r^*(\mathbf{F}_c^*)^T\|_F^2 \leq \|\mathbf{V}^* - \mathbf{Z}_1(\mathbf{Z}_2)^T\|_F^2.$$

Any local minimizer satisfies the Karush–Kuhn–Tucker (KKT) necessary conditions [99]. Noting by $\partial_{\mathbf{F}_r}\ell(\mathbf{F}_r, \mathbf{F}_c)$ and $\partial_{\mathbf{F}_c}\ell(\mathbf{F}_r, \mathbf{F}_c)$ the partial derivatives of the loss function, and by $\nabla^2\ell(\mathbf{F}_r, \mathbf{F}_c)$ the second order derivative, the KKT conditions are,

$$\begin{aligned} &\partial_{\mathbf{F}_r}\ell(\mathbf{F}_r^*, \mathbf{F}_c^*) \geq 0, \partial_{\mathbf{F}_r}\ell(\mathbf{F}_r^*, \mathbf{F}_c^*) \circ \mathbf{F}_r^* = \mathbf{0}, \\ &\partial_{\mathbf{F}_c}\ell(\mathbf{F}_r^*, \mathbf{F}_c^*) \geq 0, \partial_{\mathbf{F}_c}\ell(\mathbf{F}_r^*, \mathbf{F}_c^*) \circ \mathbf{F}_c^* = \mathbf{0}, \end{aligned} \tag{5.8}$$

and

$$\begin{aligned} &\langle \begin{pmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \end{pmatrix}, \nabla^2\ell(\mathbf{F}_r^*, \mathbf{F}_c^*) \begin{pmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \end{pmatrix} \rangle \geq 0, \forall \begin{pmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \end{pmatrix} \in \mathbb{R}^{(M+N)\times K}, \text{that satisfies} \\ &\mathbf{Z}_1 \circ \mathbb{1}_{\mathbf{F}_r=\mathbf{0}} = \mathbf{0}, \mathbf{Z}_2 \circ \mathbb{1}_{\mathbf{F}_c=\mathbf{0}} = \mathbf{0}, \end{aligned} \tag{5.9}$$

where $\circ$ denotes the element-wise matrix product, and $\mathbb{1}_{\mathbf{X}=\mathbf{0}}$ is a matrix of the same dimension of $\mathbf{X}$ for any matrix $\mathbf{X}$, with an entry 1 where $\mathbf{X}$ has an entry 0, and 0 otherwise. This means we only need to check for second-order increments in directions in which $\mathbf{F}_r$ and $\mathbf{F}_c$ are not on the boundary of the first orthant.

If $\ell(\mathbf{F}_r^*, \mathbf{F}_c^*) = 0$ is true for all matrix pairs $(\mathbf{F}_r^*, \mathbf{F}_c^*)$ that verify the KKT conditions, then all local minima of the optimization problem above is a global optimum. If, additionally, the matrix has a unique NMF up to scaling and permutation, then all local minima of the optimization program is the NMF of the matrix $\mathbf{V}^*$.

Therefore, what we would like to know is if there are sufficient conditions on $\mathbf{V}^*$ that an algorithm could check with relative ease, for the KKT conditions to imply zero residual ($\ell(\mathbf{F}_r^*, \mathbf{F}_c^*) = 0$).

5.3.3 Link with general matrix factorization

The question of global optimality of local NMF algorithms arises naturally from the work of [90], who proved that the general matrix factorization problem (namely minimization of the same loss function without non-negativity constraint on the factors) has no spurious local minima. Their work is in a more general context where the data are linear measures of $\mathbf{V}$ under a measurement operator which satisfies a restricted isometry property (RIP).

There are two-ways to view the link between general matrix factorization and NMF:

- NMF is the optimization of the same objective function as general matrix factorization, in a constrained domain (local minima could therefore exist on the boundary of the feasible domain of NMF).

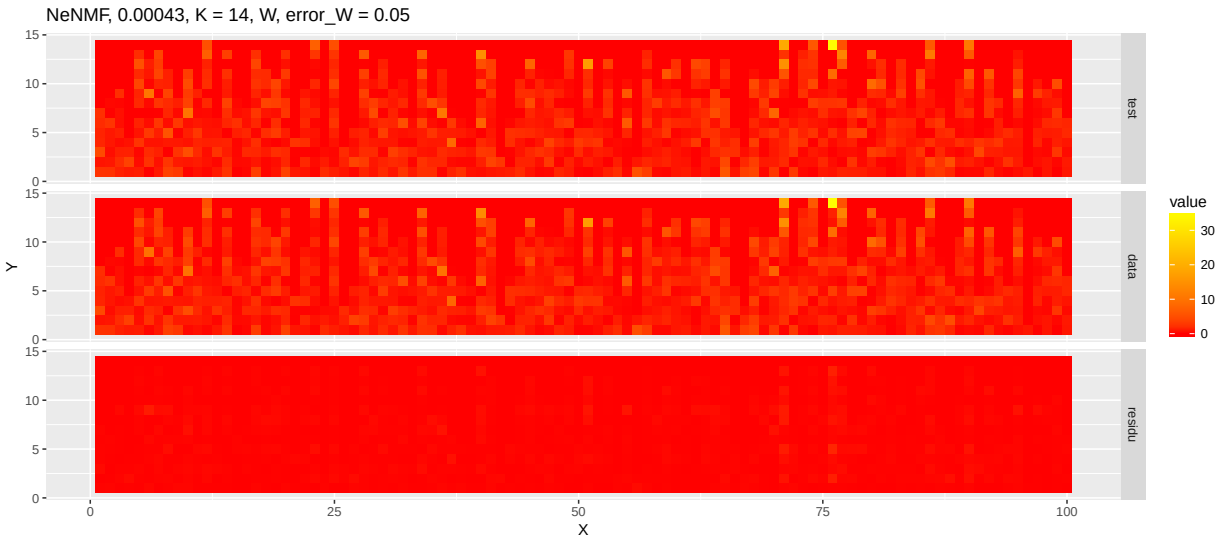


Figure 5.3: *Left rank-14 factor estimated by* `NeNMF`

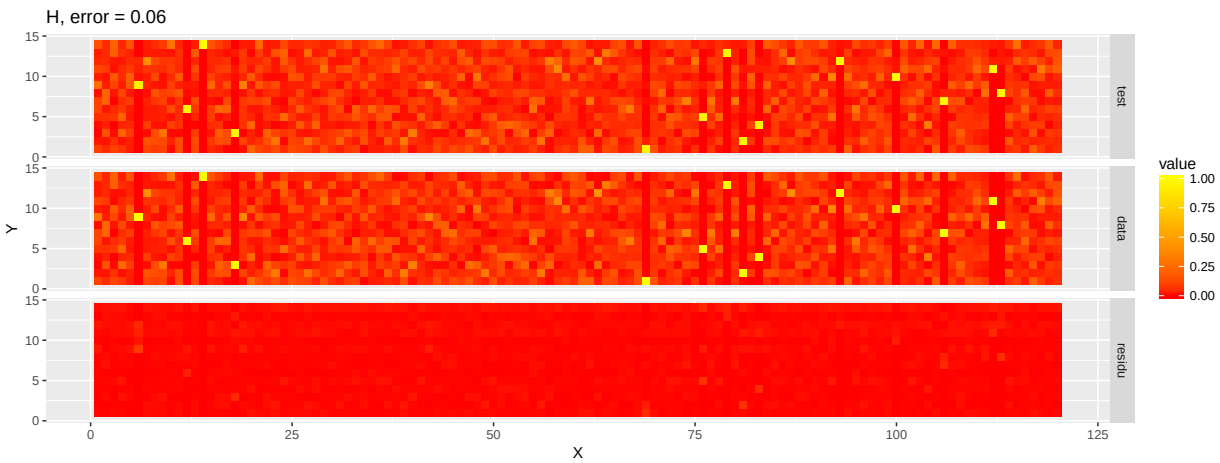


Figure 5.4: *Right rank-14 factor estimated by* `NeNMF`

- The general matrix factorization is only identified up to an invertible k-by-k matrix: for every global optimum $(\mathbf{F}_r^*, \mathbf{F}_c^*)$, $(\mathbf{F}_r^*\mathbf{R}, \mathbf{F}_c^*(\mathbf{R}^{-1})^T)$ is also a global optimum. NMF is about finding out a specific invertible matrix $\mathbf{R}$ so that both $\mathbf{F}_r\mathbf{R}$ and $\mathbf{F}_c(\mathbf{R}^{-1})^T$ are non-negative. Intuitively, this could be quite difficult, since there is no information on $\mathbf{R}$ in the data.

5.3.4 Empirical convergence

Although there is currently no prove of global convergence for local NMF algorithms, empirically there seems to be global convergence on certain class of NMF problems, especially if the true factors have many zero entries.

In Tables 5.1 and 5.2, we report the NMF objective function value and the relative error on the obtained factors $\mathbf{F}_c$ and $\mathbf{F}_r$, on matrices with unique NMFs with rank ranging from 4 to 14. The algorithms used (`NeNMF` and `HALS`) are classical NMF algorithms [20, 27].

The original and estimated factors, and the error on each element, are plotted in Figures 5.3, 5.4, 5.5, 5.6. The relative error on factors is of several percentage points, and we can see that estimated and original matrices are quite close.

Table 5.1: *Relative error (RRMSE) in Frobenius norm of the factors produced by the* `NeNMF` *algorithm.*

Rank	Objective function	Error on $\mathbf{F}\_r$	Error on $\mathbf{F}\_c$
4	5e-04	0.0111	0.0145
5	6e-04	0.0304	0.0420
6	6e-04	0.0243	0.0369
7	6e-04	0.0345	0.0406
8	5e-04	0.0215	0.0325
9	5e-04	0.0375	0.0509
10	5e-04	0.0308	0.0410
11	4e-04	0.0316	0.0444
12	4e-04	0.0216	0.0270
13	5e-04	0.0332	0.0508
14	4e-04	0.0487	0.0577

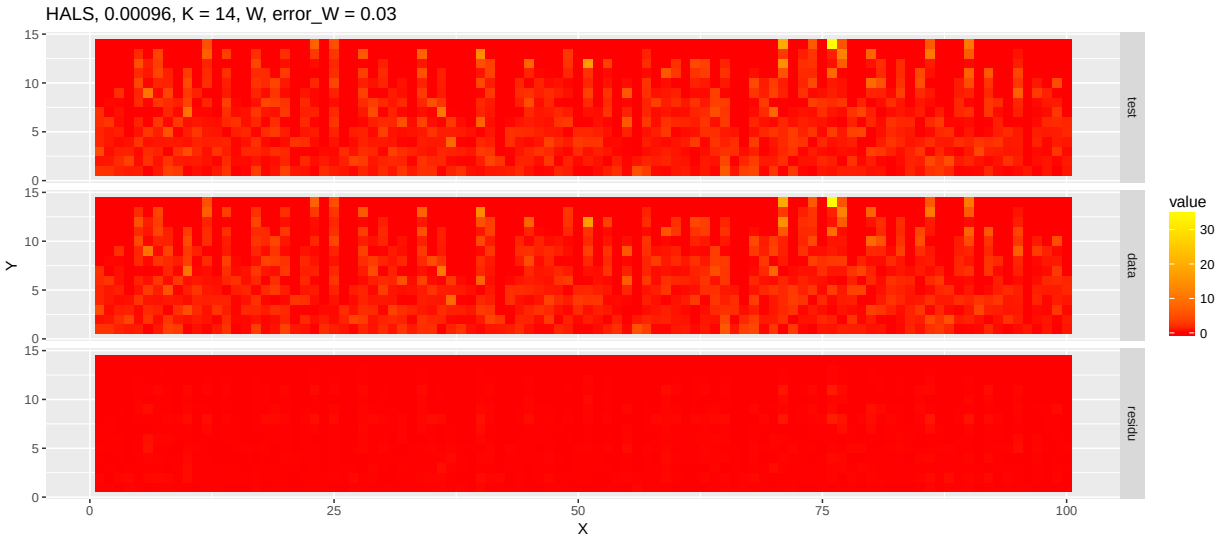


Figure 5.5: *Left rank-14 factor estimated by* `HALS`

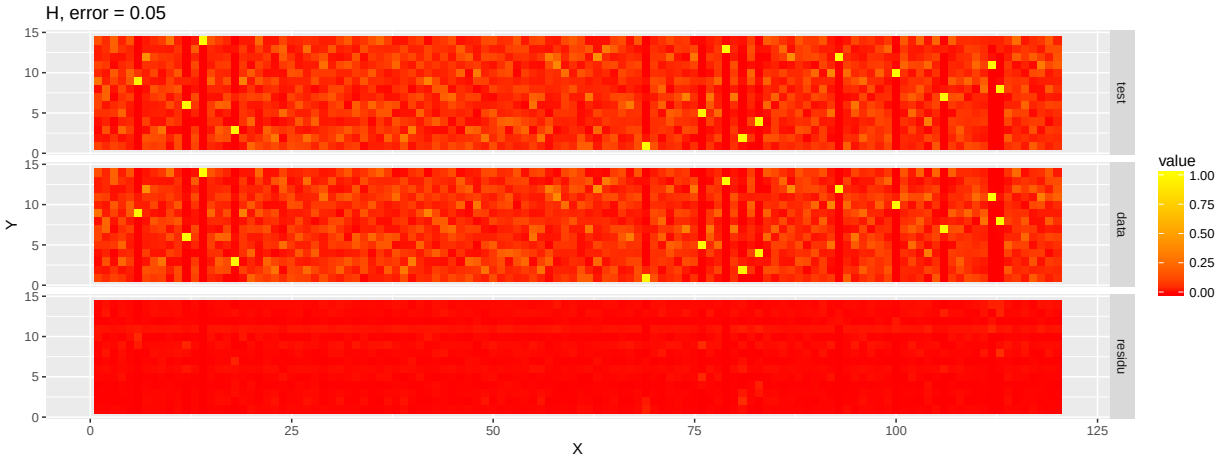


Figure 5.6: *Right rank-14 factor estimated by* `HALS`

Table 5.2: *Relative error (RRMSE) in Frobenius norm of the factors produced by the* `HALS` *algorithm.*

Rank	Objective function	Error on $\mathbf{F}\_r$	Error on $\mathbf{F}\_c$
4	1e-04	0.0038	0.0045
5	8e-04	0.0245	0.0317
6	8e-04	0.0169	0.0307
7	5e-04	0.0176	0.0235
8	7e-04	0.0231	0.0370
9	7e-04	0.0228	0.0320
10	5e-04	0.0196	0.0265
11	6e-04	0.0197	0.0291
12	6e-04	0.0304	0.0369
13	8e-04	0.0206	0.0295
14	1e-03	0.0350	0.0524

5.3.5 Local optimality conditions

By differentiating the loss function, we obtain that

$$\partial_{\mathbf{F}_r}\ell(\mathbf{F}_r,\mathbf{F}_c) = (\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)\mathbf{F}_c,$$
$$\partial_{\mathbf{F}_c}\ell(\mathbf{F}_r,\mathbf{F}_c) = (\mathbf{F}_c\mathbf{F}_r^T - (\mathbf{V}^*)^T)\mathbf{F}_r.$$

By differentiating a second time, we obtain that $\forall(\mathbf{Z}_1,\mathbf{Z}_2) \in \mathbb{R}^{n_1\times k}\times\mathbb{R}^{n_2\times k}$,

$$\begin{aligned}\nabla^2\ell(\mathbf{F}_r,\mathbf{F}_c)\begin{pmatrix}\mathbf{Z}_1\\ \mathbf{Z}_2\end{pmatrix} &= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\begin{pmatrix}\partial_{\mathbf{F}_r}\ell(\mathbf{F}_r+\epsilon\mathbf{Z}_1,\mathbf{F}_c+\epsilon\mathbf{Z}_2)-\partial_{\mathbf{F}_r}\ell(\mathbf{F}_r,\mathbf{F}_c)\\ \partial_{\mathbf{F}_r}\ell(\mathbf{F}_r+\epsilon\mathbf{Z}_1,\mathbf{F}_c+\epsilon\mathbf{Z}_2)-\partial_{\mathbf{F}_c}\ell(\mathbf{F}_r,\mathbf{F}_c)\end{pmatrix}\\ &= \begin{pmatrix}(\mathbf{F}_r\mathbf{F}_c^T-\mathbf{V}^*)\mathbf{Z}_2+(\mathbf{F}_r\mathbf{Z}_2^T+\mathbf{Z}_1\mathbf{F}_c^T)\mathbf{F}_c\\ (\mathbf{F}_c\mathbf{F}_r^T-(\mathbf{V}^*)^T)\mathbf{Z}_1+(\mathbf{F}_c\mathbf{Z}_1^T+\mathbf{Z}_2\mathbf{F}_r^T)\mathbf{F}_r\end{pmatrix}.\end{aligned}$$

This leads to

$$\begin{aligned}&\langle\begin{pmatrix}\mathbf{Z}_1\\ \mathbf{Z}_2\end{pmatrix},\nabla^2\ell(\mathbf{F}_r,\mathbf{F}_c)\begin{pmatrix}\mathbf{Z}_1\\ \mathbf{Z}_2\end{pmatrix}\rangle\\ =&\langle\mathbf{Z}_1,(\mathbf{F}_r\mathbf{F}_c^T-\mathbf{V}^*)\mathbf{Z}_2+(\mathbf{F}_r\mathbf{Z}_2^T+\mathbf{Z}_1\mathbf{F}_c^T)\mathbf{F}_c\rangle+\langle\mathbf{Z}_2,(\mathbf{F}_c\mathbf{F}_r^T-(\mathbf{V}^*)^T)\mathbf{Z}_1+(\mathbf{F}_c\mathbf{Z}_1^T+\mathbf{Z}_2\mathbf{F}_r^T)\mathbf{F}_r\rangle\\ =&2\langle\mathbf{Z}_1\mathbf{Z}_2^T,(\mathbf{F}_r\mathbf{F}_c^T-\mathbf{V}^*)\rangle+\langle\mathbf{Z}_1\mathbf{F}_c^T+\mathbf{F}_r\mathbf{Z}_2^T,\mathbf{F}_r\mathbf{Z}_2^T+\mathbf{Z}_1\mathbf{F}_c^T\rangle\\ =&2\langle\mathbf{Z}_1\mathbf{Z}_2^T,(\mathbf{F}_r\mathbf{F}_c^T-\mathbf{V}^*)\rangle+\|\mathbf{Z}_1\mathbf{F}_c^T+\mathbf{F}_r\mathbf{Z}_2^T\|_F^2.\end{aligned}$$

Therefore, the KKT conditions for the NMF problem are

$$\begin{aligned}&(\mathbf{F}_r\mathbf{F}_c^T-\mathbf{V}^*)\mathbf{F}_c\geq\mathbf{0},\quad \mathbf{F}_r\geq\mathbf{0},\quad (\mathbf{F}_r\mathbf{F}_c^T-\mathbf{V}^*)\mathbf{F}_c\circ\mathbf{F}_r=\mathbf{0},\\ &(\mathbf{F}_c\mathbf{F}_r^T-(\mathbf{V}^*)^T)\mathbf{F}_r\geq\mathbf{0},\quad \mathbf{F}_c\geq\mathbf{0},\quad (\mathbf{F}_c\mathbf{F}_r^T-(\mathbf{V}^*)^T)\mathbf{F}_r\circ\mathbf{F}_c=\mathbf{0},\\ &2\langle\mathbf{Z}_1\mathbf{Z}_2^T,(\mathbf{F}_r\mathbf{F}_c^T-\mathbf{V}^*)\rangle+\|\mathbf{Z}_1\mathbf{F}_c^T+\mathbf{F}_r\mathbf{Z}_2^T\|_F^2\geq 0,\quad \forall\begin{pmatrix}\mathbf{Z}_1\\ \mathbf{Z}_2\end{pmatrix}\in\mathbb{R}^{(n_1+n_2)\times k},\ \text{that satisfies}\\ &\mathbf{Z}_1\circ\mathbb{1}_{\mathbf{F}_r=\mathbf{0}}=\mathbf{0},\mathbf{Z}_2\circ\mathbb{1}_{\mathbf{F}_c=\mathbf{0}}=\mathbf{0}.\end{aligned}$$

Define the linear function $f_{\mathbf{F}_r,\mathbf{F}_c}:\mathbb{R}^{n_1\times k}\times\mathbb{R}^{n_2\times k}\to R^{n_1\times n_2}$, where $f_{\mathbf{F}_r,\mathbf{F}_c}(\mathbf{Z}_1,\mathbf{Z}_2)=\mathbf{Z}_1\mathbf{F}_c^T+\mathbf{F}_r\mathbf{Z}_2^T$. Consider the following linear subspace of $\mathbb{R}^{n_1\times k}\times\mathbb{R}^{n_2\times k}$:

$$e(\mathbf{F}_r,\mathbf{F}_c)\equiv\{(\mathbf{Z}_1,\mathbf{Z}_2)|\mathbf{Z}_1\circ\mathbb{1}_{\mathbf{F}_r=\mathbf{0}}=\mathbf{0},\mathbf{Z}_2\circ\mathbb{1}_{\mathbf{F}_c=\mathbf{0}}=\mathbf{0}\}.$$

and the following linear subspace of $\mathbb{R}^{n_1 \times n_2}$:

$$E(\mathbf{F}_r, \mathbf{F}_c) \equiv \{\mathbf{Z}_1 \mathbf{F}_c^T + \mathbf{F}_r \mathbf{Z}_2^T | (\mathbf{Z}_1, \mathbf{Z}_2) \in e(\mathbf{F}_r, \mathbf{F}_c)\},$$

Obviously, $f_{\mathbf{F}_r,\mathbf{F}_c}(e(\mathbf{F}_r, \mathbf{F}_c)) = E(\mathbf{F}_r, \mathbf{F}_c)$, and $e(\mathbf{F}_r, \mathbf{F}_c) \subset f^{-1}_{\mathbf{F}_r,\mathbf{F}_c}(E(\mathbf{F}_r, \mathbf{F}_c))$.

Essentially, if $(\mathbf{Z}_1, \mathbf{Z}_2) \in e(\mathbf{F}_r, \mathbf{F}_c)$ that $\mathbf{Z}_1$ and $\mathbf{Z}_2$ have zero entries, wherever $\mathbf{F}_r$ and $\mathbf{F}_c$ have zero entries. In particular, this means

$$(\mathbf{F}_r \mathbf{F}_c^T - \mathbf{V}^*)\mathbf{F}_c \circ \mathbf{Z}_1 = \mathbf{0}, \quad (\mathbf{F}_c \mathbf{F}_r^T - (\mathbf{V}^*)^T)\mathbf{F}_r \circ \mathbf{Z}_2 = \mathbf{0}.$$

Note that $f_{\mathbf{F}_r,\mathbf{F}_c}$ is not injective on $e(\mathbf{F}_r, \mathbf{F}_c)$. Especially, for all $1 \leq i \leq k$, note by $\mathbf{x}_i$ the matrix with the i-th column equal to that of $\mathbf{X}$ and 0 everywhere else, then $(\mathbf{f}_{c,i}, -\mathbf{f}_{r,i}) \in Ker(f_{\mathbf{F}_r,\mathbf{F}_c})$.

5.3.6 Projections

If $(\mathbf{F}_r, \mathbf{F}_c)$ satisfies the first-order optimality condition, then $\mathbf{F}_r \mathbf{F}_c^T$ is the projection of $\mathbf{V}^*$ in $E(\mathbf{F}_r, \mathbf{F}_c)$. This is true, because $\mathbf{F}_r \mathbf{F}_c^T \in E(\mathbf{F}_r, \mathbf{F}_c)$, and $\mathbf{V}^* - \mathbf{F}_r \mathbf{F}_c^T \in E(\mathbf{F}_r, \mathbf{F}_c)^\perp$. Note this fact by

$$\mathcal{P}_{E(\mathbf{F}_r,\mathbf{F}_c)}(\mathbf{V}^*) = \mathbf{F}_r \mathbf{F}_c^T.$$

In fact, if $(\mathbf{F}_r, \mathbf{F}_c)$ satisfies the first-order optimality condition, then $\mathbf{F}_r \mathbf{F}_c^T$ is the projection of $\mathbf{V}^*$ in a even larger set. The following closed convex cone contains $E(\mathbf{F}_r, \mathbf{F}_c)$:

$$\begin{aligned} F(\mathbf{F}_r, \mathbf{F}_c) \equiv \{&\mathbf{Z}_1 \mathbf{F}_c^T + \mathbf{F}_r \mathbf{Z}_2^T | (\mathbf{Z}_1, \mathbf{Z}_2) \in \mathbb{R}^{n_1 \times k} \times \mathbb{R}^{n_2 \times k}, \\ &\langle \mathbf{Z}_1, (\mathbf{F}_r \mathbf{F}_c^T - \mathbf{V}^*)\mathbf{F}_c \rangle \geq 0, \langle \mathbf{Z}_2, (\mathbf{F}_c \mathbf{F}_r^T - (\mathbf{V}^*)^T)\mathbf{F}_r \rangle \geq 0\}. \end{aligned}$$

By definition of $F(\mathbf{F}_r, \mathbf{F}_c)$, if $\mathbf{Z}_1 \mathbf{F}_c^T + \mathbf{F}_r \mathbf{Z}_2^T \in F(\mathbf{F}_r, \mathbf{F}_c)$, then

$$\langle \mathbf{Z}_1 \mathbf{F}_c^T + \mathbf{F}_r \mathbf{Z}_2^T, \mathbf{V}^* - \mathbf{F}_r \mathbf{F}_c^T \rangle \leq 0.$$

This means that $\mathbf{F}_r \mathbf{F}_c^T$ is in the polar cone of $F(\mathbf{F}_r, \mathbf{F}_c)$ (noted as $\mathbf{F}_r \mathbf{F}_c^T \in F(\mathbf{F}_r, \mathbf{F}_c)^\ominus$, [100, Chap.6]). Together with $\mathbf{F}_r \mathbf{F}_c^T \in F(\mathbf{F}_r, \mathbf{F}_c)$, and $\mathbf{V}^* - \mathbf{F}_r \mathbf{F}_c^T \perp \mathbf{F}_r \mathbf{F}_c^T$, this constitutes a characterization of projection of $\mathbf{V}^*$ in $F(\mathbf{F}_r, \mathbf{F}_c)$. That is

$$\mathcal{P}_{F(\mathbf{F}_r,\mathbf{F}_c)}(\mathbf{V}^*) = \mathbf{F}_r \mathbf{F}_c^T.$$

Define $\hat{\Pi}(\mathbf{F}_r^*, \mathbf{F}_c^*) = \mathcal{P}_{e(\mathbf{F}_r,\mathbf{F}_c)}(\mathbf{F}_r^* \Pi, \mathbf{F}_c^* \Pi)$, for any K-by-K permutation matrix Π ($\hat{\Pi}$ is the composition of the permutation with projection onto the support of $(\mathbf{F}_r, \mathbf{F}_c)$. I will also use $\hat{\Pi}(\mathbf{F}_r^*)$ (and $\hat{\Pi}(\mathbf{F}_c^*)$) to note the projection of $\mathbf{F}_r^*$ (and $\mathbf{F}_c^*$) onto the appropriate support, by an abuse of notation.

Consider the following subset of $\mathbb{R}^{n_1 \times n_2}$:

$$\{\mathbf{Z}_1 \mathbf{Z}_2^T | (\mathbf{Z}_1, \mathbf{Z}_2) \in e(\mathbf{F}_r, \mathbf{F}_c)\}.$$

This set contains $\hat{\Pi}(\mathbf{F}_r^*, \mathbf{F}_c^*)$, but is in general not convex. This makes it difficult to consider the projection of $\mathbf{V}^*$ in it. On the other hand,

$$G(\mathbf{F}_r, \mathbf{F}_c) \equiv \text{conv}\{\mathbf{Z}_1 \mathbf{Z}_2^T | (\mathbf{Z}_1, \mathbf{Z}_2) \in e(\mathbf{F}_r, \mathbf{F}_c)\},$$

the convex hull of the previous subset, is a convex cone. In fact

$$G(\mathbf{F}_r, \mathbf{F}_c) \subset \{\mathbf{X} \in \mathbb{R}^{n_1 \times n_2} | \mathbf{X} \circ (\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)\mathbf{F}_c\mathbf{F}_r^T(\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*) = 0\}.$$

Since both $(\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)\mathbf{F}_c$ and $\mathbf{F}_r^T(\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)$ are nonnegative, the support of the product is $\{(i,j)|supp((\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)_i\mathbf{F}_c) \bigcap supp((\mathbf{F}_c\mathbf{F}_r^T - (\mathbf{V}^*)^T)_j\mathbf{F}_r) \neq \emptyset\}$. Therefore, any element of $G(\mathbf{F}_r, \mathbf{F}_c)$ has a support of empty intersection with $(\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)\mathbf{F}_c\mathbf{F}_r^T(\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)$.

We have $\mathcal{P}_{G(\mathbf{F}_r,\mathbf{F}_c)}(\mathbf{V}^*) - \hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T \geq 0$.

5.3.7 Applying second order condition

Now suppose that $(\mathbf{F}_r, \mathbf{F}_c)$ is a local minimum, *i.e.*, it verifies both (5.8) and (5.9).

Note that if $\mathcal{P}_{G(\mathbf{F}_r,\mathbf{F}_c)}(\mathbf{V}^*) = \hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T$, then $\hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T - \mathbf{V}^* = \mathcal{P}_{G(\mathbf{F}_r,\mathbf{F}_c)^\perp}(\mathbf{V}^*)$. If this is the case, we can prove the following inequality, which is similar to the second inequality in [90, Corollary4.1]:

$$\sum_{j=1}^{k} \|(\mathbf{F}_r - \hat{\Pi}(\mathbf{F}_r^*))\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c^T + \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T(\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_c^*))^T\|_F^2 \geq 2\|\mathbf{F}_r\mathbf{F}_c^T - \hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T\|_F^2. \quad (5.10)$$

To see why (5.10) is true, consider the second-order condition with $\mathbf{Z}_1 = (\mathbf{F}_r - \hat{\Pi}(\mathbf{F}_r^*))\mathbf{e}_j$ and $\mathbf{Z}_2 = (\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_c^*))\mathbf{e}_j$, for $1 \leq j \leq k$. Therefore,

$$\begin{aligned}&2\langle(\mathbf{F}_r - \hat{\Pi}(\mathbf{F}_r^*))\mathbf{e}_j\mathbf{e}_j^T(\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_c^*))^T, (\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*)\rangle + \\ &\|(\mathbf{F}_r - \hat{\Pi}(\mathbf{F}_r^*))\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c^T + \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T(\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_c^*))^T\|_F^2 \geq 0.\end{aligned}$$

We know that

$$\begin{aligned}&\langle(\mathbf{F}_r - \hat{\Pi}(\mathbf{F}_r^*))\mathbf{e}_j\mathbf{e}_j^T(\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_c^*))^T, \mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*\rangle \\ =&\langle\mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c + \hat{\Pi}(\mathbf{F}_r^*)\mathbf{e}_j\mathbf{e}_j^T\hat{\Pi}(\mathbf{F}_c^*)^T, \mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*\rangle \\ =&\langle\hat{\Pi}(\mathbf{F}_r^*)\mathbf{e}_j\mathbf{e}_j^T\hat{\Pi}(\mathbf{F}_c^*)^T - \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c, \mathbf{F}_r\mathbf{F}_c^T - \mathbf{V}^*\rangle \\ =&\langle\hat{\Pi}(\mathbf{F}_r^*)\mathbf{e}_j\mathbf{e}_j^T\hat{\Pi}(\mathbf{F}_c^*)^T - \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c, \mathbf{F}_r\mathbf{F}_c^T - \hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T\rangle \\ &+ \langle\hat{\Pi}(\mathbf{F}_r^*)\mathbf{e}_j\mathbf{e}_j^T\hat{\Pi}(\mathbf{F}_c^*)^T - \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c, \hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T - \mathbf{V}^*\rangle.\end{aligned}$$

Since $\hat{\Pi}(\mathbf{F}_r^*)\mathbf{e}_j\mathbf{e}_j^T\hat{\Pi}(\mathbf{F}_c^*)^T - \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c^T \in G(\mathbf{F}_r, \mathbf{F}_c)$, the second term is zero.

In this case, we have

$$\begin{aligned}&\sum_{j=1}^{K} \|(\mathbf{F}_r - \hat{\Pi}(\mathbf{F}_r^*))\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c^T + \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T(\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_c^*))^T\|_F^2 \\ -2&\sum_{j=1}^{K}\langle\mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_r^*)\mathbf{e}_j\mathbf{e}_j^T\hat{\Pi}(\mathbf{F}_c^*)^T, \mathbf{F}_r\mathbf{F}_c^T - \hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T\rangle \\ =&\sum_{j=1}^{K} \|(\mathbf{F}_r - \hat{\Pi}(\mathbf{F}_r^*))\mathbf{e}_j\mathbf{e}_j^T\mathbf{F}_c^T + \mathbf{F}_r\mathbf{e}_j\mathbf{e}_j^T(\mathbf{F}_c - \hat{\Pi}(\mathbf{F}_c^*))^T\|_F^2 - 2\|\mathbf{F}_r\mathbf{F}_c^T - \hat{\Pi}(\mathbf{F}_r^*)\hat{\Pi}(\mathbf{F}_c^*)^T\|_F^2 \geq 0,\end{aligned}$$

which then leads to (5.10).

For any local minimum, (5.10) bounds the approximation error (right-hand side) by a quantity connected to the error in the factor space (left-hand side), regardless of the permutation of columns in the factor space.

In [90], once the similar inequality [90, Corollary4.1] is established, an upper bound is then proposed for the left-hand side, so as to prove that the global error is actually zero. Therefore the local minimum is actually a global minimum as well. This is achieved using the properties of the orthogonal matrix which minimizes the left-hand side. Here instead of considering an orthogonal matrix, we use a permutation matrix Π, since we are only allowed to consider points in the nonnegative orthant. It is not clear how we can translate this into the nonnegative case, since we would need $\hat{\Pi}(\mathbf{F}_r^*)\mathbf{R}$ to be nonnegative as well, therefore the properties of $\mathbf{R}$ would change.

Chapter

6 Nonnegative matrix factorization for time series recovery from a few temporal aggregates

Contents

6.1 Introduction

In this chapter, we propose a new matrix recovery method using nonnegative matrix factorization (NMF, [4]) where matrix columns represent time series at a fine temporal scale. Moreover, only temporal aggregates of these time series are observed.

As explained in previous chapters, the motivation comes from the context of electricity load balancing, where time series represent electric power consumption. To avoid failure in the electricity network, suppliers are typically required by transmission system operators (TSO) to supply as much electricity as their consumers consume at every moment. This mechanism is called balancing. In the context of an open electricity market, all market participants, such as suppliers, utility traders, and large consumers, have a balance responsibility: any imbalance caused within the perimeter of a participant is billed by the TSO. To calculate the imbalance caused by a market participant, one needs an estimation of the consumption and production within its perimeter at a small temporal scale, for example, half-hourly ([10], [101], [102]).

This chapter is a joint work with Yohann De Castro, Yannig Goude and Georges Hébrail. It is based on the ICML conference paper [2].

However, for many customers (for instance residential) within a perimeter, electricity consumption is not recorded at that scale. Although smart meter readings may be recorded locally up to every minute, utility companies often have very limited access to such data, due to data transmission and processing costs and/or privacy issues. Following a fixed schedule, cumulative consumption of each meter is recorded by the utility company, for instance every day or every month. By differentiating consecutive readings, the utility obtains the consumption of a customer between two reading dates. Currently, TSOs use proportional rules to reconstitute consumption from these measurements, based on national consumption profiles adjusted by temperature. In this chapter, we develop an NMF-based matrix recovery method providing a solution to consumption reconstitution from such temporal aggregates.

Recent advances in matrix completion have made it clear that when a large number of individuals and features are involved, even partial data could be enough to recover much of lost information, thanks to the low-rank property [86]: although the whole data matrix $\mathbf{V}^* \in \mathbb{R}^{n_1 \times n_2}$ is only partially known, if $\mathbf{V}^* = \mathbf{F}_r \mathbf{F}_c^T$, where $\mathbf{F}_r \in \mathbb{R}^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}^{n_2 \times k}$, with k much smaller than both n_1 and n_2, one could recover $\mathbf{V}^*$ entirely under some conditions over the sampling process.

6.1.1 Temporal aggregated electricity consumption

In this chapter, we address electricity consumption reconstitution as a matrix recovery problem. Consider the electricity consumption of n_2 consumers during n_1 periods. Since consumption is always positive, the n_2 time series are organized into a nonnegative matrix $\mathbf{V}^* \in \mathbb{R}_+^{n_1 \times n_2}$. An entry of this matrix, $v_{t,n}^*$ represents, for example, the electricity consumption of Consumer n for Period t.

Information about consumption is revealed as meter readings which do not correspond to matrix entries but to cumulative sums of each column of $\mathbf{V}^*$: at a meter-reading date t, we observe that Consumer n has consumed $\sum_{i=1}^{t} v_{i,n}^*$ since the first period. Several readings are available for each consumer.

An alternative matrix representation could be to define entries directly as the cumulative consumption since the first period. Again, this matrix has missing values and a matrix completion algorithm can be applied. However, this cumulative matrix has increasing columns, which is quite different from matrices considered in the standard matrix completion literature, where matrix completion error is typically bounded by the largest entry of the matrix.

We represent meter readings as linear measures on the consumption matrix $\mathbf{V}^*$. Temporal aggregates are derived from meter readings by differentiating consecutive meter readings: we will consider these temporal aggregates as "observations" in the rest of the chapter. We consider D scalar observations, represented by a data vector $\boldsymbol{\alpha} \equiv \mathcal{A}(\mathbf{V}^*) \in \mathbb{R}_+^D$, where $\mathcal{A}$ is a D-dimensional linear operator. To recover $\mathbf{V}^*$ from $\boldsymbol{\alpha}$, we look for a low-rank NMF of $\mathbf{V}^*$: $\mathbf{F}_r \mathbf{F}_c^T \simeq \mathbf{V}^*$, where $\mathbf{F}_r \in \mathbb{R}_+^{n_1 \times k}, \mathbf{F}_c \in \mathbb{R}_+^{n_2 \times k}$. The columns of $\mathbf{F}_r$ are k nonnegative factors, which can be interpreted as typical profiles of the time series, and the columns of $\mathbf{F}_c$ as the weights of each individual on each profile. The problem is formalized as the minimization of a quadratic loss function under nonnegativity and data constraints, as in (5.6):

$$\begin{aligned} \min_{\mathbf{V},\, \mathbf{F}_r,\, \mathbf{F}_c} \quad & \ell(\mathbf{V}, \mathbf{F}_r, \mathbf{F}_c) = \|\mathbf{V} - \mathbf{F}_r \mathbf{F}_c^T\|_F^2 \\ \text{s.t.} \quad & \mathbf{V} \geq \mathbf{0}, \quad \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}, \quad \mathcal{A}(\mathbf{V}) = \boldsymbol{\alpha}, \end{aligned} \tag{6.1}$$

where $\mathbf{X} \geq \mathbf{0}$ (or $\mathbf{x} \geq \mathbf{0}$) means that the matrix $\mathbf{X}$ (or the vector $\mathbf{x}$) is element-wise nonnegative.

Note the difference between (6.1) and another potential estimator, as in (5.7),

$$\begin{aligned} \min_{\mathbf{F}_r,\, \mathbf{F}_c} \quad & \hat{\ell}(\mathbf{F}_r, \mathbf{F}_c) = \|\mathcal{A}(\mathbf{F}_r \mathbf{F}_c^T) - \boldsymbol{\alpha}\|_2^2 \\ \text{s.t.} \quad & \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}, \end{aligned} \tag{6.2}$$

studied in [79]. If $\mathbf{V}$ is a solution to (6.1), it satisfies exactly the measurement constraint, but is approximately low-rank, while $\mathbf{F}_r \mathbf{F}_c^T$, a solution to (6.2) is exactly of low rank, but only matches the measurements approximately. Since in our application, the estimated time series matrix is to be used for billing, the match to metering data is essential. Therefore, we use (6.1) in this chapter.

6.1.2 Autocorrelation

Electricity consumption time series has the nice property of having high temporal autocorrelation. In order to better recovery consumption from temporal aggregates, we propose to exploit this information.

We suppose that the temporal autocorrelation of each individual considered in the dataset is available to us. In practice, this information could be obtained by exploiting historical data of each client. It is also possible to use the results of some prior modeling, which estimates the autocorrelation of each class of clients.

We add a penalty term to the base algorithm to take advantage of this information. Efficient computation is then deduce by using recent convex relaxation of quadratically constrained quadratic programs [103].

6.1.3 Prior works

The measurement operator $\mathcal{A}$ considered here is a special instance of the trace regression model [104] which generalizes the matrix completion setting. In matrix completion, each measurement is exactly one entry. Various forms of linear measurements other than matrix completion have been considered for matrix recovery without nonnegativity [78, 85, 105].

The NMF literature is generally focused on full observation [6, 106], or on matrix completion [107, 108] Random projection measurements are used in an NMF context in [109], where a maximum likelihood estimator is developed based on a specific generative model in neural imaging. The particular form of measurement operator considered here arises from meter reading, and can be used in other fields, such as Internet traffic matrix estimation [79]. Because of our choice of estimator (6.1) over estimator (6.2), we derive a novel algorithm for this measurement operator, which has a smaller time complexity than previously studied ones (more details in Section 6.2.1).

The idea of imposing time series structure on matrix factorization is not new. Previous approaches combining matrix factorization and autoregressive structure are often focused on obtaining factors that are more smooth and/or sparse, both in NMF [22–24] and without nonnegativity [25, 26]. Our objective is different from these studies: we try to further improve the matrix recovery by constraining temporal correlation on individual time series (not factors). We use a recent convex relaxation of quadratically constrained quadratic programs [103] to deduce a closed-form projection step in this case.

We propose an algorithm to solve (6.1) in Section 6.2.1. To take into account individual autocorrelation, a second algorithm is proposed in Section 6.2.2. In Section 6.3, both algorithms are validated on synthetic and real electricity consumption datasets, compared to a linear benchmark and a state-of-art matrix completion method.

6.2 Reconstitution of time series with NMF

6.2.1 Iterative algorithm with simplex projection

We represent temporal aggregation by a linear operator $\mathcal{A}$. For each $1 \leq d \leq D$, the d-th measurement on $\mathbf{X}$, $\mathcal{A}(\mathbf{X})_d$, is the sum of several consecutive rows on one column of $\mathbf{X}$, that is,

$$\mathcal{A}(\mathbf{X})_d = \sum_{(t,n)\in I_d} x_{t,n},$$

where $I_d = \{(t,n)|t_0(d)+1 \leq t \leq t_0(d)+h(d), n = n_d\}$, is the index set over $h(d)$ consecutive periods of Consumer n_d, starting from Period $t_0(d)+1$. Each measurement covers a disjoint index set. All entries of $\mathbf{X}$ are not necessarily involved in the measurements.

Note that although we are focused on temporal aggregates in this chapter, when applicable, the results announced can also be generalized to general linear measurements (see Chapter 7 for more details).

A block Gauss-Seidel algorithm (Algorithm 1) is used to solve (6.1). We alternate by minimizing $\ell(\mathbf{V}, \mathbf{F}_r, \mathbf{F}_c)$ over $\mathbf{F}_r, \mathbf{F}_c$ or $\mathbf{V}$, keeping the other two matrices fixed. Methods from classical NMF problems are used to update $\mathbf{F}_r$ and $\mathbf{F}_c$ [18]. In the implementation, we use two variants that seem similarly efficient (more details in Section 6.3): Hierarchical Alternating Least Squares (HALS, [19]), and a matrix-base NMF solver with Nesterov-type acceleration (NeNMF, [20]).

When $\mathbf{F}_r$ and $\mathbf{F}_c$ are fixed, the optimization problem on $\mathbf{V}$ is equivalent to D simplex projection problems, one for each scalar measurement. For $1 \leq d \leq D$, we have to solve

$$\begin{aligned} \min_{\mathbf{v}_{I_d}} \quad & \|\mathbf{v}_{I_d} - (\mathbf{F}_r)_{t_0(d)+1\leq t\leq t_0(d)+h(d)}(\mathbf{F}_c)_{n_d}\|^2 \\ \text{s.t.} \quad & \mathbf{v}_{I_d} \geq 0, \quad \mathbf{v}_{I_d}^T\mathbf{1} = \alpha_d, \end{aligned} \tag{6.3}$$

where $(\mathbf{F}_r)_{t_0(d)+1\leq t\leq t_0(d)+h(d)}$ is the $h(d)$ rows of $\mathbf{F}_r$ which correspond to the periods of the measurement, and $(\mathbf{F}_c)_{n_d}$ is the n_d-th row of $\mathbf{F}_c$ stacked as a column vector. The simplex projection algorithm introduced by [21] solves this subproblem efficiently. Define the operator, $\mathcal{P}_A$, as the orthogonal projection into the simplex $A \equiv \{\mathbf{X} \in \mathbb{R}_+^{n_1\times n_2} | \mathcal{A}(\mathbf{X}) = \boldsymbol{\alpha}\}$. The simplex A is the intersection of the affine subspace $\{\mathbf{X} \in \mathbb{R}^{n_1\times n_2} | \mathcal{A}(\mathbf{X}) = \boldsymbol{\alpha}\}$ and the first orthant. Projector $\mathcal{P}_A$ encodes the measurement data $\boldsymbol{\alpha} = \mathcal{A}(\mathbf{V}^*)$. In Algorithm 1, we apply $\mathcal{P}_A$ to a working copy of $\mathbf{V}$ in order to obtain its projection in A.

Contrary to previously studied algorithms [79], by choosing estimator (6.1) over (6.2), the simplex projection step is separated from the classical NMF update steps in our algorithm. Instead of multiplying the rank and the complexity introduced by the number of measurements, we have an algorithm whose complexity is the sum of the two. In cases where the number of measurements is large, this difference can be crucial [1].

A classical stopping criterion in the NMF literature is based on Karush-Kuhn-Tucker (KKT) conditions on (6.1) [6, Section 3.1.7]. We calculate

$$\begin{aligned} \mathcal{R}(\mathbf{F}_r)_{i,j} &= |(\mathbf{F}_r\mathbf{F}_c^T - \mathbf{V})\mathbf{F}_c^T)_{i,j}|\mathbb{1}_{(\mathbf{F}_r)_{i,j}\neq 0}, \\ \text{and } \mathcal{R}(\mathbf{F}_c)_{i,j} &= |((\mathbf{F}_c\mathbf{F}_r^T - \mathbf{V}^T)\mathbf{F}_r^T)_{i,j}|\mathbb{1}_{(\mathbf{F}_c)_{i,j}\neq 0}. \end{aligned}$$

The algorithm is stopped if $\|\mathcal{R}(\mathbf{F}_r)\|_F^2 + \|\mathcal{R}(\mathbf{F}_c)\|_F^2 \leq \epsilon$, for a small threshold $\epsilon > 0$.

[1] This intuition is confirmed by the comparison of Algorithm 1 and our implementation of the algorithm proposed in [79].

Algorithm 1 Block coordinate descent for NMF from temporal aggregates

input $\mathcal{P}_A, 1 \leq K \leq \min\{T, N\}$
  Initialize $\mathbf{F}_r^0, \mathbf{F}_c^0 \geq 0, \mathbf{V}^0 = \mathcal{P}_A(\mathbf{F}_r^0\mathbf{F}_c^0), i = 0$
  while Stopping criterion is not satisfied **do**
    $\mathbf{F}_r^{i+1} = \text{Update}(\mathbf{F}_r^i, \mathbf{F}_c^i, \mathbf{V}^i)$
    $\mathbf{F}_c^{i+1} = \text{Update}(\mathbf{F}_r^{i+1}, \mathbf{F}_c^i, \mathbf{V}^i)$
    $\mathbf{V}^{i+1} = \mathcal{P}_A(\mathbf{F}_r^{i+1}\mathbf{F}_c^{i+1})$
    $i = i + 1$
  end while
output $\mathbf{V}^i \in A, \mathbf{F}_r^i \in \mathbb{R}_+^{n_1 \times k}, \mathbf{F}_c^i \in \mathbb{R}_+^{n_2 \times k}$

Convergence to a stationary point has been proved for past NMF solvers with the full observation or the matrix completion setting [18, 20]. Although the subproblems on $\mathbf{F}_r$ and $\mathbf{F}_c$ do not necessarily have unique optimum, the projection of $\mathbf{V}$ attains a unique minimizer. By [110, Proposition 5], all limiting points of Algorithm 1 is a stationary point.

Algorithm 1 can be generalized to other types of measurement operators $\mathcal{A}$, as long as a projection into the simplex defined by the data constraint $\mathcal{A}(\mathbf{X}) = \boldsymbol{\alpha}$ and the positivity constraint can be efficiently computed.

6.2.2 From autocorrelation constraint to penalization

In addition to the measurements in $\boldsymbol{\alpha}$, we have some prior knowledge on the temporal autocorrelation of the individuals. To take into account information about autocorrelation, we add a penalization term to the original matrix recovery problem, replacing (6.1) by:

$$\begin{aligned} \min_{\mathbf{V},\mathbf{F}_r,\mathbf{F}_c} \quad & \|\mathbf{V} - \mathbf{F}_r\mathbf{F}_c^T\|_F^2 - \lambda \sum_{n=1}^{N} \mathbf{v}_n^T \boldsymbol{\Delta}_{\rho_n} \mathbf{v}_n \\ \text{s.t.} \quad & \mathbf{V} \geq 0, \quad \mathbf{F}_r \geq 0, \quad \mathbf{F}_c \geq 0, \quad \mathcal{A}(\mathbf{V}) = \boldsymbol{\alpha}, \end{aligned} \tag{6.4}$$

where $\lambda \geq 0$ is a single fixed penalization parameter, and $\boldsymbol{\Delta}_{\rho_n}$ is a symmetric matrix described shortly after. In the rest of this section, we first show by Theorem 1, that with an appropriately chosen value of λ, adding the penalization term $\mathbf{v}_n^T \boldsymbol{\Delta}_{\rho_n} \mathbf{v}_n$ is equivalent to impose that the temporal autocorrelation of $\mathbf{v}_n$ to be at least equal to ρ_n, a prior threshold. Then we modify Algorithm 1 to solve this penalized problem.

For $1 \leq n \leq n_2$, suppose that the lag-1 autocorrelation of Individual n's time series is at least equal to a threshold ρ_n (e.g. from historical data, excluded from observed temporal aggregates), that is,

$$\sum_{t=1}^{n_1-1} v_{t+1,n} v_{t,n} \geq \rho_n \sum_{t=1}^{n_1} v_{t,n}^2. \tag{6.5}$$

Notice that with the lag matrix,

$$\boldsymbol{\Delta} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix},$$

we have $\sum_{t=1}^{n_1-1} v_{t+1,n} v_{t,n} = \mathbf{v}_n^T \mathbf{\Delta} \mathbf{v}_n$. Define $\mathbf{\Delta}_\rho \equiv \mathbf{\Delta} + \mathbf{\Delta}^T - 2\rho\mathbf{I}$, for a threshold $-1 \leq \rho \leq 1$. Inequality (6.5) is then equivalent to

$$\mathbf{v}_n^T \mathbf{\Delta}_{\rho_n} \mathbf{v}_n \geq 0. \tag{6.6}$$

Imposing (6.6) would require one to solve, at each iteration, n_2 quadratically constrained quadratic programs (QCQP) of the form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & ||\mathbf{x} - \mathbf{x}_0||^2 \\ \text{s.t.} \quad & \mathbf{x}^T \mathbf{S} \mathbf{x} \geq 0, \end{aligned} \tag{6.7}$$

where $\mathbf{S}$ is a general symmetric matrix, not necessarily semi-definite positive. This means that the QCQP is in general a non-convex problem. Let $\boldsymbol{\delta}$ be the vector of eigenvalues of $\mathbf{\Delta}$. By eigendecomposition, $\mathbf{S} = \mathbf{U}^T \mathbf{D} \mathbf{U}$. where the matrix $\mathbf{U}$ is orthogonal. The entries of $\boldsymbol{\delta}$ are the diagonal entries of $\mathbf{D}$. The following theorem justifies the choice of penalization term in (6.4), by showing with an appropriate λ, adding this penalization term is equivalent to imposing the autocorrelation constraint (6.5).

Theorem 1. *Suppose that δ_1, the largest eigenvalue of $\mathbf{S}$, is strictly positive. Suppose that $\mathbf{z}_0 \equiv \mathbf{U}\mathbf{x}_0$ has no zero component. Then there exists $0 \leq \lambda < \frac{1}{\delta_1}$, that verifies $\sum_{t=1}^{n_1} \delta_t \frac{z_{0,t}^2}{2(1-\lambda\delta_t)^2} = 0$, so that $\mathbf{x}^* \equiv (\mathbf{I} - \lambda\mathbf{S})^{-1}\mathbf{x}_0$ is an optimal solution of (6.7).*

Proof. We follow [103] to obtain a convex relaxation of (6.7).

Define $\mathbf{z} \equiv \mathbf{U}\mathbf{x}$, $\mathbf{z}_0 \equiv \mathbf{U}\mathbf{x}_0$, $y_t \equiv \frac{1}{2} z_t^2$, $\forall 1 \leq t \leq n_1$. Recall that $\delta_1 > 0$, and that $\forall t, 1 \leq t \leq n_1, z_{0,t} \neq 0$.

Problem (6.7) is equivalent to the non-convex problem

$$\begin{aligned} \min_{\mathbf{y},\mathbf{z}} \quad & \mathbf{1}^T \mathbf{y} - \mathbf{z}_0^T \mathbf{z} \\ \text{s.t.} \quad & -\boldsymbol{\delta}^T \mathbf{y} \leq 0, \quad \frac{1}{2} z_t^2 = y_t, \quad \forall 1 \leq t \leq n_1. \end{aligned} \tag{6.8}$$

Now consider its convex relaxation

$$\begin{aligned} \min_{\mathbf{y},\mathbf{z}} \quad & \mathbf{1}^T \mathbf{y} - \mathbf{z}_0^T \mathbf{z} \\ \text{s.t.} \quad & -\boldsymbol{\delta}^T \mathbf{y} \leq 0, \quad \frac{1}{2} z_t^2 - y_t \leq 0, \quad \forall 1 \leq t \leq n_1. \end{aligned} \tag{6.9}$$

By [103, Theorem 3], if $(\mathbf{z}^*, \mathbf{y}^*)$ is an optimal solution of (6.9), and if $\frac{1}{2}(z_t^*)^2 = y_t^*$, $\forall 1 \leq t \leq n_1$, then $(\mathbf{z}^*, \mathbf{y}^*)$ is also an optimal solution of (6.8), which makes $\mathbf{x}^* = \mathbf{U}^T \mathbf{z}^*$ an optimal solution of (6.7).

We will look for such a solution to (6.9) by examining its first-order conditions of optimality. Problem (6.9) is convex, and it verifies the Slater condition: $\exists(\hat{\mathbf{y}}, \hat{\mathbf{z}}), -\boldsymbol{\delta}^T \hat{\mathbf{y}} < 0, \frac{1}{2}\hat{z}_t^2 < \hat{y}_t, \forall 1 \leq t \leq n_1$. This is true, because $\delta_1 > 0$. We could choose an arbitrary value of $\hat{y}_1 > 0$ and strictly positive but small values for other components of $\hat{\mathbf{y}}$ so as to have $-\boldsymbol{\delta}^T \hat{\mathbf{y}} < 0$, and $\hat{\mathbf{z}} = \mathbf{0}$. Thus, Problem (6.9) always has an optimal solution, because the objective function is coercive over the constraint. This shows the existence of $(\mathbf{z}^*, \mathbf{y}^*)$.

Now we show that $\frac{1}{2}(z_t^*)^2 = y_t^*$, $\forall 1 \leq t \leq n_1$. The KKT conditions of (6.9) are verified by

$(\mathbf{z}^*, \mathbf{y}^*)$. In particular, there is some dual variable $\lambda \geq 0, \boldsymbol{\mu} \in \mathbb{R}_+^T$ that verifies,

$$\mathbf{1} - \lambda\boldsymbol{\delta} - \boldsymbol{\mu} = \mathbf{0}, \tag{6.10}$$

$$-\boldsymbol{\delta}^T\mathbf{y}^* \leq 0, \tag{6.11}$$

$$\lambda\boldsymbol{\delta}^T\mathbf{y}^* = 0, \tag{6.12}$$

$$-z_{0,t} + \mu_t z_t^* = 0, \quad \forall 1 \leq t \leq n_1, \tag{6.13}$$

$$\frac{1}{2}(z_t^*)^2 - y_t^* \leq 0, \quad \forall 1 \leq t \leq n_1, \tag{6.14}$$

$$\mu_t(\frac{1}{2}(z_t^*)^2 - y_t^*) = 0, \quad \forall 1 \leq t \leq n_1. \tag{6.15}$$

Since $z_{0,t} \neq 0$, we have $\mu_t \neq 0, z_t^* = \frac{1}{\mu_t} z_{0,t}, \forall 1 \leq t \leq n_1$ by (6.13). Therefore, by (6.15), $y_t^* = \frac{1}{2}(z_t^*)^2, \forall 1 \leq t \leq n_1$.

By (6.10), the values of $\mu_t = 1 - \lambda\delta_t$ can be deduced from that of λ. Since $\boldsymbol{\mu} > \mathbf{0}$, we obtain that $\lambda < \frac{1}{\delta_1}$.

By (6.13), $z_t^* = \frac{z_{0,t}}{1-\lambda\delta_t}, \forall 1 \leq t \leq n_1$. This shows that $\mathbf{x}^* = \mathbf{U}^T\mathbf{z}^* = \mathbf{U}^T(\mathbf{I} - \lambda\mathbf{D})^{-1}\mathbf{z}_0^* = (\mathbf{I} - \lambda\mathbf{S})^{-1}\mathbf{x}_0$ is an optimal solution for (6.7).

□

Theorem 1 shows that with a well chosen λ, the constraint in (6.7) can be replaced by a penalization.

There are in fact two cases. Either $\mathbf{x}_0$ verifies the constraint, in which case $\lambda = 0$, and $\mathbf{x}^* = \mathbf{x}_0$ is the solution. Otherwise, $\lambda > 0$. We replug the values of

$$y_t^* = \frac{1}{2}(z_t^*)^2 = \frac{z_{0,t}^2}{2(1 - \lambda\delta_t)^2}$$

back into (6.12), and obtain that λ verifies

$$\sum_{t=1}^{T} \frac{\delta_t z_{0,t}^2}{2(1 - \lambda\delta_t)^2} = 0.$$

When $\mathbf{F}_r$ and $\mathbf{F}_c$ are fixed, the subproblem of (6.4) on $\mathbf{V}$ can be separated into N constrained problems of the form,

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x} - \mathbf{x}_0\|^2 - \lambda\mathbf{x}^T\boldsymbol{\Delta}_{\rho_n}\mathbf{x}, \\ \text{s.t.} \quad & \mathbf{A}_n\mathbf{x} = \mathbf{c}_n, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{6.16}$$

where $\mathbf{x}_0$ is the n-th column of $\mathbf{F}_r\mathbf{F}_c^T$, $\mathbf{c}_n$ is the observations on the n-th column, and $\mathbf{A}_n$ is a matrix which encodes the measurement operator over that column. The following theorem shows how to solve this problem.

Theorem 2. *Suppose that* $\mathbf{S}$ *is a symmetric matrix with eigenvalues* $\boldsymbol{\delta}$*, and* $\lambda_1 > 0$*. Suppose* $\mathbf{A} \in \mathbb{R}^{m,l}$ *a full-rank matrix with* $m \leq l$*,* $\mathbf{x}_0 \in \mathbb{R}^l$*,* $\mathbf{c} \in \mathbb{R}^m$*,* $\lambda \geq 0$*. Define* $\mathbf{Q} \equiv (\mathbf{I} - \lambda\mathbf{S})^{-1}\mathbf{A}^T(\mathbf{A}(\mathbf{I} - \lambda\mathbf{S})^{-1}\mathbf{A}^T)^{-1}$*. If* $\lambda < \frac{1}{\delta_{\rho,1}}$*, then* $\mathbf{Qc} + (\mathbf{I} - \mathbf{QA})(\mathbf{I} - \lambda\mathbf{S})^{-1}\mathbf{x}_0$ *is a minimizer of*

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x} - \mathbf{x}_0\|^2 - \lambda\mathbf{x}^T\mathbf{S}\mathbf{x}, \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{c}, \end{aligned} \tag{6.17}$$

Proof. Let l be the dimension of $\mathbf{c}$. Define I_C as the indicator function for the constraint of (6.17), that is

$$\begin{aligned} I_C(\mathbf{x}) &= 0, \text{ if } \mathbf{A}\mathbf{x} = \mathbf{c}, \\ \text{and } I_C(\mathbf{x}) &= +\infty, \text{ if } \mathbf{A}\mathbf{x} \neq \mathbf{c}. \end{aligned}$$

Problem (6.17) is then equivalent to

$$\min_{\mathbf{x}} F(\mathbf{x}) \equiv \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2 - \frac{1}{2}\lambda \mathbf{x}^T \mathbf{S}\mathbf{x} + I_C(\mathbf{x}). \tag{6.18}$$

The subgradient of (6.18) is $\partial F(\mathbf{x}) = \{\mathbf{x} - \mathbf{x}_0 - \lambda \mathbf{S}\mathbf{x} - \mathbf{A}^T\boldsymbol{\epsilon} | \boldsymbol{\epsilon} \in \mathbb{R}^l\}$. When $\lambda < \frac{1}{\delta_1}$, (6.18) is convex. Therefore, $\mathbf{x}^*$ is a minimizer if and only if $0 \in \partial F(\mathbf{x})$, and $\mathbf{A}\mathbf{x}^* = \mathbf{c}$. That is, $\exists \boldsymbol{\epsilon} \in \mathbb{R}^l$,

$$\begin{aligned} &(\mathbf{I} - \lambda \mathbf{S})\mathbf{x}^* - \mathbf{x}_0 - \mathbf{A}^T\boldsymbol{\epsilon} = \mathbf{0}, \\ &\mathbf{A}\mathbf{x}^* = \mathbf{c}. \end{aligned}$$

The vector $\boldsymbol{\epsilon}$ thereby verifies $\mathbf{A}(\mathbf{I} - \lambda \mathbf{S})^{-1}(\mathbf{x}_0 + \mathbf{A}^T\boldsymbol{\epsilon}) = \mathbf{c}$.

The l-by-l matrix $\mathbf{A}(\mathbf{I} - \lambda \mathbf{S})^{-1}\mathbf{A}^T$ is invertible, because l is smaller than m, and $\mathbf{A}$ is of full rank (because each measurement covers disjoint periods). Therefore,

$$\begin{aligned} \boldsymbol{\epsilon} &= (\mathbf{A}(\mathbf{I} - \lambda \mathbf{S})^{-1}\mathbf{A}^T)^{-1}(\mathbf{c} - \mathbf{A}(\mathbf{I} - \lambda \mathbf{S})^{-1}\mathbf{x}_0), \\ \mathbf{x}^* &= (\mathbf{I} - \lambda \mathbf{S})^{-1}(\mathbf{x}_0 + \mathbf{A}^T\boldsymbol{\epsilon}) \\ &= \mathbf{Q}\mathbf{c} + (\mathbf{I} - \mathbf{Q}\mathbf{A})(\mathbf{I} - \lambda \mathbf{S})^{-1}\mathbf{x}_0. \end{aligned}$$

□

In our particular problem, the eigenvalues of $\boldsymbol{\Delta}_{\rho_n}$ are

$$\delta_{\rho_n,t} = 2\cos(\frac{t}{n_2+1}\pi) - 2\rho_n,$$

with t taking every value from 1 to n_2. This means that for most of the autocorrelation threshold that we could need to impose ($-1 \leq \rho_n \leq 1$), $\boldsymbol{\Delta}_{\rho_n}$ has both strictly positive and strictly negative eigenvalues, allowing the above theorems to apply.

Both $\mathbf{I} - \lambda\boldsymbol{\Delta}_{\rho_n}$ and $\mathbf{A}_n(\mathbf{I} - \lambda\boldsymbol{\Delta}_{\rho_n})^{-1}\mathbf{A}_n^T$ are invertible with $\lambda < \delta_{\rho_n,1}$. The matrix inversion only needs to be done once for each individual. After computing $\mathbf{Q}_n \equiv (\mathbf{I} - \lambda\boldsymbol{\Delta}_{\rho_n})^{-1}\mathbf{A}_n^T(\mathbf{A}_n(\mathbf{I} - \lambda\boldsymbol{\Delta}_{\rho_n})^{-1}\mathbf{A}_n^T)^{-1}$, $\mathbf{Q}_n\mathbf{c}_n$ and $(\mathbf{I} - \mathbf{Q}_n\mathbf{A}_n)(\mathbf{I} - \lambda\boldsymbol{\Delta}_{\rho_n})^{-1}$ for each n, we use Algorithm 2 to solve (6.4).

Choosing λ An optimal value of λ could be calculated. Substituting the values of $\mathbf{y}^*$ in (6.12), shows that the optimal λ is a root of the polynomial $\sum_{t=1}^{n_2} \delta_{\rho,t}\frac{z_{0,t}^2}{2(1-\lambda\delta_{\rho,t})^2}$. The root-finding is too expensive to perform at every iteration. However, the optimal λ verifies

$$0 < \lambda < \frac{1}{\delta_{\rho,1}},$$

where $\delta_{\rho,1} = 2\cos(\frac{1}{n_2+1}\pi) - 2\rho$ is the biggest eigenvalue of $\boldsymbol{\Delta}_\rho$. This gives us a good enough idea about how large a λ to use. In the numerical experiments, we chose $\lambda = \min(1, \frac{1}{2\max_n \delta_{\rho_n,1}})$ in the penalization when the constraint in (6.7) is active, and $\lambda = 0$ (no penalization) when the constraint is verified by $\mathbf{x}_0$.

Algorithm 2 Block coordinate descent for NMF from temporal aggregates and autocorrelation penalty

input $\rho_n, \mathbf{A}_n, \mathbf{Q}_n, \mathbf{Q}_n\mathbf{c}_n, \forall 1 \leq n \leq N$, and $1 \leq K \leq \min\{T, N\}$
  Initialize $\mathbf{F}_r^0, \mathbf{F}_c^0 \geq 0, \mathbf{V}^0 = \mathcal{P}_A(\mathbf{F}_r^0\mathbf{F}_c^0), i = 0$
  while Stopping criterion is not satisfied **do**
    $\mathbf{F}_r^{i+1} = \text{Update}(\mathbf{F}_r^i, \mathbf{F}_c^i, \mathbf{V}^i)$
    $\mathbf{F}_c^{i+1} = \text{Update}(\mathbf{F}_r^{i+1}, \mathbf{F}_c^i, \mathbf{V}^i)$
    for all $1 \leq n \leq N$ **do**
      $\mathbf{v}_n^{i+1} = (\mathbf{Q}_n\mathbf{c}_n + (\mathbf{I} - \mathbf{Q}_n\mathbf{A}_n)(\mathbf{I} - \lambda\mathbf{\Delta}_{\rho_n})^{-1}\mathbf{F}_r^{i+1}\mathbf{h}_n^{i+1})_+$
    end for
    $i = i + 1$
  end while
output $\mathbf{V}^i \in A, \mathbf{F}_r^i \in \mathbb{R}_+^{n_1 \times k}, \mathbf{F}_c^i \in \mathbb{R}_+^{n_2 \times k}$

6.3 Experimental results

6.3.1 Datasets

We use one synthetic dataset and three real-world electricity consumption datasets to evaluate the proposed algorithms. In each dataset, the individual autocorrelation is calculated on historical data from the corresponding datasets, not used for evaluation.

- **Synthetic data**: 20 independent Gaussian processes with Matern covariance function (shifted to be nonnegative) are sampled over 150 periods to form the factor matrix $\mathbf{F}_r$. A 20-by-120 weight matrix $\mathbf{F}_c$ is generated by sampling from a standard normal distribution truncated at 0, independently for each entry. The data matrix is obtained as $\mathbf{F}_r \times \mathbf{F}_c$ ($n_1 = 150, n_2 = 120$). This matrix is exactly of rank 20.
- **French electricity consumption** (proprietary dataset): daily consumption of 636 medium-voltage feeders gathering each around 1,500 consumers based near Lyon in France during 2012 ($n_1 = 365, n_2 = 636$).
- **Portuguese electricity consumption** [28] daily consumption of 370 Portuguese clients during 2014 ($n_1 = 365, n_2 = 370$).
- **Electricity consumption of small Irish companies** [29, 30] daily consumption of 426 small Irish companies during 200 days in 2010 ($n_2 = 200, n_2 = 426$).

6.3.2 Evaluation procedure

For each individual in a dataset, we generate observations by selecting a number of observation periods. The temporal aggregates are the sum of the time series between two consecutive observation periods. The observation periods are chosen in two possible ways: periodically (at regular intervals with the first observation period sampled at random), or uniformly at random. The regular intervals for periodic observations are $p \in \{2, 3, 5, 7, 10, 15, 30\}$. This is motivated by the real application where meter readings are recorded regularly. With random observations, we use sampling rates that are equivalent to the regular intervals. That is, the number of observations D verifies $\frac{D}{n_1 n_2} = \frac{1}{p} \in \{0.5, 0.33, 0.2, 0.14, 0.1, 0.07, 0.03\}$.

We apply the following methods to recover the data matrix from each set of sampled observations:

- `interpolation` Temporal aggregates are distributed equally over the covered periods.
- `softImpute` As an alternative method, we apply a state-of-art matrix completion algorithm to complete the cumulative matrix. The observed entries are the cumulative values of the column from the first period to the observation dates. We use a nuclear-norm minimization algorithm, implemented in the `R` package, `softImpute` [111], to complete the cumulative consumption matrix, before differentiating each column to the obtain recovered matrix. To choose the thresholding parameter, we use the warm start procedure documented in `softImpute`.
- `HALS`, and `NeNMF` These are the proposed matrix recovery algorithms using two classical $\mathbf{F}_r$ and $\mathbf{F}_c$ update implementations: `HALS`, and `NeNMF`. When autocorrelation penalization is used, we choose $\lambda = \min(1, \frac{1}{2\max_n \delta_{\rho_n,1}})$, as explained in the previous section. The rank used in proposed algorithms is chosen by a 5-fold cross validation procedure: we split the observations randomly into 5 folds, and apply the algorithm to 4 of the 5 folds with ranks $2 \leq k \leq 30$. We then calculate the ℓ_2-distance between the temporal aggregates on the recovered matrix with the 1-fold holdout. Repeating this procedure onto the 5 folds separately, we choose the rank which minimizes the average ℓ_2-distance, to perform the algorithm on all observations. The cross validation procedure is carried out on the unpenalized version of the algorithms, and used both in the validation of unpenalized and penalized algorithms.

With each recovered matrix $\mathbf{V}$ obtained in an algorithm run, we compute the relative root-mean-squared error (RRMSE):

$$\text{RRMSE}(\mathbf{V}, \mathbf{V}^*) = \frac{\|\mathbf{V} - \mathbf{V}^*\|_F}{\|\mathbf{V}^*\|_F}.$$

Each experiment (dataset, sampling scheme, sampling rate, recovery method, unpenalized or penalized) is run three times to calculate the average RRMSE. Qualitatively similar results have been obtained using other error metrics.

6.3.3 Results

In Figure 6.1, one example of each data is shown. In the experiment depicted by the examples, 20% random observations are sampled. That is, observations dates are chosen uniformly at random, with one temporal aggregate every five days in average. In each dataset, the ground truth is drawn in black. The unpenalized `NeNMF` recovered the times series in green. We can see that generally trends are well estimated. However, unpenalized recovery produces an estimation which is more wiggly than the ground truth. With the autocorrelation-penalized version of `NeNMF` (orange lines), this aspect is improved.

The average RRMSE of experiments is reported in Figure 6.2. The figure is zoomed to show the RRMSE of the proposed algorithms. Much higher error rates for reference methods are sometimes not shown.

On sample sets with random observation periods (lower panel), proposed methods (`HALS` and `NeNMF`, blue and purple lines), whether unpenalized (solid lines) or penalized (dashed lines), out-performs the `interpolation` benchmark (red solid lines) by large in all datasets. This is especially the case when the sampling rate is small, *i.e.* when the task is more difficult. On the Irish dataset (lower panel, furthest to the right), penalized `HALS` and `NeNMF` (dashed blue and purple lines) are an improvement to unpenalized `HALS` and `NeNMF` when the sampling rate is low.

With periodic observations (upper panel), the RRMSE is higher for every method. Proposed unpenalized methods, `HALS` and `NeNMF` (blue and purple solid lines) are equivalent to

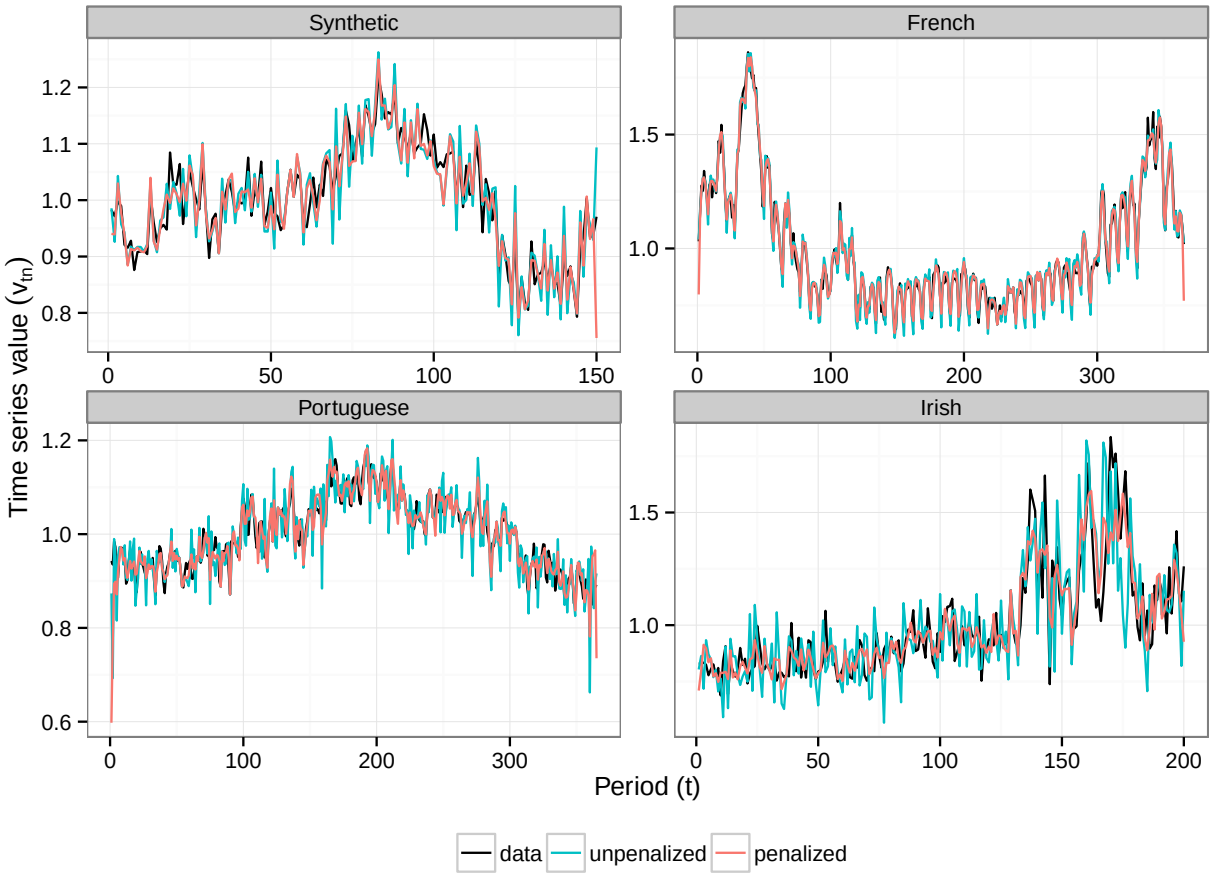


Figure 6.1: *Recovery examples from each of the four datasets. The data matrix is sampled with the scheme* `random`*, with 20% sampling rate in this experiment. The black line is the ground truth. The green line the recovery made by unpenalized* `NeNMF`*. The red line is the recovery made by penalized* `NeNMF`*.*

`interpolation` benchmark (red solid lines) for synthetic data, but sometimes worse for real datasets. Real electricity consumption has significant weekly periodicity, which is poorly captured by observations at similar periods. However, this shortcoming of the unpenalized method is more than compensated for by the penalization (dashed blue and purple lines).

We notice that penalized `HALS` and `NeNMF` consistently outperform `interpolation` with both observation schemes. This makes penalized methods particularly useful for the application of electricity consumption reconstitution, where it may be costly to install a random observation scheme, or to change the current periodic observation scheme to a random one.

It is also interesting to note that the rank chosen by the cross validation procedure is higher in higher sampling rate scenarios (Figure 6.3). This shows that the cross validation procedure is able to relax the rank constraint when more information is available in the data.

The traditional matrix completion method seems to fail in this application: softImpute (green solid lines) only has comparable results to `interpolation` or proposed methods in two of the four datasets, with 50% sampling rate in the random sampling scheme, which is the easiest case. In most cases, `softImpute` has an RRMSE much larger than 100%, and thus is not shown in the graphic. This indicates that the cumulative matrix considered in this application does not verify assumptions which guarantee matrix completion success.

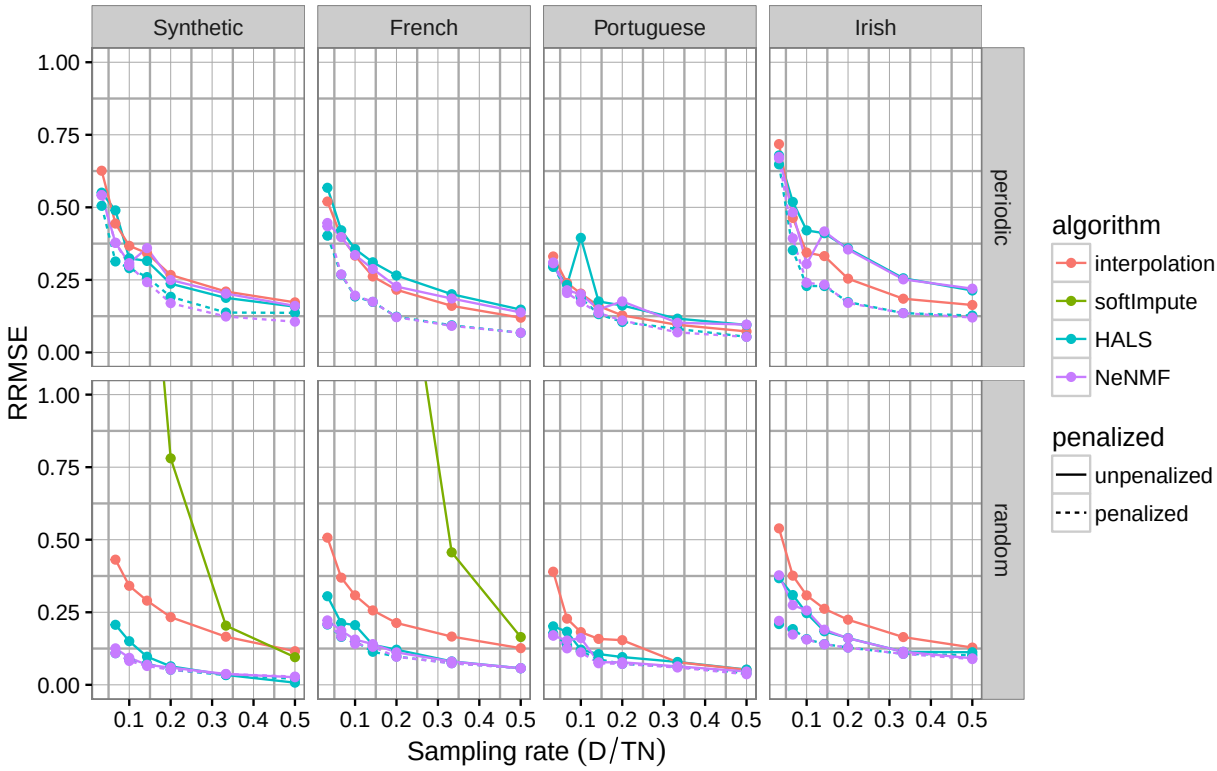


Figure 6.2: *Mean RRMSE of the recovered matrices over three separate runs over the four datasets. On the samples with random observation periods, proposed methods (`HALS` and `NeNMF`, blue and purple lines, both penalized and unpenalized) out-performs the `interpolation` benchmark (solid red line). On the samples with regular observation periods, unpenalized `HALS` and `NeNMF` (solid blue and purple lines) are similar to the `interpolation` benchmark, whiled penalized `HALS` and `NeNMF` (dashed blue and purple lines) are an important improvement. The `softImpute` method (solid green line) only has comparable performance in two of the datasets, in the easiest task (50% sampling rate at random periods). In most cases, RRMSE of `softImpute` is larger than 100%.*

6.4 Perspectives

Motivated by a new industrial application, we extended NMF to use temporal aggregates as input data, by adding a projection step into NMF algorithms. With appropriate projection algorithms, this approach could be further generalized to other types of data, such as disaggregating spatially aggregated data, or general linear measures. When such information is available, we introduce a penalization on individual autocorrelation, which improves the recovery performance of the base algorithm. This component can be generalized to larger lags (with a matrix $\mathbf{\Delta}$ with 1's further off the diagonal), or multiple lags (by adding several lag matrices together). It is also possible to generalize this approach to other types of expert knowledge, such as exogenous variables, which is discussed in the next chapter.

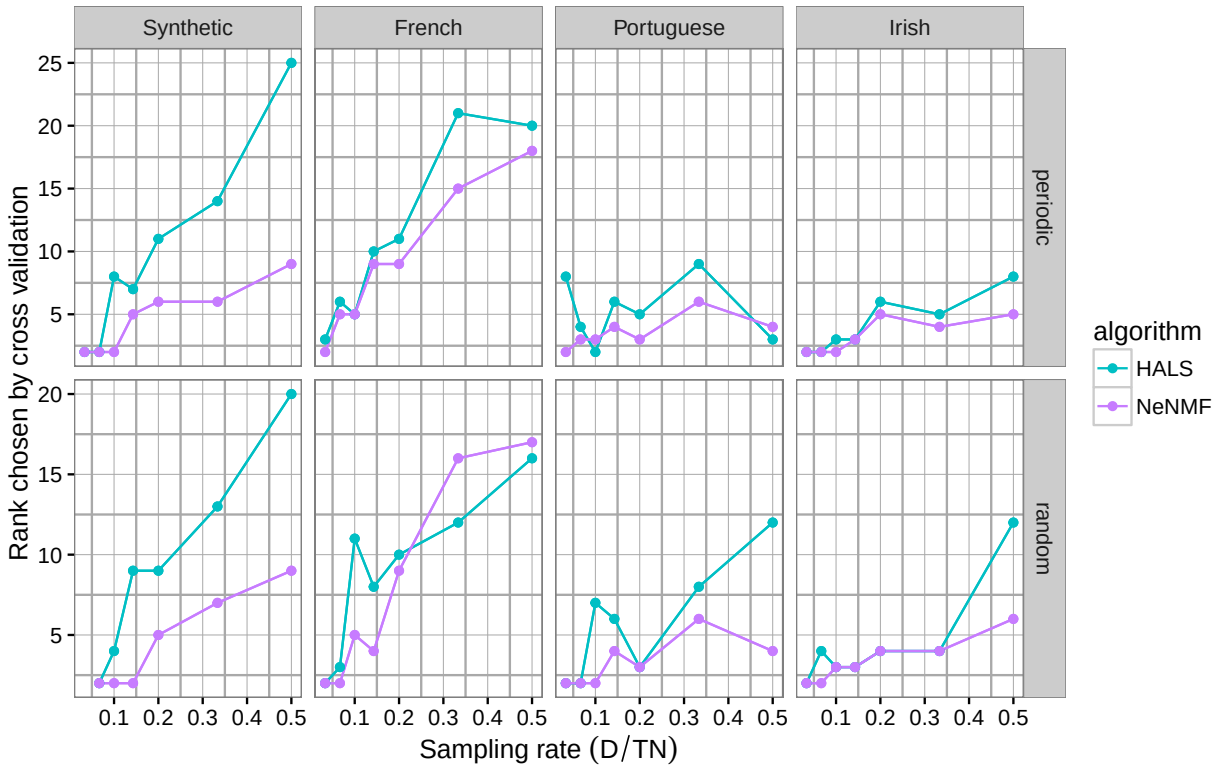


Figure 6.3: *The rank chosen by the cross validation procedure generally increases with the sampling rate, for the four datasets. This shows that the procedure is able to relax the rank constraint when more information is available in the data.*

Chapter

7 Time series recovery and prediction using NMF with side information

Contents

7.1 Introduction

In recent years, a large number of methods have been developed to improve matrix completion methods using side information [112–114]. By including features linked to the row and/or columns of a matrix, these methods have a better performance at estimating the missing entries.

This chapter is a joint work with Yohann De Castro, Yannig Goude, Georges Hébrail and Jean-Marc Azaïs. It is based on the submitted article [3].

In this chapter, we generalize this idea to nonnegative matrix factorization. Although more difficult to identify, NMF often has a better empirical performance, and provides factors that are more interpretable in an appropriate context. We propose an NMF method that takes into account side information. Given some observations of a matrix, this method jointly estimates the nonnegative factors of the matrix, and regression models of these factors on the side information. This allows us to improve the matrix recovery performance of NMF. Moreover, using the regression models, we can predict the value of interest for new rows and columns that are previously unseen. We develop this method in the general matrix recovery context, where linear measurements are observed instead of matrix entries.

This choice is again motivated by applications in electricity consumption. We are interested in estimating and predicting the electricity load from temporal aggregates. In the context of load balancing in the power market, electric transmission system operators (TSO) of the electricity network are typically legally bound to estimate the electricity consumption and production at a small temporal scale (half-hourly or hourly), for market participants within their perimeter, *i.e.* utility providers, traders, large consumers, groups of consumers, *etc.* [2]. Most traditional electricity meters do not provide information at such a fine temporal scale. Although smart meters can record consumption locally every minute, the usage of such data can be extremely constrained for TSOs, because of the high cost of data transmission and processing and/or privacy issues. Nowadays, TSOs often use regulator-approved proportional consumption profiles to estimate market participants' load. In Chapter 6, we proposed to solve the estimation problem by NMF using temporal aggregate data.

Using the method developed in this chapter, we put in parallel temporal aggregate data with features that are known to have a correlation with electricity consumption, such as the temperature, the day of the week, or the type of client. This not only improves the performance of load estimation, but also allows us to predict future load for users in the dataset, and estimate and predict the consumption of new users previously unseen. In electrical power networks, load prediction for new periods is useful for balancing offer-demand on the network, and prediction for new individuals is useful for network planning. Moreover, by examining the relationship between external features, the factors produced by NMF can be much more interpretable.

In the rest of this section, we introduce the general framework of this method, and the related literature. In Section 7.2, we deduce a sufficient condition on the side information for the NMF to be unique. In Section 7.3, we present HALSX, an algorithm which solves the NMF with side information problem, that we prove to converge to stationary points. In Section 6.3, we present experimental results, applying HALSX on simulated and real datasets, both for the electricity consumption application, and for a standard task in collaborative filtering.

7.1.1 General model definition

In this chapter, we use general linear measurements. We do not restrict us to temporally aggregated observations.

As mentioned in Chapter 5, we are interested in reconstructing a nonnegative matrix $\mathbf{V}^* \in \mathbb{R}_+^{n_1 \times n_2}$, from N linear measurements,

$$\boldsymbol{\alpha} = \mathcal{A}(\mathbf{V}^*) \in \mathbb{R}^N, \tag{7.1}$$

where $\mathcal{A} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^N$ is a linear operator. Formally, $\mathcal{A}$ can be represented by $\mathbf{A}_1$, ..., $\mathbf{A}_N$, N design matrices of dimension $n_1 \times n_2$, and each linear measurement can be represented by

$$\alpha_i = \mathrm{Tr}(\mathbf{V}^* \mathbf{A}_i^T) = \langle \mathbf{V}^*, \mathbf{A}_i \rangle. \tag{7.2}$$

The design matrices $\mathbf{A}_1$, ..., $\mathbf{A}_N$ are called *masks*.

Moreover, we suppose that the matrix of interest, $\mathbf{V}^*$, stems from a generative low-rank nonnegative model, in the following sense:

1. The matrix $\mathbf{V}^*$ is of *nonnegative rank* k, with $k \ll n_1, n_2$. This means, k is the smallest number so that we can find two nonnegative matrices $\mathbf{F}_r \in \mathbb{R}_+^{n_1 \times k}$ and $\mathbf{F}_c \in \mathbb{R}_+^{n_2 \times k}$ satisfying

$$\mathbf{V}^* = \mathbf{F}_r \mathbf{F}_c^T.$$

   Note that this implies that $\mathbf{V}^*$ is of rank at most k, and therefore is of low rank.

2. There are d_1 row features $\mathbf{X}_r \in \mathbb{R}^{n_1 \times d_1}$ and d_2 column features $\mathbf{X}_c \in \mathbb{R}^{n_2 \times d_2}$ connected to each row and column of $\mathbf{V}^*$. We note by $\mathbf{x}_r^i$ the i-th row of $\mathbf{X}_r$, and by $\mathbf{x}_c^i$ the i-th row of $\mathbf{X}_c$. There are two link functions $f_r : \mathbb{R}^{d_1} \to \mathbb{R}^k$ and $f_c : \mathbb{R}^{d_2} \to \mathbb{R}^k$, so that

$$\mathbf{F}_r = (f_r(\mathbf{X}_r))_+,$$
$$\mathbf{F}_c = (f_c(\mathbf{X}_c))_+,$$

   where $f_r(\mathbf{X}_r) \in \mathbb{R}^{n_1 \times k}$ is the matrix obtained by stacking row vectors $f_r(\mathbf{x}_r^i)$, for $1 \leq i \leq n_1$ (*idem* for $f_c(\mathbf{X}_c) \in \mathbb{R}^{n_2 \times k}$), and $(\cdot)_+$ is the ramp function which corresponds to thresholding operation at 0 for any matrix or vector.

In this general setting, the features $\mathbf{X}_r$ and $\mathbf{X}_c$, the measurement operator $\mathcal{A}$, and the measurements $\boldsymbol{\alpha}$ are observed. The objective is to estimate the true matrix $\mathbf{V}^*$ as well as the factor matrices $\mathbf{F}_r$ and $\mathbf{F}_c$, by estimating the link functions f_r and f_c.

To obtain a solution to this matrix recovery problem, we minimize the quadratic error of the matrix factorization. In Section 7.3, we will propose an algorithm for the following optimization problem:

$$\begin{aligned} \min_{\mathbf{V}, f_r \in F_r^k, f_c \in F_c^k} \quad & \|\mathbf{V} - (f_r(\mathbf{X}_r))_+ (f_c(\mathbf{X}_c))_+^T\|_F^2 \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{V}) = \mathbf{b}, \quad \mathbf{V} \geq \mathbf{0}, \end{aligned} \tag{7.3}$$

where $F_r \subseteq (\mathbb{R})^{\mathbb{R}^{d_1}}$ and $F_c \subseteq (\mathbb{R})^{\mathbb{R}^{d_2}}$ are some functional spaces in which the row and column link functions are to be searched.

By specializing $\mathbf{X}_r$, $\mathbf{X}_c$, and $\mathcal{A}$, or restricting the search space of f_r and f_c, this general model includes a number of interesting applications, old and new.

The masks $\mathbf{A}_1$, ..., $\mathbf{A}_N$

As mentioned in Chapter 5, varying the masks allows us to consider various applications in the matrix recovery context.

- **Complete observation**: $N = n_1 n_2, \mathbf{A}_{i_1,i_2} = \mathbf{e}_{i_1} \mathbf{e}_{i_2}^T$, where $\mathbf{e}_i$ is the i-th canonical vector. This means every entry of $\mathbf{V}^*$ is observed.
- **Matrix completion**: the set of masks is a subset of complete observation masks, with $N < n_1 n_2$.
- **Matrix sensing**: the design matrices $\mathbf{A}_i$ are random matrices, sampled from a certain probability distribution. Typically, the probability distribution needs to verify certain conditions, so that with a large probability, $\mathcal{A}$ verifies the Restricted Isometry Property [78].

- **Rank-one projections** [84, 85]: the design matrices are random rank-one matrices, that is $\mathbf{A}_i = \boldsymbol{\alpha}_i \boldsymbol{\beta}_i^T$, where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are respectively random vectors of dimension n_1 and n_2. The main advantage to this setting is that much less memory is needed to store the masks, since we can store the vectors $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ (dimension-$(n_1 + n_2)$) instead of $\mathbf{A}_i$ (dimension-$(n_1 \times n_2)$). In [84, 85], theoretical properties are proved for the case where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are vectors with independent Gaussian entries and/or drawn uniformly from the vectors of the canonical basis.
- **Temporal aggregate measurements**: in this case, the matrix is composed of n_1 time series concerning n_2 individuals, and each measure is a temporal aggregate of the time series of an individual. The design matrices are defined as $\mathbf{A}_i = \sum_{t=t_0(i)+1}^{t_0(i)+h(i)} \mathbf{e}_t \mathbf{e}_{s_i}^T$, where s_i is the individual concerned by the i-th measure, $t_0(i) + 1$ the first period covered by the measure, and $h(i)$ the number of periods covered by the measure.

The features $\mathbf{X}_r$ and $\mathbf{X}_c$

- **Individual features**: $\mathbf{X}_r = \mathbf{I}_{n_1}, \mathbf{X}_c = \mathbf{I}_{n_2}$. Basically, no side information is available. The row individuals and column individuals are each different.
- **General numeric features**: $\mathbf{X}_r \in \mathbb{R}^{n_1 \times d_1}$ and $\mathbf{X}_c \in \mathbb{R}^{n_2 \times d_2}$. This includes all numeric features.
- **Features generated from a kernel**: certain information about the row and column individuals may not be in the form of a numeric vector. For example, if the row individuals are vertices of a graph, their connection to each other is interesting information for the problem, but it is difficult to encode as real vectors. In this case, features can be generated through a transformation, or by defining a kernel function.

The link functions f_r and f_c

- **Linear**: $\mathbf{F}_r = f_r(\mathbf{X}_r) = \mathbf{X}_r \mathbf{B}_r$, and $\mathbf{F}_c = f_c(\mathbf{X}_c) = \mathbf{X}_c \mathbf{B}_c$. In this case, we need to estimate $\mathbf{B}_r$ and $\mathbf{B}_c$ to fit the model. With identity matrices as row and column features, this case is reduced to the traditional matrix factorization model with

  $$\mathbf{F}_r = \mathbf{B}_r, \quad \mathbf{F}_c = \mathbf{B}_c, \quad \mathbf{V}^* = \mathbf{F}_r \mathbf{F}_c^T = \mathbf{B}_r \mathbf{B}_c^T.$$

  When the features are generated from a kernel function, even a linear link function permits non-linear relationship between the features and the factor matrices.
- **General regression models**: when the relationship between the features and the variable of interest is not linear, any off-the-shelf regression methods can be plugged in to search for a non-linear link function.

The choice of the optimization problem (7.3)

Notice that with individual row and column features, linear link functions and complete observations, (7.3) becomes

$$\min_{\mathbf{F}_r, \mathbf{F}_c} \quad \|\mathbf{V} - (\mathbf{F}_r)_+ (\mathbf{F}_c)_+^T\|_F^2. \tag{7.4}$$

This is equivalent to the classical NMF problem,

$$\begin{aligned} \min_{\mathbf{F}_r, \mathbf{F}_c} \quad & \|\mathbf{V} - \mathbf{F}_r (\mathbf{F}_c)^T\|_F^2 \\ \text{s.t.} \quad & \mathbf{F}_r \geq \mathbf{0}, \quad \mathbf{F}_c \geq \mathbf{0}, \end{aligned} \tag{7.5}$$

in the sense that for any solution $(\mathbf{E}_r, \mathbf{E}_c)$ to (7.4), $((\mathbf{E}_r)_+, (\mathbf{E}_c)_+)$ is a solution to (7.5).

A more immediate generalization of (7.5) to include exogenous variables would be in the form

$$\begin{aligned} \min_{\mathbf{V}, f_r \in F_r^k, f_c \in F_c^k} \quad & \|\mathbf{V} - f_r(\mathbf{X}_r)(f_c(\mathbf{X}_c))^T\|_F^2 \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{V}) = \mathbf{b}, \quad \mathbf{V} \geq \mathbf{0}, \\ & f_r(\mathbf{X}_r) \geq \mathbf{0}, \quad f_c(\mathbf{X}_c) \geq \mathbf{0}. \end{aligned} \tag{7.6}$$

Solving (7.6) would involve identifying the subset of F_r and F_c that only produce nonnegative value on the row and column features, which could be difficult.

Table 7.1: *Classification of matrix factorization with side information by the mask, the link function, and the features included as side information, with some problems previously addressed in the literature.*

	Link function	Linear			Other regression methods
	Features	Identity	General numeric features	Kernel features	General numeric features
Mask	Identity	Matrix factorization	Reduced-regression rank [115–117]	Multiple kernel learning [118]	Nonparametric RRR [119]
	Matrix completion	Matrix completion [86]	IMC[112, 120–122]	GRMF [113, 114, 123]	
	Rank-one projections	[84, 85]			
	Temporal aggregates	[2]			
	General masks	Matrix recovery [78, 105]			

By using (7.3), we actually shifted the search space F_r and F_c to $(F_r)_+$ and $(F_c)_+$ which consists of composing all functions of F_r and F_c with the ramp function (thresholding at 0). In a word, (7.3) is equivalent to

$$\begin{aligned} \min_{\mathbf{V}, f_r \in (F_r)_+^k, f_c \in (F_c)_+^k} \quad & \|\mathbf{V} - f_r(\mathbf{X}_r)(f_c(\mathbf{X}_c))^T\|_F^2 \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{V}) = \mathbf{b}, \quad \mathbf{V} \geq \mathbf{0}. \end{aligned} \tag{7.7}$$

Problem (7.7) also helps us to reason on the identifiability of (7.3). In a sense, (7.3) is not well-identified: two distinct elements in $F_c^k \times F_r^k$ have the same evaluation value of the objective function, if they only differ on their negative parts. In fact, this does not affect the interpretation of the model, because these distinct elements correspond to the same element in $(F_r)_+^k \times (F_c)_+^k$. Since we are only going to use the positive parts of the function both in recovery and prediction, this becomes a parameterization choice which has no consequence on the applications.

As a comparison, we also propose an algorithm for the following optimization problem in Section 7.3:

$$\min_{f_r \in F_r^k, f_c \in F_c^k} \quad \|\mathbf{b} - \mathcal{A}((f_r(\mathbf{X}_r))_+ (f_c(\mathbf{X}_c))_+^T)\|_2^2, \tag{7.8}$$

which corresponds to 5.7 mentioned in Chapter 5. Instead of minimizing the low-rank approximation error for a matrix that matches the data as a linear matrix equation in (7.3), (7.8) minimizes the sampling error of an exactly low-rank matrix. Both have been studied in the literature. For example, objective functions similar to (7.3) have been considered in [78], and ones similar to (7.8) in [79]. We will see in both Sections 7.3 and 7.4 that (7.3) has a better performance than (7.8).

7.1.2 Prior works

Table 7.1 shows a taxonomy of matrix factorization models with side information, by the mask, the link function and the features used as side information.

There is an abundant literature that studies the general matrix factorization problem under various measurement operators, when no additional information is provided (see [78, 85, 86, 104] for various masks considered, and [90] for a recent global convergence result with RIP measurements). The NMF with general linear measurements is studied in various applications [2, 79, 83].

On the other hand, with complete observations, the multiple regression problem taking into account the low-rank structure of the (multi-dimensional) variable of interest is known as reduced-rank regression. This approach was first developed very early (see [115] for a review). Recent developments on rank selection [116], adaptive estimation procedures [124], using non-parametric link function [119], often draw the parallel between reduced-rank regression and the matrix completion problem. However, measurement operators other than complete observations or matrix completion are rarely considered in this community.

Building on theoretical boundaries on matrix completion, the authors of [112, 121, 122] showed that by providing side information (the matrix $\mathbf{X}$), the number of measurements needed for exact matrix completion can be reduced. Moreover, the number of measurements necessary for successful matrix completion can be quantified by measuring the quality of the side information [122].

Collaborative filtering with side information has received much attention from practitioners and academic researchers alike, for its huge impact in e-commerce and web applications [112–114, 120–122]. One of the first methods for including side information in collaborative filtering systems (matrix completion masks) was proposed by [123]. The authors generalized collaborative filtering into a operator estimation problem. This method allows more general feature spaces than a numerical matrix, by applying a kernel function to side information. [114] proposed choosing the kernel function based on the goal of the application. [118] applied the kernel-based collaborative filtering framework to electricity price forecasting. Their kernel choice is determined by multi-kernel learning methods.

To the best of our knowledge, matrix factorization (nonnegative or not) with side information, from general linear measurements has rarely been considered, nor is general non-linear functions other than with features obtained from kernels. This chapter aims at proposing a general approach which fills this gap.

7.2 Identifiability of nonnegative matrix factorization with side information

Matrix factorization is not a well-identified problem: for one pair of factors $(\mathbf{F}_r, \mathbf{F}_c)$, with $\mathbf{V}^* = \mathbf{F}_r\mathbf{F}_c^T$, any invertible matrix $\mathbf{R}$ produces another pair of factors, since $(\mathbf{F}_r\mathbf{R})(\mathbf{F}_c(\mathbf{R}^{-1})^T)^T$ is also equal to $\mathbf{V}^*$. In order to address this identifiability problem, one has to introduce extra constraints on the factors.

When the nonnegativity constraint is imposed on $\mathbf{F}_r$ and $\mathbf{F}_c$, however, it has been shown that sometimes the only invertible matrices that verify $\mathbf{F}_r\mathbf{R} \geq 0$ and $\mathbf{R}^{-1}\mathbf{F}_c \geq 0$ are the composition of a permutation matrix and a diagonal matrix with strictly positive diagonal elements. A nonnegative matrix factorization is said to be "identified" if the factors are unique up to permutation and scaling. The identifiability conditions for NMF are a hard problem,

because it turns out that NMF identifiability is equivalent to conditions that are computationally difficult to check. In this section, we review some known necessary and sufficient conditions for NMF identifiability in the literature, and develop a sufficient condition for NMF identifiability in the context of linear numerical features.

The reason for studying identifaibility of NMF with side information is two-fold. First of all, it is possible that using side information could restrict the solution space of the NMF problem, and makes the factorization problem more identifiable. Secondly, if we wish to find interpretation of the factors obtained in NMF, it is desirable if the factors are "unique".

In order to simplify our theoretical analysis, we focus on the complete observation case in this section (every entry in $\mathbf{V}^*$ is observed). Without loss of generality, we derive the sufficient condition for row features. That is, we will derive conditions on $\mathbf{V}^* \in \mathbb{R}_+^{n_1 \times n_2}$ and $\mathbf{X}_r \in \mathbb{R}^{n_1 \times d_1}$, so that the nonnegative matrix factorization $\mathbf{V}^* = \mathbf{X}_r \mathbf{B}_r \mathbf{F}_c^T$, with $\mathbf{X}_r \mathbf{B}_r \geq 0, \mathbf{F}_c \geq 0$, is unique. A generalization to column features can be easily obtained. In this section, we assume that in addition to be of nonnegative rank k, matrix $\mathbf{V}^*$ is also exactly of *rank* k.

7.2.1 Identifiability of NMF

The authors of [5] and [12] proposed two necessary and sufficient conditions for the factorization to be unique. Both conditions use the following geometric interpretation of NMF introduced by [5].

Since $\mathbf{V}^* = \mathbf{F}_r \mathbf{F}_c^T$, the columns of $\mathbf{V}^*$ are conical combinations of the columns in $\mathbf{F}_r$. Formally, $\text{cone}(\mathbf{F}_r)$, the conical hull of the columns of $\mathbf{F}_r$, is a polyhedral cone contained in the first orthant of $\mathbb{R}^{n_1}$. As $\mathbf{V}^*$ is of rank k, the rank of $\mathbf{F}_r$ is also k. This implies that the extreme rays (also called *generators*) of $\text{cone}(\mathbf{F}_r)$ are exactly the columns of $\mathbf{F}_r$, which are linearly independent. $\text{cone}(\mathbf{F}_r)$ is therefore

- a *simplicial* cone of k generators,
- contained in $\mathbb{R}_+^{n_1}$,
- containing all columns of $\mathbf{V}^*$.

Inversely, if we take any cone $\mathcal{F} \subseteq \mathbb{R}^{n_1}$ verifying these three conditions, and define a matrix $\mathbf{F}$ whose columns are the k generators of $\mathcal{F}$, there will be a nonnegative matrix $\mathbf{G}$, so that $\mathbf{V}^* = \mathbf{FG}$. The uniqueness of NMF is therefore equivalent to the uniqueness of simplicial cones of k generators contained in the first orthant of $\mathbb{R}^{n_1}$ and containing all columns $\mathbf{V}^*$.

In [12], an equivalent geometric interpretation in $\mathbb{R}^k$ is given in the following theorem:

Theorem 3. *[12] A k-dimensional NMF $\mathbf{V}^* = \mathbf{F}_r \mathbf{F}_c$ of a rank-k nonnegative matrix $\mathbf{V}^*$ is unique if and only if the nonnegative orthant $\mathbb{R}_+^k$ is the only simplicial cone $\mathcal{A}$ with k extreme rays satisfying*

$$\text{cone}(\mathbf{F}_r^T) \subseteq \mathcal{A} \subseteq \text{cone}(\mathbf{F}_c^\star).$$

Despite the apparent simplicity of the theorem, the necessary and sufficient conditions are very difficult to check. Based on the theorem above, several sufficient conditions have been proposed. The most widely used condition is called the separability condition. Before introducing this condition (in its least restrictive version presented by [12]), we need the following two definitions.

Definition 1 (Separability)**.** *Suppose $m \leq n$. A nonnegative matrix $\mathbf{M} \in \mathbb{R}_+^{m\times n}$ is said to be* separable *if there is a m-by-m permutation matrix Π which verifies*

$$\mathbf{M} = \Pi \begin{pmatrix} \mathbf{D}_n \\ \mathbf{M}_0 \end{pmatrix},$$

where $\mathbf{D}_n$ is a n-by-n diagonal matrix with only strictly positive coefficients on the diagonal and zeros everywhere else, and the $(m-n)$-by-n matrix $\mathbf{M}_0$ is a collection of the other $m-n$ rows of $\mathbf{M}$.

Definition 2 (Strongly Boundary Closeness)**.** *A nonnegative matrix $\mathbf{M} \in \mathbb{R}_+^{m\times n}$ is said to be* strongly boundary close *if the following conditions are satisfied.*

1. $\mathbf{M}$ *is* boundary close*: for all $i, j \in \{1, ..., n\}, i \neq j$, there is a row $\mathbf{m}$ in $\mathbf{M}$ which satisfies $m_i = 0, m_j > 0$;*
2. *There is a permutation of $\{1, ..., n\}$ such that for all $i \in \{1, ..., n-1\}$, there are $n-i$ rows $\mathbf{m}^1, ..., \mathbf{m}^{n-i}$ in $\mathbf{M}$ which satisfy*
   (a) $m_i^j = 0, \sum_{s=i+1}^n m_i^j > 0$ *for all $j \in \{1, ..., n-i\}$;*
   (b) *the square matrix $(m_s^j)_{1\leq j\leq n-i, i+1\leq s\leq n}$ is of full rank $(n-i)$.*

Strongly boundary closeness demands, *modulo* a permutation in $\{1, ..., n\}$, that for each $1 \leq i \leq n-1$, there are $n-i$ rows $\mathbf{m}^1, ..., \mathbf{m}^{n-i}$ of $\mathbf{M}$ that have the following form,

$$\begin{pmatrix} \mathbf{m}^1 \\ \vdots \\ \mathbf{m}^{n-i} \end{pmatrix}^T = \begin{pmatrix} \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \\ m_{i+1}^1 & \cdots & m_{i+1}^{n-i} \\ \vdots & \ddots & \vdots \\ m_n^1 & \cdots & m_n^{n-i} \end{pmatrix} \begin{array}{l} (i-1) \text{ first rows} \\ i\text{-th row is all zero} \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} (n-i)\times(n-i) \text{ full rank matrix} \end{array} \tag{7.9}$$

These row vectors, $\mathbf{m}^1, ..., \mathbf{m}^{n-i}$, all have 0 on the i-th element, and its lower square matrix of is of full rank. There are therefore enough linearly independent points on each $n-1$-dimensional facet $\mathbb{R}_+^n$, which shows that cone$(\mathbf{M}^T)$ is somewhat maximal in $\mathbb{R}_+^n$.

The following was proved in [12]:

Theorem 4. *[12] If $\mathbf{F}_r$ is strongly boundary close, then the only simplicial cone with k generators in $\mathbb{R}_+^k$ containing cone$(\mathbf{F}_r^T)$ is $\mathbb{R}_+^k$. Moreover, if $\mathbf{F}_c$ is separable, then $\mathbf{V}^* = \mathbf{F}_r\mathbf{F}_c^T$ is the unique NMF of $\mathbf{V}^*$ up to permutation and scaling.*

7.2.2 Identifiability with side information

The NMF with linear row features, $\mathbf{V}^* = \mathbf{X}_r\mathbf{B}_r\mathbf{F}_c^T$, is said to be *unique*, if for all matrix pairs $(\tilde{\mathbf{B}}_r, \tilde{\mathbf{F}}_c) \in \mathbb{R}^{d_1\times k} \times \mathbb{R}^{n_2\times k}$ that verifies

$$\mathbf{X}_r\tilde{\mathbf{B}}_r \geq 0, \quad \tilde{\mathbf{F}}_c \geq 0, \quad \mathbf{V}^* = \mathbf{X}_r\tilde{\mathbf{B}}_r\tilde{\mathbf{F}}_c,$$

we have $\tilde{\mathbf{B}} = \mathbf{B}_r$, $\tilde{\mathbf{F}}_c = \mathbf{F}_c$ up to permutation of columns and scaling.

For a given full-rank matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times d_1}$, consider the following two sets of matrices:

$$E = \{\mathbf{M} \in \mathbb{R}_+^{n_1 \times k} | \text{The columns of } \mathbf{M} \text{ are strongly boundary close}\};$$
$$F(\mathbf{X}) = \{\mathbf{M} \in \mathbb{R}_+^{n_1 \times k} | \text{rank}(\mathbf{M}) = k, \text{span}(\mathbf{M}) \in \text{span}(\mathbf{X})\}.$$

Theorem 5. *If $E \bigcap F(\mathbf{X}_r) \neq \varnothing$, and $\mathbf{B}_r \in (\mathbf{X}_r^T \mathbf{X}_r)^{-1} \mathbf{X}_r^T (E \bigcap F(\mathbf{X}_r))$, and $\mathbf{F}_c$ is separable, then the factorization $\mathbf{V}^* = \mathbf{X}_r \mathbf{B}_r \mathbf{F}_c^T$ is unique.*

Proof. Notice that for $\mathbf{B}_r \in (\mathbf{X}_r^T \mathbf{X}_r)^{-1} \mathbf{X}_r^T (E \bigcap F(\mathbf{X}_r))$, the nonnegative matrix $\mathbf{X}_r \mathbf{B}_r$ is strongly boundary close. The factorization $(\mathbf{X}_r \mathbf{B}_r, \mathbf{F}_c)$ is therefore unique. The model identifiability follows immediately, since $\mathbf{X}_r$ is of full rank. □

Example of $\mathbf{X}_r$ that verifies $E \bigcap F(\mathbf{X}_r) \neq \varnothing$

For this theorem to have practical consequences, one needs to find appropriate row features so that $E \bigcap F(\mathbf{X}_r) \neq \varnothing$.

Here we provide a family of matrices $\mathbf{X}_r$ so that $E \bigcap F(\mathbf{X}_r) \neq \varnothing$.

With a fixed $k \geq 2$, suppose that $\mathbf{X}_r$ has $k(k-1)/2$ columns, and at least $k(k-1)/2$ rows, with the first $k(k-1)/2 + 1$ rows defined as the following:

- the first row and column have 0 on the first entry and positive entries elsewhere;
- for $2 \leq i \leq k$, $\mathbf{X}_r$ has strictly positive entries on the first $((i-1)(i-2)/2 + 1)$ columns, from Row $(i-1)(i-2)/2 + 3$ to Row $(i-1)(i-2)/2 + 1 + i$, and zero entries everywhere else. These $(k-1)$ rows are linearly independent.

Then we have $E \bigcap F(\mathbf{X}_r) \neq \varnothing$, because the following $k(k-1)/2$-by-k matrix $\mathbf{B}_r$ is in this set:

- for $1 \leq i \leq k$, $\mathbf{B}_K^*$ has i consecutive strictly positive entries on the i-th column, between Row $i(i-1)/2 + 1$ and Row $i(i-1)/2 + i$.

The following matrices instantiate the case of $k = 4$:

$$\mathbf{B}_r = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{X}_r = \begin{pmatrix} 0 & 5 & 14 & 7 & 9 & 15 & 13 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 & 0 \\ 12 & 4 & 0 & 0 & 0 & 0 & 0 \\ 10 & 7 & 10 & 7 & 0 & 0 & 0 \\ 13 & 10 & 12 & 9 & 0 & 0 & 0 \\ 12 & 10 & 16 & 8 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}, \mathbf{F}_r = \begin{pmatrix} 0 & 5 & 21 & 37 \\ 10 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 \\ 12 & 4 & 0 & 0 \\ 10 & 7 & 17 & 0 \\ 13 & 10 & 21 & 0 \\ 12 & 10 & 24 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}.$$

If $E \bigcap F(\mathbf{X}) \neq \varnothing$, for any invertible matrix $\mathbf{R} \in \mathbb{R}^{K \times K}$, $E \bigcap F(\mathbf{XR}) \neq \varnothing$.

As a conclusion to this section, we notice that since the sufficient condition proposed by Theorem 5 is based on a known uniqueness condition of classical NMF, this does not make the problem more identifiable. Rather, we derived a sufficient condition on the feature matrix and the coefficient matrix ($\mathbf{X}_r$ and $\mathbf{B_r}$) for the NMF with side information to be unique. This is a generalization of identifiability analysis of NMF to the case with side information.

7.3 HALSX algorithm

In this section, we propose HALSX, or *Hierarchical Alternating Least Squares with eXogeneous variables*, a general algorithm to estimate the nonnegative matrix factorization problem with side information, from linear measurement, by solving (7.3). It is an extension to a popular NMF algorithm: Hierarchical Alternating Least Squares (HALS) (see [18, 19]).

Before discussing HALSX, we will first present a result on the local convergence of Gauss-Seidel algorithms. This result guarantees that any legitimate limiting points generated by HALSX are stationary points of (7.3).

While presenting specific methods to estimate link functions, we will only discuss row features, as a generalization to column features is immediate.

7.3.1 Relaxation of convexity assumption for the convergence of Gauss-Seidel algorithm

To show that all legitimate limiting points of HALSX are stationary points, we first extend a classical result concerning block nonlinear Gauss-Seidel algorithm [110, Proposition 4].

Consider the minimization problem,

$$\begin{aligned} \min \quad & g(x) \\ \text{s.t.} \quad & x \in X = X_1 \times X_2 \times ... \times X_m \subseteq \mathbb{R}^n, \end{aligned} \tag{7.10}$$

where g is a continuously differentiable real-valued function, and the feasible set X is the Cartesian product of closed, nonempty and convex subsets $X_i \subset \mathbb{R}^{n_i}$, for $1 \leq i \leq m$, with $\sum_i n_i = n$. Suppose that the global minimum is reached at a point in X. The m-block Gauss-Seidel algorithm is defined as Algorithm 3.

Algorithm 3 Gauss-Seidel algorithm

```
Initialize x^0 ∈ X, t = 0
while Stopping criterion is not satisfied do
  for i = 1, 2, ..., m do
    Calculate x_i^{t+1} = arg min_{y_i ∈ X_i} g(x_1^{t+1}, ..., y_i, ..., x_m^t)
  end for
  Set x^{t+1} = (x_1^{t+1}, ..., x_m^{t+1})
  t = t + 1
end while
```

Define formally the notion of component-wise quasi-convexity.

Definition 3. *Let $i \in \{1, 2, ..., m\}$. The function g is quasi-convex with respect to the i-th*

component on X if for every $x \in X$ and $y_i \in X_i$, we have

$$\begin{aligned} & g(x_1, x_2, ...tx_i + (1-t)y_i, ..., x_m) \\ \leq & \max\{g(x), g(x_1, x_2, ..., y_i, ...x_m)\} \end{aligned}$$

for all $t \in [0,1]$. g is said to be strictly quasi-convex with respect to the i-th component, if with the additional assumption that $y_i \neq x_i$, we have

$$\begin{aligned} & g(x_1, x_2, ...tx_i + (1-t)y_i, ..., x_m) \\ < & \max\{g(x), g(x_1, x_2, ..., y_i, ...x_m)\} \end{aligned}$$

for all $t \in]0,1[$.

It has been shown that if g is strictly quasi-convex with respect to the first $m-2$ blocks of components on X, then a limiting point produced by a Gauss-Seidel algorithm is a critical point [110].

This result is not directly applicable for the HALS algorithm. Typically, if $\mathbf{f}_{c,i}$, the $i-th$ column of $\mathbf{F}_c$, is identically zero, the loss function is completely flat respect to $\mathbf{f}_{c,i}$, the i-th column of $\mathbf{F}_r$. Therefore the loss function is not strictly quasi-convex. In order to avoid this scenario, [18] suggests thresholding at a small positive number ϵ instead of at 0, when updating each column of the factor matrices.

In fact the convexity assumption of [110] can be slightly relaxed to directly apply to HALS, as demonstrated by the following proposition.

Theorem 6. *Suppose that the function g is quasi-convex with respect to x_i on X, for $i = 1, ..., m-2$. Suppose that some limit points $\bar{x}$ of the sequence $\{x^t\}_{(t \in \mathbb{N})}$ verify that g is strictly quasi-convex with respect to x_i on the product set $\{\bar{x}_1\} \times \{\bar{x}_2\} \times ... \times X_i \times, ... \times \{\bar{x}_m\}$, for $i = 1, ..., m-2$. Then every such limiting point is a critical point of Problem (3).*

Compared to the result of [110], this shows that the strict convexity with respect to one block does not have to hold universally for feasible regions of other blocks. It only needs to hold at the limiting point.

This theorem can be established following the proof of Proposition 5 of [110], using the following lemma.

Lemma 1. *Suppose that the function g is quasi-convex with respect to x_i on X, for some $i \in \{1, ..., m\}$. Suppose that some limit points $\bar{y}$ of $\{y^t\}$ verify that g is strictly quasi-convex with respect to x_i on $\{\bar{y}_1\} \times \{\bar{y}_2\} \times ... \times X_i \times, ... \times \{\bar{y}_m\}$. Let $\{v^t\}$ be a sequence of vectors defined as follows:*

$$v_j^t = \begin{cases} y_j^t & \text{if } j \neq i, \\ \arg\min_{z_i \in X_i} g(y_1^t, ..., z_i, ..., y_m^t) & \text{if } j = i. \end{cases}$$

Then, if $\lim_{t \to +\infty} g(y^t) - g(v^t) = 0$, we have $\lim_{t \to +\infty} ||v_i^t - y_i^t|| = 0$. That is $\lim_{t \to +\infty} ||v^t - y^t|| = 0$.

Proof. (The proof of the lemma is based on [125].)

Suppose on the contrary that $||v_i^t - y_i^t||$ does not converge to 0. Define $\tau_k = ||v_i^t - y_i^t||$. Restricting to a subsequence, we can obtain that $\tau_k \geq \tau_0 > 0$. Define $s^t = \frac{v^t - y^t}{\tau_k}$. Notice that $\{s^t\}$ is of unit norm, and $v^t = y^t + \tau_k s^t$. Since $\{s^t\}$ is on the unit sphere, it has a converging subsequence. By restricting to a subsequence again, we could suppose that $\{s^t\}$ converges to $\bar{s}$.

For all $\epsilon \in [0,1]$, we have $0 \leq \epsilon\tau_0 \leq \tau_k$, which implies $y^t + \epsilon\tau_0 s^t \in X$ is on the segment $[y^t, v^t]$. This segment has strictly positive dimension in the subspace corresponding to X_i.

By the definition of $\{v^t\}$, $g(v^t) \leq g(y_1^t, ..., z_i, ..., y_m^t)$, for all t, and for all $z_i \in X_i$. In particular,

$$g(v^t) \leq g(y^t + \epsilon\tau_0 s^t).$$

By quasi-convexity of g on X,

$$g(y^t + \epsilon\tau_0 s^t) \leq \max\{g(y^t), g(v^t)\} = g(y^t).$$

Taking the limit when t converges to $+\infty$ on both equalities, we obtain

$$\begin{aligned} g(\bar{y}) &= \lim_{t\to+\infty} g(v^t) \leq \lim_{t\to+\infty} g(y^t + \epsilon\tau_0 s^t) \\ &= g(\bar{y} + \epsilon\tau_0 \bar{s}) \leq \lim_{t\to+\infty} g(y^t) = g(\bar{y}). \end{aligned}$$

In other words, $g(\bar{y} + \epsilon\tau_0\bar{s}) = g(\bar{y})$, $\forall \epsilon \in [0,1]$, which contradicts the strict quasi-convexity of g on $\{\bar{y}_1\} \times \{\bar{y}_2\} \times ... \times X_i \times, ... \times \{\bar{y}_m\}$. □

7.3.2 HALSX algorithm

To solve (7.3), we propose HALSX (Algorithm 4). When complete observations are available, the feature matrices are identity matrices, and when only linear functions are allowed as link functions, Algorithm 4 is equivalent to HALS [19].

From Theorem 6, one deduces that every full-rank limiting point produced by the popular HALS algorithm is a critical point.

In this algorithm, at each elementary update step, we first look for a link function which minimizes the quadratic error, without concerning ourselves with its nonnegativity. The obtained evaluation of the minimizer function is then thresholded at $\mathbf{0}$ to update the factors.

To obtain that the limiting points of HALSX are stationary points, we need to ensure that for some functional spaces F_r and F_c, such an update solves a corresponding subproblem of (7.3). To do this, we will use the following proposition:

Proposition 1. *Suppose that $\mathbf{R} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{f}_c \in \mathbb{R}_+^{n_2}$ are not identically equal to zero, and $g : \mathbb{R}^d \to \mathbb{R}^{n_1}$, with $d \geq n_1$, is a convex differentiable function. Suppose*

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \|\mathbf{R} - g(\boldsymbol{\theta})(\mathbf{f}_c)^T\|_F^2.$$

If $\nabla g_{\boldsymbol{\theta}^*}$, the Jacobian matrix of g at $\boldsymbol{\theta}^*$, is of rank n_1, then $\boldsymbol{\theta}^*$ is also a solution to

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \|\mathbf{R} - (g(\boldsymbol{\theta}))_+(\mathbf{f}_c)^T\|_F^2. \tag{7.11}$$

Proof. Take $\mathbf{R} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{f}_c \in \mathbb{R}_+^{n_2}$ not identically equal to zero. We will note by L the loss function, so that $L(\mathbf{f}) = \|\mathbf{R} - (\mathbf{f})_+(\mathbf{f}_c)^T\|_F^2$ for all $\mathbf{f} \in \mathbb{R}^{n_1}$. The function L is convex.

Problem (7.11), which can be rewritten as

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} L(g(\boldsymbol{\theta})),$$

Algorithm 4 Hierarchical Alternating Least Squares with eXogeneous variables for NMF (HALSX)

Require: Measurement operator $\mathcal{A}$, measurements $\mathbf{b}$, features $\mathbf{X}_r$ and $\mathbf{X}_c$, functional spaces F_r and F_c in which to search the link functions, and $1 \leq k \leq \min\{n_1, n_2\}$.
  Initialize $\mathbf{F}_r^0, \mathbf{F}_c^0 \geq 0, t = 0$
  while Stopping criterion is not satisfied **do**
    $\mathbf{V}^t = \arg\min_{\mathbf{V}|\mathcal{A}(\mathbf{V})=\mathbf{b},\mathbf{V}\geq 0} \|\mathbf{V} - \mathbf{F}_r^t(\mathbf{F}_c^t)^T\|_F^2$
    $\mathbf{R}^t = \mathbf{V}^t - \mathbf{F}_r^t(\mathbf{F}_c^t)^T$
    for $i = 1, 2, ..., k$ **do**
      $\mathbf{R}_i^t = \mathbf{R}^t + \mathbf{f}_{r,i}^t(\mathbf{f}_{c,i}^t)^T$
      Calculate $f_{r,i}^{t+1} = \arg\min_{f\in F_r} \|\mathbf{R}_i^t - f(\mathbf{X}_r)(\mathbf{f}_{c,i}^t)^T\|_F^2$
      $\mathbf{f}_{r,i}^{t+1} = \max(0, f_{r,i}^{t+1}(\mathbf{X}_r))$
      $\mathbf{R}^t = \mathbf{R}_i^t - \mathbf{f}_{r,i}^{t+1}(\mathbf{f}_{c,i}^t)^T$
    end for
    for $i = 1, 2, ..., k$ **do**
      $\mathbf{R}_i^t = \mathbf{R}^t + \mathbf{f}_{r,i}^{t+1}(\mathbf{f}_{c,i}^t)^T$
      Calculate $f_{c,i}^{t+1} = \arg\min_{f\in F_c} \|\mathbf{R}_i^t - \mathbf{f}_{r,i}^{t+1} f(\mathbf{X}_c)^T\|_F^2$
      $\mathbf{f}_{c,i}^{t+1} = \max(0, f_{c,i}^{t+1}(\mathbf{X}_c))$
      $\mathbf{R}^t = \mathbf{R}_i^t - \mathbf{f}_{r,i}^{t+1}(\mathbf{f}_{c,i}^{t+1})^T$
    end for
    $t = t + 1$
  end while
output $\mathbf{V}^t = \arg\min_{\mathbf{V}|\mathcal{A}(\mathbf{V})=\mathbf{b},\mathbf{V}\geq 0} \|\mathbf{V} - \mathbf{F}_r^t(\mathbf{F}_c^t)^T\|_F^2$,
  $\mathbf{F}_r^t \in \mathbb{R}_+^{n_1\times k}, f_{r,1}^t, ..., f_{r,k}^t \in F_r$,
  $\mathbf{F}_c^t \in \mathbb{R}_+^{n_2\times k}, f_{c,1}^t, ..., f_{c,k}^t \in F_c$.

is also convex. The subgradient of the composition function $L \circ g$ at $\boldsymbol{\theta} \in \mathbb{R}^d$ is simply obtained by multiplying $\nabla g_{\boldsymbol{\theta}}$, the Jacobian matrix of g at $\boldsymbol{\theta}$, to each element of $\partial L_{g(\boldsymbol{\theta})}$, or $\partial L_{g(\boldsymbol{\theta})} \equiv \nabla g_{\boldsymbol{\theta}} \partial L_{g(\boldsymbol{\theta})} = \{\nabla g_{\boldsymbol{\theta}} \mathbf{y} | \mathbf{y} \in \partial L_{g(\boldsymbol{\theta})}\}$. Therefore $\forall \boldsymbol{\theta} \in \mathbb{R}^d$, $\boldsymbol{\theta}$ is a minimizer of (7.11), if and only if $\mathbf{0} \in \nabla g_{\boldsymbol{\theta}} \partial L_{g(\boldsymbol{\theta})}$.

Since

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \|\mathbf{R} - g(\boldsymbol{\theta})(\mathbf{f}_c)^T\|_F^2,$$

is a minimizer of a smooth convex problem,

$$\frac{\partial}{\partial \boldsymbol{\theta}} \|\mathbf{R} - g(\boldsymbol{\theta})(\mathbf{f}_c)^T\|_F^2 (\boldsymbol{\theta}^*) = \nabla g_{\boldsymbol{\theta}^*} (\mathbf{R} - g(\boldsymbol{\theta}^*)(\mathbf{f}_c)^T)\mathbf{f}_c = \mathbf{0}.$$

This means $(\mathbf{R} - g(\boldsymbol{\theta}^*)(\mathbf{f}_c)^T)\mathbf{f}_c = \mathbf{0}$, because $\nabla g_{\boldsymbol{\theta}^*}$ is of full rank. Consequently

$$g(\boldsymbol{\theta}^*) = \frac{1}{\|\mathbf{f}_c\|_2^2} \mathbf{R}\mathbf{f}_c.$$

It has been shown in NMF literature (for example [18, Theorem 2]) that,

$$(g(\boldsymbol{\theta}^*))_+ = \arg\min_{\mathbf{f} \in \mathbb{R}_+^{n_1}} \|\mathbf{R} - \mathbf{f}(\mathbf{f}_c)^T\|_F^2.$$

This is equivalent to

$$g(\boldsymbol{\theta}^*) \in \arg\min_{\mathbf{f} \in \mathbb{R}^{n_1}} L(\mathbf{f}),$$

or

$$\mathbf{0} \in \partial L_{g(\boldsymbol{\theta}^*)}.$$

We therefore conclude with $\mathbf{0} \in \nabla g_{\boldsymbol{\theta}^*} \partial L_{g(\boldsymbol{\theta}^*)}$. □

In many regression methods, even when a non-linear transformation is applied to the data, the regression function is linear in its parameters. A non-exhaustive list of methods include linear regression ($g(\boldsymbol{\theta}) = \mathbf{X}_r \boldsymbol{\theta}$), spline regression ($g(\boldsymbol{\theta}) = \phi(\mathbf{X}_r)\boldsymbol{\theta}$), or support vector regression (SVR) ($g(\boldsymbol{\theta}) = K(\mathbf{X}_r, \mathbf{X}_r)\boldsymbol{\theta}$). In this case, g has a constant Jacobian matrix. In the case of linear and spline regression, the Jacobian matrix is of rank n_1 if there are no less features than examples. For SVR, this is true for any positive definite kernels. This allows us to apply the previous lemma to each column update step of Algorithm 4. By calculating $f_{r,i}^{t+1} = \arg\min_{f \in F_r} \|\mathbf{R}^t - f(\mathbf{X}_r)(\mathbf{f}_{c,i}^t)^T\|_F^2$ at Step t for Column i in $\mathbf{F}_r$, we actually have

$$f_{r,i}^{t+1} = \arg\min_{f \in F_r} \|\mathbf{R}^t - (f(\mathbf{X}_r))_+ (\mathbf{f}_{c,i}^t)^T\|_F^2.$$

This shows that at each iteration, we solve the subproblems of (7.3).

In these cases, by rewriting the functional space F_r and F_c in a parametric form, the search space is actually $\mathbb{R}^{r_1}$ and $\mathbb{R}^{r_2}$, for some r_1 and r_2.

Theorem 7. *If $n_1 \leq r_1$, $n_2 \leq r_2$, every full-rank factorization produced by HALSX (Algorithm 4) is a critical point of Problem (7.3).*

Remark 1. *In order to obtain a similar result for Problem (7.6), one would need to ensure the obtained functions have non-negative values on the features. This could be done by alternating projection.*

7.3.3 Designs and HALSX

At each iteration of Algorithm 4, we need to project the working matrix $\mathbf{F}_r^t(\mathbf{F}_c^t)^T$ into the convex polytope defined by the measurements and nonnegativity:

$$\mathbf{V}^t = \arg \min_{\mathbf{V}|\mathcal{A}(\mathbf{V})=\mathbf{b},\mathbf{V}\geq 0} \|\mathbf{V} - \mathbf{F}_r^t(\mathbf{F}_c^t)^T\|_F^2. \tag{7.12}$$

In general, the polytope projection can be obtained by alternating projection. Namely, we can alternate between:

- $\mathbf{V} = \mathbf{V} + \mathcal{A}^\dagger(\mathbf{b} - \mathcal{A}(\mathbf{V}))$;
- $v_{i,j} = \max(0, v_{i,j})$,

where $\mathcal{A}^\dagger$ is the right pseudo-inverse of $\mathcal{A}$, viewed as an N-by-n_1n_2 matrix.

For some measurement operators, there are efficient ways to solve (7.12).

- Matrix completion mask:
  $v_{i,j} = \begin{cases} \alpha_l, & \text{if } \exists 1 \leq l \leq N, \mathbf{A}_l = \mathbf{e}_i\mathbf{e}_j^T; \\ \max(0, v_{i,j}), & \text{if not.} \end{cases}$
- Temporal aggregate mask: simplex projection (see [2] for details).

7.3.4 Linear HALSX

In this section, we consider HALSX with numeric row features and linear row link functions. That is, given $\mathbf{X}_r$ and $\boldsymbol{\alpha} = \mathcal{A}(\mathbf{V}^*)$, we need to solve

$$\begin{aligned} \min_{\mathbf{V}\in\mathbb{R}^{n_1\times n_2},\mathbf{B}_r\in\mathbb{R}^{d_1\times k},\mathbf{F}_c\in\mathbb{R}^{n_2\times k}} \quad & \|\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T\|_F^2 \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{V}) = \mathbf{b}, \quad \mathbf{V} \geq \mathbf{0}, \end{aligned} \tag{7.13}$$

Following Algorithm 4, we need to update the columns of $\mathbf{B}_r$ at each iteration. At the t-th step, for $1 \leq i \leq k$, we solve the subproblem

$$\arg\min_{\mathbf{b}_{r,i}} \|\mathbf{R}_i^t - \mathbf{X}_r\mathbf{b}_{r,i}(\mathbf{f}_{c,i}^t)^T\|_F^2,$$

where $\mathbf{R}_i^t = \mathbf{V}^t - \sum_{j=1,j\neq i}^k \mathbf{X}_r\mathbf{b}_{r,l}\mathbf{f}_{c,l}^T$. This minimization problem has a closed-form solution:

$$\mathbf{b}_{r,i}^{t+1} = \frac{1}{\|\mathbf{f}_{c,i}^t\|_2^2}(\mathbf{X}_r^T\mathbf{X}_r)^{-1}\mathbf{X}_r^T\mathbf{R}_i^t\mathbf{f}_{c,i}^t.$$

In order to accelerate the numerical algorithm, a QR decomposition of $\mathbf{X}_r = \mathbf{QR}$ is done before the iterations, where $\mathbf{Q}$ is an orthogonal matrix, and $\mathbf{R}$ is a square upper triangular matrix. When $\mathbf{X}_r$ is of full rank, $\mathbf{X}_r^T\mathbf{X}_r$ is invertible. We compute one time $(\mathbf{X}_r^T\mathbf{X}_r)^{-1}\mathbf{X}_r = \mathbf{R}^{-1}\mathbf{Q}^T$ before the iterations, and use the result at each iteration.

Stopping criterion

As in classical NMF algorithms, we will use the Karush–Kuhn–Tucker conditions (KKT) to provide a stopping criterion. The KKT conditions of (7.13) are,

$$\begin{aligned}
&\mathbf{V} \geq \mathbf{0}, \quad \mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T \geq \mathbf{0}, \quad \mathcal{A}(\mathbf{V}) = \mathbf{b}, \\
&\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T \circ \mathbf{V} = \mathbf{0}, \\
&\nabla_{\mathbf{F}_c}\|\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T\|_F^2 \ni \mathbf{0}, \\
&\nabla_{\mathbf{B}_r}\|\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T\|_F^2 \ni \mathbf{0},
\end{aligned}$$

where $\mathbf{A} \circ \mathbf{B}$ is the entry-wise product (Hadamard product) for $\mathbf{A}, \mathbf{B}$ of the same dimension, and $\nabla_x f(x_0)$ is the subgradient of the function f at point x_0, with respect to the variable x. Note that $\mathbf{V} \geq \mathbf{0}$ and $\mathcal{A}(\mathbf{V}) = \mathbf{b}$ are always satisfied at the end of an iteration.

As $(\mathbf{X}_r\mathbf{B}_r)_+^T(\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T) \circ \mathbb{1}_{(\mathbf{F}_c)>\mathbf{0}}$ and $\mathbf{X}_r^T((\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T) \circ \mathbb{1}_{\mathbf{X}_r\mathbf{B}_r>\mathbf{0}})$ are respectively in the subgradient with respect to $\mathbf{F}_c$ and $\mathbf{B}_r$, we will stop the algorithm when the norm of following vector

$$\begin{aligned}
&[vect((\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T)_-)^T, \\
&vect(\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T \circ \mathbf{V})^T, \\
&vect((\mathbf{X}_r\mathbf{B}_r)_+^T(\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T) \circ \mathbb{1}_{(\mathbf{F}_c)>\mathbf{0}})^T, \\
&vect(\mathbf{X}_r^T((\mathbf{V} - (\mathbf{X}_r\mathbf{B}_r)_+(\mathbf{F}_c)_+^T) \circ \mathbb{1}_{\mathbf{X}_r\mathbf{B}_r>\mathbf{0}}))^T],
\end{aligned}$$

is smaller than ϵ times its initial value, with a small ϵ. For the algorithms presented in the next sections, this stopping criterion is generalized quite easily.

7.3.5 HALSX with smoothing splines

The computation considered above can estimate an NMF with linear features fairly efficiently. However, in real applications, linear link functions are too restrictive. In the following, we will estimate non-linear link functions that are Generalized Additive Models (GAM, [126]).

A Generalized Additive Model is a generalization to Generalized Linear Model (GLM) which includes additive non-linear components. Consider n observations $\mathbf{x}_i, y_i$, for $1 \leq i \leq n$, where $\mathbf{x}_i$ is the vector of features, and y_i is an observation of a random variable Y_i. Suppose that $Y_i = \mu_i + \epsilon_i$, where ϵ_i are independent identically distributed zero-mean Gaussian variables, and $\mu_i = \mathbf{E}(Y_i)$ has the following relationship to the features:

$$g(\mu_i) = \mathbf{x}_i^T\boldsymbol{\theta} + h_1(x_{i,1}) + h_2(x_{i,2}) + h_3(x_{i,3}, x_{i,4}) + ...$$

where $\boldsymbol{\theta}$ is the vector of parametric model components, g is a known, monotonic, twice-differentiable function, h_1, h_2, h_3, ..., are the non-linear functions to be estimated.

We note by $\mathbf{X}$ the matrix grouping the features of all observations. We use penalized regression spline to fit the GAMs. For $j = 1, 2, 3, ...$, define a spline basis $\mathbf{a}^j = (a_1^j, a_2^j, ...)$ in which h_j, the j-th component of the GAM, is to be estimated. Practically, we search for h_j is the L-dimensional vector space

$$H(\mathbf{a}^j, L_j) = \{\sum_{l=1}^{L_j} \beta_l^j a_l^j | \boldsymbol{\beta}^j = (\beta_1^j, ...\beta_{L_j}^j) \in \mathbb{R}^{L_j}\}.$$

Noting by $\mathbf{X}^j = \{a_l^j(\mathbf{x}_i)\}_{i,l}$ the design matrix, for $h_j = \sum_{l=1}^{L_j} \beta_l^j a_l^j \in H(\mathbf{a}^j, L_j)$, an element of the functional space, we have

$$h_j(\mathbf{X}) = \mathbf{X}^j \boldsymbol{\beta}^j.$$

The whole model of g, can then be represented linearly:

$$\begin{aligned} g(\boldsymbol{\mu}) &= \mathbf{X}\theta + (\mathbf{X}^1, \mathbf{X}^2, ...) \begin{pmatrix} \boldsymbol{\beta}^1 \\ \boldsymbol{\beta}^2 \\ \vdots \end{pmatrix} \\ &= \mathbf{X}\boldsymbol{\theta} + \sum_j \mathbf{X}^j \boldsymbol{\beta}^j. \end{aligned}$$

The dimension of $H(\mathbf{a}^j, L_j)$, L_j, controls the smoothness of the functions to be estimated. As little information is available on the degree of smoothness of the functions, we use a rather large L_j, and add a penalty on the wiggliness, $\int (h_j'')^2 dx$, as in [126]. The least squares estimator of this model is therefore

$$\arg \min_{\boldsymbol{\theta}, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2, ...} \|g(\boldsymbol{\mu}) - \mathbf{X}\boldsymbol{\theta} - \sum_j \mathbf{X}^j \boldsymbol{\beta}^j\|^2 + \sum_j \lambda^j (\boldsymbol{\beta}^j)^T \mathbf{S}^j \boldsymbol{\beta}^j,$$

where λ_j is the penalization parameter of the j-th non-linear component, and $\mathbf{S}^j$ is a positive definite matrix depending on $\mathbf{X}$ and $\mathbf{a}^j$. The penalization parameter, λ^j, is chosen by a generalized cross validation criterion.

HALSX-GAM

At each iteration of the algorithm, for $i = 1, ..., k$, we re-estimate the link function $f_{r,i}$ of the i-th column of $\mathbf{F}_r$ as a GAM.

The subproblem for i is the following

$$\begin{aligned} \arg \min_{\boldsymbol{\theta}, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2, ...} &\|\mathbf{R}_i^t - (\mathbf{X}_r \boldsymbol{\theta} + \sum_{j=1} \mathbf{X}^j \boldsymbol{\beta}^j)(\mathbf{f}_{c,i}^t)^T\|_F^2 + \\ &\sum_j \lambda^j (\boldsymbol{\beta}^j)^T \mathbf{S}^j \boldsymbol{\beta}^j. \end{aligned} \tag{7.14}$$

With fixed penalization parameters λ_1, λ_2, ..., the optimization above can be solved by

$$\begin{aligned} &\begin{pmatrix} \boldsymbol{\theta} \\ \boldsymbol{\beta}^1 \\ \boldsymbol{\beta}^2 \\ \vdots \end{pmatrix}^{t+1} = \frac{1}{\|f_{c,i}^t\|^2} \times \\ &\begin{pmatrix} \mathbf{X}_r^T \mathbf{X}_r & \mathbf{X}_r^T \mathbf{X}^1 & \mathbf{X}_r^T \mathbf{X}^2 & \cdots \\ (\mathbf{X}^1)^T \mathbf{X}_r & (\mathbf{X}^1)^T \mathbf{X}^1 + \frac{\lambda^1}{\|f_{c,i}^t\|^2} \mathbf{S}^j & (\mathbf{X}^1)^T \mathbf{X}^2 & \cdots \\ \vdots & \vdots & \ddots & \cdots \end{pmatrix}^{-1} \times \\ &\begin{pmatrix} \mathbf{X}_r^T \\ (\mathbf{X}^1)^T \\ (\mathbf{X}^2)^T \\ \vdots \end{pmatrix} \mathbf{R}_i^t \mathbf{f}_{c,i}^t. \end{aligned}$$

In practice, we use the GAM estimation routines implemented in the *R* package *mgcv* [126] to choose the penalization parameter and estimate the model at the same time.

7.3.6 HALSX with other regression models

We can replicate the strategy above to work with other regression models. As mentioned before, the convergence to critical point is guaranteed, as long as the regression model estimation can be re-parameterized to verify the conditions of Proposition 1. Using this strategy, many off-the-shelf algorithms for regression model training can be plugged in. In the experiments described in the next section, we use the predictive model API provided in the *R* package *caret* [127].

Meta-parameters in regression models

As in HALSX-GAM, we build the estimation of meta-parameters using cross validation as a part of the link function estimation step, can treat them indifferently as regular parameters.

7.3.7 An HALS-like algorithm for (7.8)

Before detailing Algorithm 5 which aims to solve (7.8), we will first develop the elemental HALS iteration in the context of (7.8) where no supplemental information is supplied for the factorization model, namely $\mathbf{X}_r = \mathbf{I}_{n_1}, \mathbf{X}_c = \mathbf{I}_{n_2}$. Indeed, when updating one column of $\mathbf{F}_r$, the sub-problem becomes: how to solve $\arg\min_{\mathbf{f}} \|\mathbf{b} - \mathcal{A}(\mathbf{f}(\mathbf{f}_c)^T)\|_2^2$?

We will use the fact that for all $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$,

$$\mathcal{A}(\mathbf{M}) = (\langle \mathbf{A}_i, \mathbf{M} \rangle)_{1 \leq i \leq N},$$

and for all $\mathbf{b} \in \mathbb{R}^N$, $\mathcal{A}^*$, the transpose of $\mathcal{A}$ is defined by

$$\mathcal{A}^*(\mathbf{b}) = \sum_{i=1}^{N} b_i \mathbf{A}_i.$$

Since

$$\frac{\partial}{\partial \mathbf{f}} \|\mathbf{b} - \mathcal{A}(\mathbf{f}(\mathbf{f}_c)^T)\|_2^2 = \mathcal{A}^*[\mathcal{A}(\mathbf{f}(\mathbf{f}_c)^T) - \mathbf{b}]\mathbf{f}_c,$$

the first order optimality condition $\frac{\partial}{\partial \mathbf{f}} \|\mathbf{b} - \mathcal{A}(\mathbf{f}(\mathbf{f}_c)^T)\|_2^2 0$ is therefore equivalent to

$$\mathcal{A}^*[\mathcal{A}(\mathbf{f}(\mathbf{f}_c)^T)]\mathbf{f}_c = \mathcal{A}^*[\mathbf{b}]\mathbf{f}_c.$$

The left-hand side of the equation can be written as

$$\begin{aligned}
\mathcal{A}^*[\mathcal{A}(\mathbf{f}(\mathbf{f}_c)^T)]\mathbf{f}_c &= (\sum_{i=1}^{N} \langle \mathbf{A}_i, \mathbf{f}(\mathbf{f}_c)^T \rangle \mathbf{A}_i)\mathbf{f}_c \\
&= \sum_{i=1}^{N} \mathrm{Tr}(\mathbf{f}(\mathbf{A}_i\mathbf{f}_c)^T)(\mathbf{A}_i\mathbf{f}_c) \\
&= \sum_{i=1}^{N} (\mathbf{A}_i\mathbf{f}_c)\mathrm{Tr}((\mathbf{A}_i\mathbf{f}_c)^T\mathbf{f}) \\
&= \sum_{i=1}^{N} (\mathbf{A}_i\mathbf{f}_c)(\mathbf{A}_i\mathbf{f}_c)^T\mathbf{f},
\end{aligned}$$

which leads to the following symmetric n_1-by-n_1 system on $\mathbf{f}$:

$$(\sum_{i=1}^{N} (\mathbf{A}_i\mathbf{f}_c)(\mathbf{A}_i\mathbf{f}_c)^T)\mathbf{f} = \sum_{i=1}^{N} b_i \mathbf{A}_i \mathbf{f}_c,$$

or

$$\mathbf{f} = (\sum_{i=1}^{N}(\mathbf{A}_i\mathbf{f}_c)(\mathbf{A}_i\mathbf{f}_c)^T)^{-1}\sum_{i=1}^{N} b_i\mathbf{A}_i\mathbf{f}_c$$

This computation generalizes to linear exogenous variables, with the optimality condition:

$$\boldsymbol{\beta} = (\sum_{i=1}^{N}(\mathbf{X}\mathbf{A}_i\mathbf{f}_c)(\mathbf{X}\mathbf{A}_i\mathbf{f}_c)^T)^{-1}\sum_{i=1}^{N} b_i\mathbf{A}_i\mathbf{f}_c.$$

When the matrices to be inversed in these equations are not invertible, we will use the generalized inverse instead.

Using these elementary steps, we propose Algorithm 5 to solve Problem (7.8). Compared to Algorithm 4, Algorithm 5

- has one less block (the slack variable $\mathbf{V}$ is not present);
- checks the deviation with data more frequently;
- each subproblem is more costly because of the presence of $\mathcal{A}$ in the subproblem. As we will see in the detailed development of the computation, when N, the sample size (the dimension of image of $\mathcal{A}$) is large, each update involves rather costly computations.

Algorithm 5 Hierarchical Alternating Least Squares with eXogeneous variables for NMF (HALSX2)

Require: Measurement operator $\mathcal{A}$, measurements $\mathbf{b}$, rank $1 \leq k \leq \min\{n_1, n_2\}$, features $\mathbf{X}_r$ and $\mathbf{X}_c$, functional spaces F_r and F_c in which to search the link functions.
  Initialize $\mathbf{F}_r^0, \mathbf{F}_c^0 \geq 0, t = 0$
  while Stopping criterion is not satisfied **do**
    $\mathbf{R}^t = \mathbf{b} - \mathcal{A}(\mathbf{F}_r^t(\mathbf{F}_c^t)^T)$
    for $i = 1, 2, ..., k$ **do**
      $\mathbf{R}^t = \mathbf{R}^t + \mathcal{A}(\mathbf{f}_{r,i}^t(\mathbf{f}_{c,i}^t)^T)$
      $f_{r,i}^{t+1} = \arg\min_{f \in F_r} \|\mathbf{R}^t - \mathcal{A}(f(\mathbf{X}_r)(\mathbf{f}_{c,i}^t)^T)\|_2^2$
      $\mathbf{f}_{r,i}^{t+1} = \max(0, f_{r,i}^{t+1}(\mathbf{X}_r))$
      $\mathbf{R}^t = \mathbf{R}^t - \mathcal{A}(\mathbf{f}_{r,i}^{t+1}(\mathbf{f}_{c,i}^t)^T)$
    end for
    for $i = 1, 2, ..., k$ **do**
      $\mathbf{R}^t = \mathbf{R}^t + \mathcal{A}(\mathbf{f}_{r,i}^{t+1}(\mathbf{f}_{c,i}^t)^T)$
      $f_{c,i}^{t+1} = \arg\min_{f \in F_c} \|\mathbf{R}^t - \mathcal{A}(\mathbf{f}_{r,i}^{t+1} f(\mathbf{X}_c)^T)\|_2^2$
      $\mathbf{f}_{c,i}^{t+1} = \max(0, f_{c,i}^{t+1}(\mathbf{X}_c))$
      $\mathbf{R}^t = \mathbf{R}^t - \mathcal{A}(\mathbf{f}_{r,i}^{t+1}(\mathbf{f}_{c,i}^{t+1})^T)$
    end for
    $t = t + 1$
  end while
output $\mathbf{F}_r^t \in \mathbb{R}_+^{n_1 \times k}, f_{r,1}^t, ..., f_{r,k}^t \in F_r$,
  $\mathbf{F}_c^t \in \mathbb{R}_+^{n_2 \times k}, f_{c,1}^t, ..., f_{c,k}^t \in F_c$.

Complexity

At each sub-iteration Algorithm 5, we need to calculate N n_1-by-n_1 or n_2-by-n_2 matrices $((\mathbf{A}_i\mathbf{f}_c)(\mathbf{A}_i\mathbf{f}_c)^T)$, then inverse the sum of the these N matrices. While the computation of the sum is map-reducible, on a single-threaded machine, this can be computationally expensive when N is large. Each iteration of Algorithm 5 has a multiplicative complexity of $O(kN(n_1^2 + n_2^2))$, while each iteration of Algorithm 4 has a complexity of $O(k(n_1^2+n_2^2)+Nn_1n_2)$ with general linear measurement operator. This difference in complexity can be very important when N or k is large.

7.4 Experiments

7.4.1 Datasets

We use one synthetic dataset and three real datasets to evaluate the proposed methods.

- **Synthetic data**: a rank-20 150-by-180 nonnegative matrix simulated following the generative model (Section 7.1.1), with $\mathbf{X}_r \in \mathbb{R}^{150\times 3}$ and $\mathbf{X}_c \in \mathbb{R}^{180\times 4}$ matrices with independent Gaussian entries, $f_r : \mathbb{R}^3 \to \mathbb{R}^{20}$ ($f_c : \mathbb{R}^4 \to \mathbb{R}^{20}$) is a function formed with a dimension-33 (44 for f_c) spline basis with random weights, truncated at 0 ($T = 150, N = 180$). The simulated features matrices are used as side information.
- **French electricity consumption** (proprietary dataset): daily consumption of 473 medium-voltage feeders gathering each around 1,500 consumers based near Lyon in France from 2010 to 2012. The first two years are used as training data ($T = 1096, N = 473$). The daily temperature of a weather station in this area, calendar variables such as weekday/weekend, position of the year, bank holidays, and percentage of four types of clients (residential, professional, industrial, high-voltage clients) are used as side information (similar as in Chapter 4).
- **Portuguese electricity consumption** [28] daily consumption of 370 Portuguese clients from 2010 to 2014 ($T = 1461, N = 369$). The daily temperature in Portugal and calendar variables are used as side information.
- **MovieLens 100k** [128] an anonymized public dataset with 100,000 movie scores for 1682 movies from 943 users ($T = 943, N = 1682$). This is a standard public dataset for matrix completion. Note that the data matrix is not complete. Error rates are calculated on the vector of available scores. The genres of the movies, and gender, age and profession of the users are used as side information.

The subsets of the two real-world electricity consumption datasets are also used in Chapter 6.

7.4.2 Validation procedure

Several methods that can be used in time series recovery and prediction are compared to the HALSX algorithm proposed in this chapter.

Among these methods, the following methods (introduced in Chapter 6) are used to compare the matrix recovery/completion performance.

- **empty_model** For temporal aggregates only: to recover the target matrix, temporal aggregates are distributed equally over the covered periods.

- **HALS, NeNMF, softImpute** Matrix recovery/completion methods without side information. See the experiments of the previous chapter for more details.

The following methods use side information, and can be used for time series prediction of new columns and/or rows from incomplete data:

- **individual_gam** Estimating separate GAMs on each column or row, on the matrix obtained from **empty_model** or on the whole data matrix when it is available.
- **factor_gam** Estimating GAMs on the factors obtained by **HALS** or **NeNMF**.
- **rrr** [129] Applying reduced-rank regression on the matrix obtained from **empty_model** or on the whole data matrix when it is available.
- **grmf** [113] A matrix completion algorithm using graph-based side information to enhance collaborative filtering performance.
- **trmf** [26] A matrix completion algorithm tailored to time series, by adding three penalization terms to the matrix factorization quadratic error. When only temporal aggregate measurements are available, we apply this method on the matrix obtained from **empty_model**.
- **HALSX_model** Algorithm 4.

For all matrix factorization methods (HALS, NeNMF, rrr, factor_gam, grmf, trmf, HALSX_model), we use the method with several ranks, then choose the best rank ($k \in \{2, 3, ..., 20\}$ for synthetic and Portuguese data, $k \in \{2, 3, ..., 10\}$ for French and MovieLens data). For trmf, we do a grid search on the three penalization parameters, and choose the best combination.

As explained in the previous section, a regression method needs to be specified in order to use HALSX_model. This regression method specifies the fonctional spaces F_r and F_c in which the link functions are to be looked for. In our experiments, we use four different regression methods with HALSX: linear model, GAM, support vector regression with linear kernel, and Gaussian process regression with radial basis kernel (**lm**, **gam**, **svmLinear**, **gaussprRadial**). We use the implementation of **lm** in standard `R`, and **svmLinear** and **gaussprRadial** of the `R` package `kernlab`[130], through the `caret` API. As of **gam**, we use the `mgcv` implementation directly.

For each data matrix, we apply a linear measurement operator on an upper-left submatrix to obtain the measures (of dimension 100-by-130 for synthetic data, 730-by-270 for French data, 731-by-369 for Portuguese data, and 666-by-1189 for MovieLens data). A random shuffle between columns is done before dividing the matrix into submatrices. As in Chapter 6, on time series datasets (the first three), both periodic and random temporal aggregates are generated as observations. On the MovieLens dataset, observations are random entries, as in matrix completion. Using the measurements of the upper-left submatrix, we use each method to estimate a matrix factorization model, with or without side information.

We then report the matrix recovery error on this sampled submatrix for all methods, and the prediction errors on the rest of the data matrix for the methods can produce predictions for new columns and/or rows. We distinguish row prediction error, column prediction error, and row-column prediction error, as the error calculated on the lower-left, upper-right and lower-right submatrix (see Figure 7.1).

We report the relative root-mean-squared error as error metric for time series datasets:

- for electricity datasets, $\text{RRMSE}(\mathbf{V}, \mathbf{V}^*) = \frac{\|\mathbf{V}-\mathbf{V}^*\|_F}{\|\mathbf{V}^*\|_F}$,

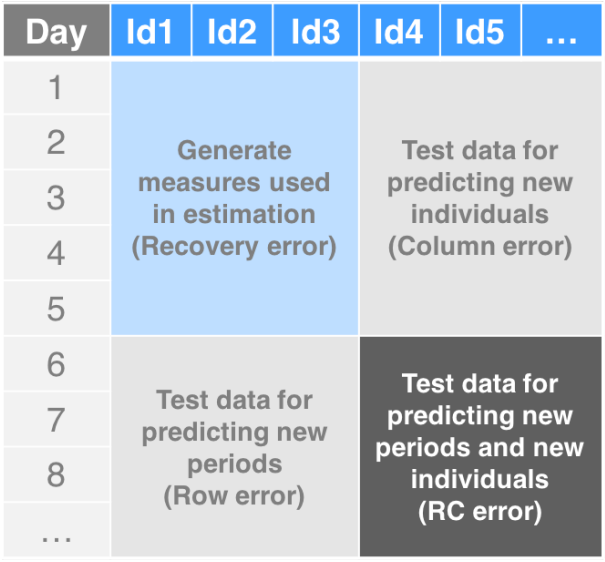


Figure 7.1: *Dividing the data matrix into four submatrices: one for generating observations to estimate the model, the other three for prediction.*

- for MovieLens, we calculate the l_2-norm version on the vector of all available movie scores.

In preliminary tests, other error metrics were also evaluated, and were not qualitatively different from the chosen metric.

7.4.3 Execution time and precision between HALS and HALS2

To show-case the computational complexity difference discussed in Section 7.3.7, we run HALS (Algorithm 4) and HALS2 (Algorithm 5) on random temporal aggregate masks, with no side information. In HALS, we treated observations as general linear measurements, without using the efficient simplex projection specific to temporal aggregate masks discussed in Chapter 6. The per iteration execution time is shown in Figure 7.2. The difference in complexity discussed in Section 7.3.7 is fairly clear here. In HALS2, the execution time per iteration increases both with the rank and the number of samples in data, at least for sampling rate from 14.3% on. For low sampling rates, HALS2 often diverges, where as HALS is always stable.

As discussed in Section 7.3.7, although the execution time per iteration is greater, HALS2 could be more efficient if it does much less iterations than HALS. In Figure 7.3, we can see that this is not the case: for problems with high sampling rates, HALS2 indeed does less iterations to converge. However, the total execution time is still larger than HALS. For lower sampling rates, HALS2 has troubles converging, and only stops when the maximal execution time allowed (300 seconds) is reached. This is also confirmed in Figure 7.4, where HALS2 has much worse recovery error than HALS.

Given these results, we will only use the HALSX (Algorithm 4), the version of HALS with side information, in the following tests.

7.4.4 Performance on temporal aggregate measurements

On the synthetic and the two real electricity consumption datasets, we use Algorithm 4 to perform matrix recovery and prediction on temporal aggregate measurements. Every method

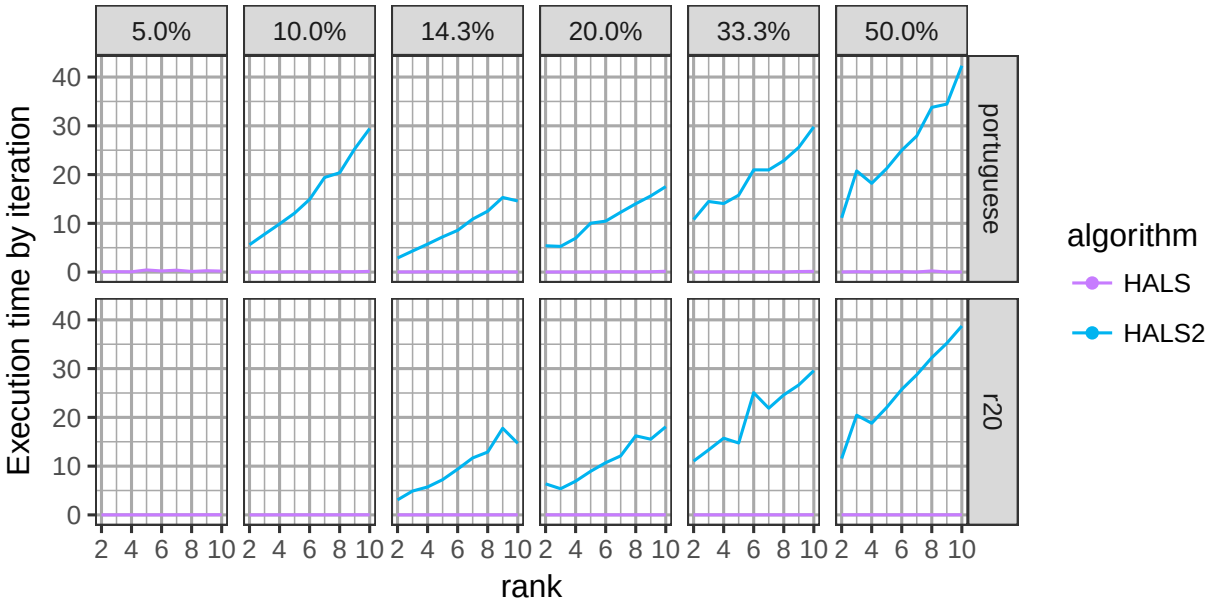


Figure 7.2: *Execution time per iteration of Algorithm 4 and Algorithm 5 without exogenous variables*

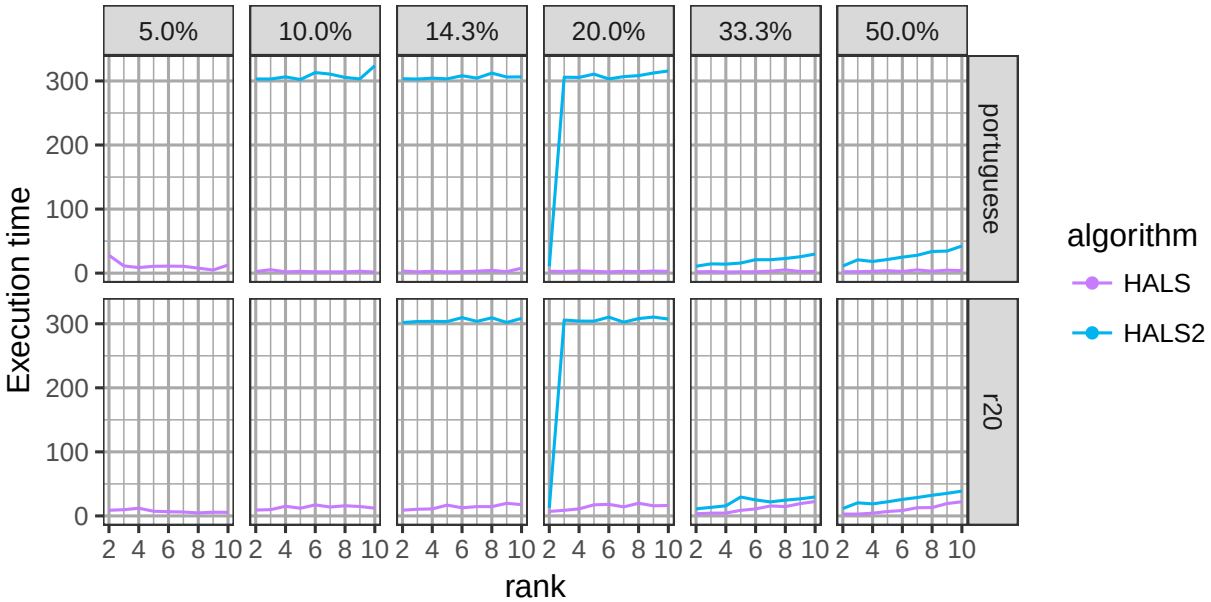


Figure 7.3: *Total execution time of Algorithm 4 and Algorithm 5 without exogenous variables*

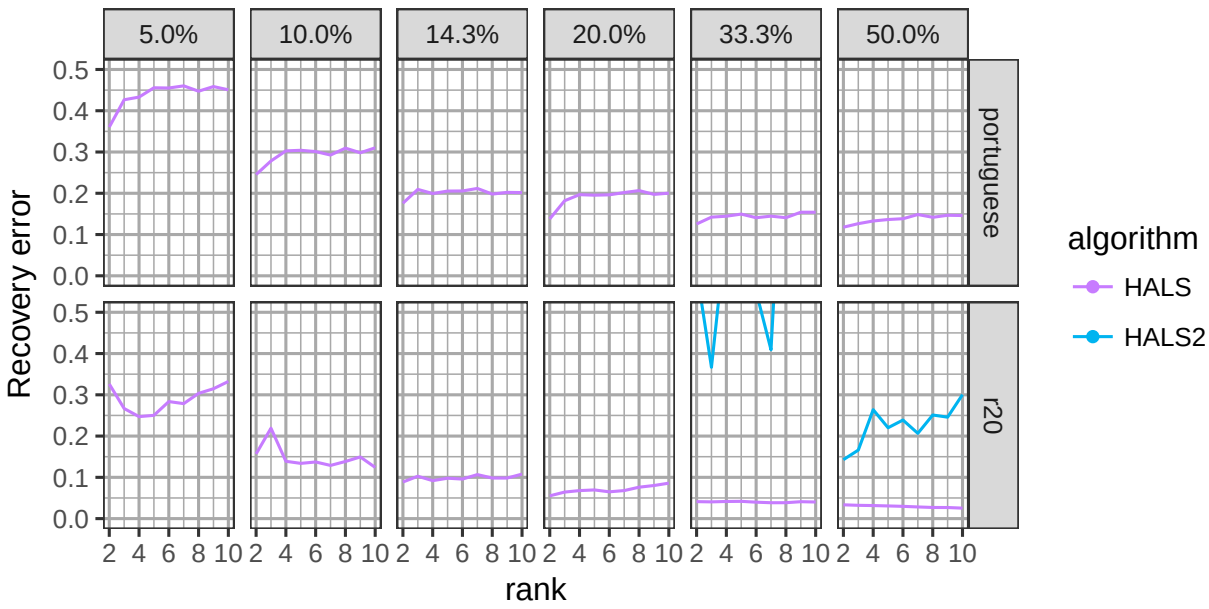


Figure 7.4: *Reconstitution precision of Algorithm 4 and Algorithm 5 without exogenous variables*

except for **grmf** is used in this setting.

We use two types of temporal aggregate measurements: periodic and random. In periodic measurements, each scalar measure covers a fixed number of periods of one individual. In random measurements, the number of periods covered by a measure is random (see Chapter 6 for more details). In electricity consumption data, periodic measurements are closer to the actual meter reading schedules of utility companies, while matrix recovery with random measurements is an easier problem. For both sampling types, we sample 10%, 20%, ..., 50% of the data to show-case the matrix recovery performance of the proposed method.

For the synthetic data, we use the true row and column features used to produce the simulations. For the French electricity data, the row features are variables known to have an influence on electricity consumption: the temperature, the day type (weekday, weekend, or holiday), the position of the year. The column features are the percentage of residential, professional, or industrial usages in the group of users for each column. For the Portuguese electricity data, as no individual features are available, we only use the same row features as for the French dataset (temperature, day type, position of the year).

Figure 7.5 shows the matrix recovery error. For most of the scenarios, HALSX_models (red lines with symbols) are comparable or better than the other methods without side information. The only case where an HALSX_model is a little worse is when compared with HALS and NeNMF (two NMF methods) in synthetic data with random measurements, which is the least close to the real application. As is seen in Chapter 6, the **softImpute** [111] method is not well adapted to temporal aggregate measurements, and has much higher error (higher than the maximal value in these graphics. When comparing the four regression methods used in HALSX, we see that GAM and Gaussian process regression (**gam** and **gaussprRadial**) are the best for synthetic data, linear model (**lm**) is the best for Portuguese data, and Gaussian process regression (**gaussprRadial**) is the best for French data.

Figures 7.6, 7.7, and 7.8 show the prediction error on the three datasets. We can see that **trmf** [26] and **rrr** [129] are not well adapted to temporal aggregate measurements. When they are applicable (**trmf** is only applicable to row prediction, **rrr** only applicable to row or column predictions, not RC predictions), they have much worse performance than the other

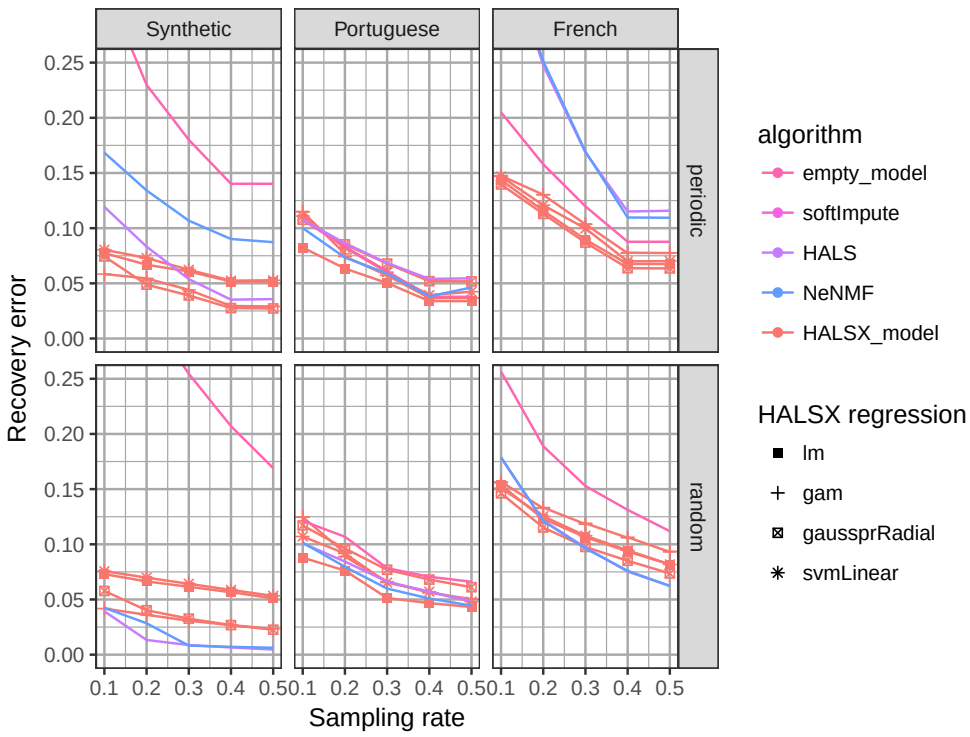


Figure 7.5: *Recovery performance on synthetic and real electricity data*

methods, except in Synthetic data with complete observations. In most cases, HALSX_models are comparable to or better than factor_gam and individual_gam, which shows that using side information while estimating the factorization model produces factors more adapted for prediction. When comparing the regression methods used with HALSX_models in prediction, **gam** is consistently the best for synthetic and Portuguese data, **gaussprRadial** best for the French dataset.

It is interesting to note that the performance HALSX_models is the least sensitive to sampling rates: it is mostly constant from 30% of data. This shows that using side information is supplementary to observing more data.

7.4.5 Performance on matrix completion mask

On the MovieLens dataset, we use Algorithm 4 to perform matrix recovery and prediction with uniformly sampled matrix entries. The sampling rates are $\{10\%, 20\%, 30\%, 40\%, 50\%, 90\%\}$.

As is the case for time series datasets, we use samples from the upper-left submatrix to estimate the model, evaluate matrix recovery on that submatrix, and evaluate row and/or column prediction errors. Every method except for trmf is used in this setting. As side information, we use the gender, the age, and the occupation of the users and the genre (a dimension-19 binary variable) of the movies.

For **grmf**, we produce a graph where each individual is connected to its ten nearest neighbors with euclidean distance with the features, as suggested in the reference [113]. As the parameters estimated in **grmf** is per user/movie, it can not be used to predict new individuals, even though it uses side information.

Figure 7.9 shows the recovery error on MovieLens data. We can see that in the low sampling rate case (10%), HALSX_model with **lm** and **gam** works the best. In higher sampling

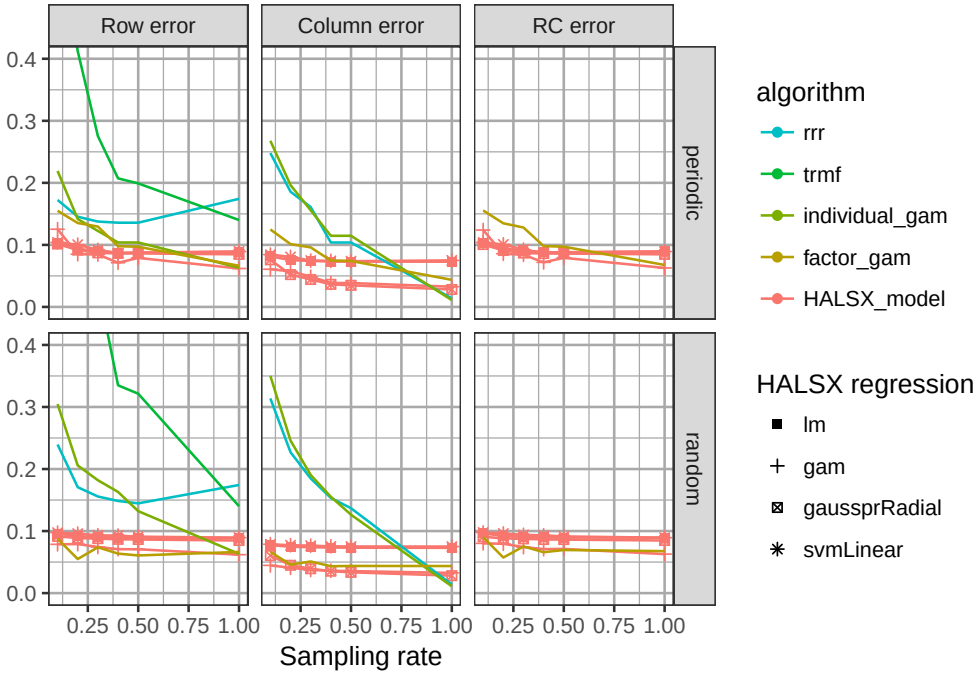


Figure 7.6: *Prediction performance on synthetic data*

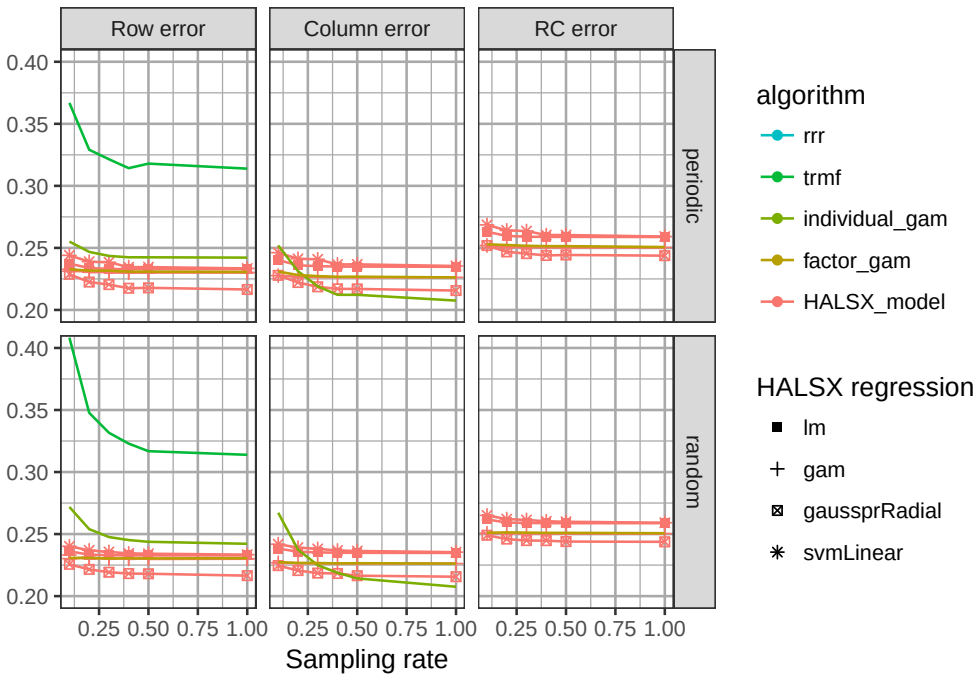


Figure 7.7: *Prediction performance on real French electricity data*

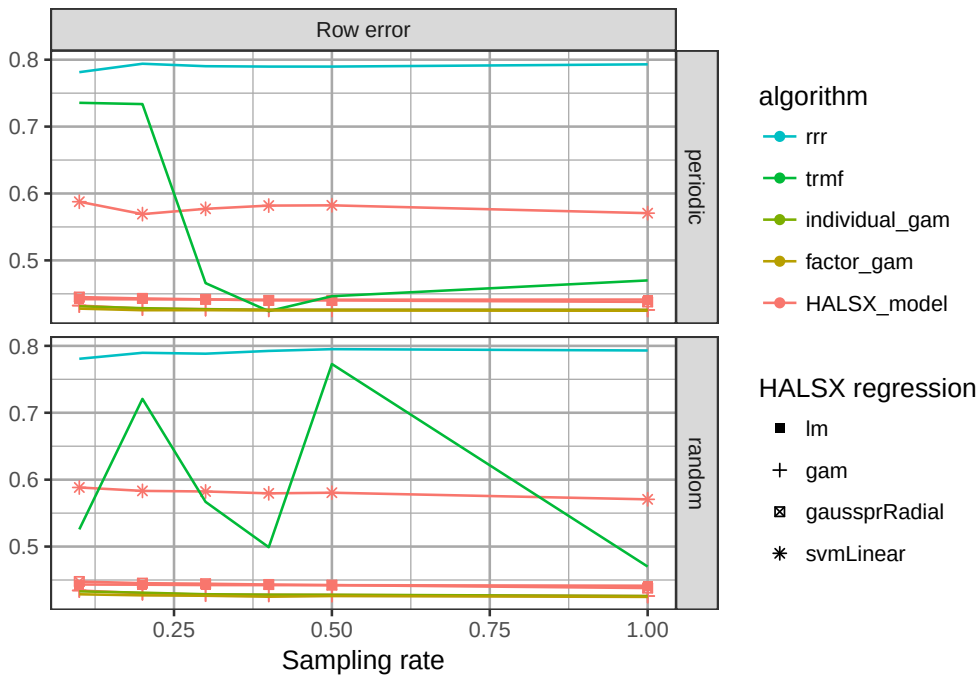


Figure 7.8: *Prediction performance on real Portuguese electricity data*

rate cases, the NMF methods without side information (HALS and NeNMF) work the best, with HALSX_model with **lm**, **gam** or **svmLinear** are second to best. HALSX_model with **gaussprRadial** does not work very in this case, as is the case with the other comparison methods (grmf, softImpute and empty_model)

Figure 7.10 shows the prediction error on MovieLens data for new users and/or new movies. The order of the variants of the HALSX_models is conserved: **gam** and **lm** are the best, **svmLinear** is not very good for 10%, but better with higher sampling rates, and **gaussprRadial** does not work well in this problem. Otherwise, the factor_gam method is slightly better for column predictions (new movies) in higher sampling rate cases, but worse in other cases. Both individual_gam and rrr are much worse than the proposed methods.

7.5 Conclusion

Motivated by electricity consumption estimation, we proposed a general approach for including side information on the columns and row in nonnegative matrix factorization methods, with general linear measurements. Based on a generative model, the framework we propose generalizes many prior works in multivariate regression and matrix factorization.

In order to explore the identifiability of the model, we established a sufficient condition on the features for the factorization to be unique. We deduced HALSX, a general algorithm to estimate the generative model, and showed that the algorithm converges to a critical point in rather mild conditions.

The proposed algorithm is tested on synthetic and real datasets, both in electricity consumption and recommendation systems. In various sampling scenarios, HALSX produced better or equivalent performance both in matrix recovery and in prediction, compared to a number of reference methods.

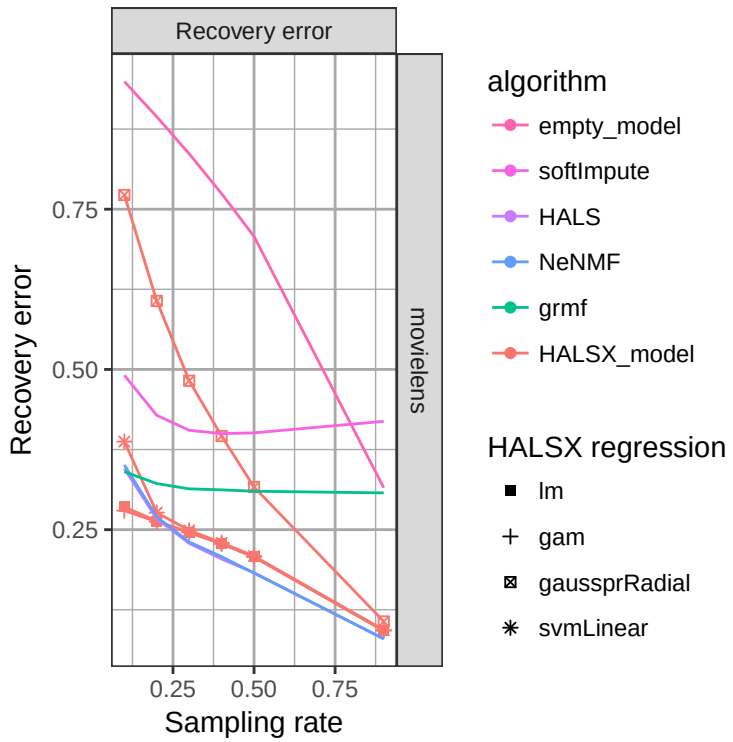


Figure 7.9: *Matrix completion performance on MovieLens 100K data*

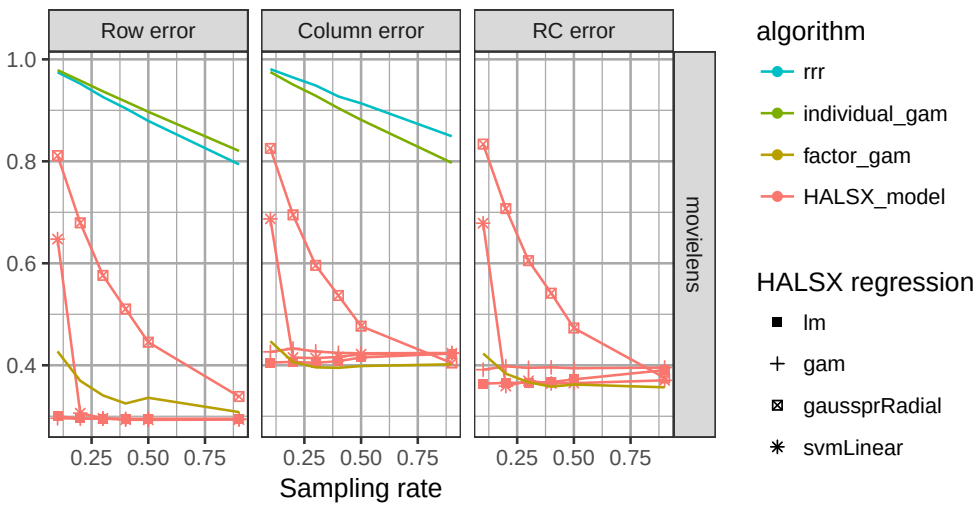


Figure 7.10: *Matrix completion performance for new rows and new columns on MovieLens 100K data*

Chapter

8 Comparing matrix factorization with traditional methods

Contents

At this point of the thesis, we have seen both several methods that are derived from different contexts: spatial statistics (Chapter 3), socio-demographic clustering (Chapter 4), and NMF (Chapters 5, 6, 7). In this chapter, we discuss the link between the main proposition of this thesis, using nonnegative matrix factorization to recover and predict partially observed time series, with the two state-of-art methods investigated in Chapters 3 and 4.

In 8.1, we apply the proposed general matrix factorization framework to spatial and spatio-temporal kriging problems. In Section 8.2, we argue that the method proposed in Chapter 4 can be seen as a special case of matrix factorization with exogenous variables.

8.1 Kriging and matrix factorization

Consider the following matrix factorization model:

$$\begin{aligned} \min_{\mathbf{V}, f_r \in F_r^k, f_c \in F_c^k} \quad & \|\mathbf{V} - f_r(\mathbf{X}_r) f_c(\mathbf{X}_c))^T\|_F^2, \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{V}) = \boldsymbol{\alpha}. \end{aligned} \tag{8.1}$$

As in Chapter 7, $\mathbf{V}$ is a n_1-by-n_2 matrix, $F_r \subseteq (\mathbb{R})^{\mathbb{R}^{d_1}}$ and $F_c \subseteq (\mathbb{R})^{\mathbb{R}^{d_2}}$ are the functional spaces in which we try to find the link functions, $\mathbf{X}_r$ and $\mathbf{X}_c$ are known row and column features, $\mathcal{A}$ a linear measurement operator, and $\boldsymbol{\alpha}$ is the measurements that we obtained by applying $\mathcal{A}$ on the ground truth $\mathbf{V}^*$. For the moment, we do not constrain the factors to be nonnegative.

In particular, we will consider the column features $\mathbf{X}_c$ to be spatial coordinates of the column individuals, and the row features $\mathbf{X}_r$ to be temporal coordinates (timestamps).

8.1.1 Kriging as matrix factorization with a feature map

The link between Gaussian process regression and kernel regression has been extensively studied (see [44, Chap. 2]). In this section, we first examine a toy model, to see that univariate kriging is a special case of (8.1).

Consider the simple case where $n_1 = 1$, and the measurements are simply the observations at the first $n_2 - 1$ locations. Since the matrix to be estimated is of dimension 1-by-n_2, the rank k can only be 1. We set the column factor $f_c(\mathbf{X}_c)$ to be 1, since it is just a real number. We are then in the exact same setting as univariate kriging considered in Chapter 3. That is, knowing the value of $\mathbf{V}^*$ at locations $\mathbf{x}_{c,1}, \mathbf{x}_{c,2}, ..., \mathbf{x}_{c,n_2-1}$: $v_1^*, v_2^* ..., v_{n_2-1}^*$, we'd like to estimate the its value at position $\mathbf{x}_{c,n_2}$: $v_{n_2}^*$.

In simple kriging, the idea is to suppose that the underlying Gaussian process is second-order stationary. By minimizing the quadratic risk, we obtain that the optimal prediction is

$$v_{n_2} = (C(\mathbf{x}_{c,1} - \mathbf{x}_{c,n_2}), ..., C(\mathbf{x}_{c,n_2-1} - \mathbf{x}_{c,n_2}))(C(\mathbf{x}_{c,i} - \mathbf{x}_{c,j})_{1 \leq i,j \leq n_2-1})^{-1}(v_1^*, v_2^* ..., v_{n_2-1}^*)^T,$$

where C is the covariance function of the Gaussian process.

Another way to view the kriging result is the following: we transform the spatial coordinates with a feature map ϕ, so that

$$\phi(\mathbf{x}) = (C(\mathbf{x}_{c,1} - \mathbf{x}), C(\mathbf{x}_{c,2} - \mathbf{x}), ..., C(\mathbf{x}_{c,n_2-1} - \mathbf{x}))^T,$$

and we restrict the link function f_c to be a linear function of the transformed features. This way, the matrix factorization problem (8.1) becomes

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_2-1}, \mathbf{V} \in \mathbb{R}^{1 \times n_2}} \quad & \|\mathbf{V} - (\phi(\mathbf{X}_c)\mathbf{w})^T\|_F^2, \\ \text{s.t.} \quad & v_1 = v_1^*, \quad ..., \quad v_{n_2-1} = v_{n_2-1}^*. \end{aligned}$$

We can easily obtain that the optimal value of $\mathbf{w}$ is $(C(\mathbf{x}_{c,i} - \mathbf{x}_{c,j})_{1 \leq i,j \leq n_2-1})^{-1}(v_1^*, v_2^* ..., v_{n_2-1}^*)^T$. This yields exactly the same optimal prediction value for v_{n_2}.

8.1.2 Matrix factorization and sum-product covariance functions

Here we try to recover a matrix $\mathbf{V}^*$ with more than one row and column: both n_1 and n_2 are potentially larger than 2. The covariance structure of spatio-temporal processes is an object that is more complex, as discussed in Chapter 3. A spatio-temporal covariance function C can be evaluated on any spatio-temporal increment $(\mathbf{s}, \tau)$. Typically, we would evaluate C on $(\mathbf{s}, \tau)$ where $\mathbf{s}$ is the (element-wise) difference between two rows in $\mathbf{X}_c$, and τ the difference between two rows in $\mathbf{X}_r$. We start by drawing the analogy between a separable spatio-temporal process with a rank-1 matrix factorization model. We use the subscript index r and c to designate temporal and spatial components, as rows correspond to periods and columns correspond to spatially located individuals in our notation.

A separable spatio-temporal covariance function is one that verifies $C(\mathbf{s}, \tau) = C_r(\tau)C_c(\mathbf{s})$. One spatio-temporal process that verifies such a covariance function is the product of a spatial Gaussian process Z_c and a temporal Gaussian process Z_s. In this case, it is clear that the matrix that we wish to recover, $\mathbf{V}^*$, has a rank-1 factorization $\mathbf{f}_r\mathbf{f}_c^T$, where $\mathbf{f}_r$ and $\mathbf{f}_r$ are simply the observations of the temporal and spatial processes at the periods and the locations in the dataset. Obviously, a rank-1 matrix has proportional rows and columns. Therefore, if one single row and one column are entirely observed, the matrix completion or kriging problem becomes trivial in this case.

As mentioned in Chapter 3, a way to obtain slightly more general covariance structure is to use sum-product covariance. Here, consider a sum-product covariance with k independent separable components:

$$C(\mathbf{s}, \tau) = \sum_{i=1}^{k} C_r^i(\tau)C_c^i(\mathbf{s}).$$

In a similar fashion as in the univariate case, we use the spatial and temporal covariance functions to create feature maps ϕ_r^i and ϕ_c^i, where

$$\phi_c^i(\mathbf{x}_c) = (C_c(\mathbf{x}_{c,1} - \mathbf{x}_c), C_c(\mathbf{x}_{c,2} - \mathbf{x}_c), ..., C_c(\mathbf{x}_{c,n_2} - \mathbf{x}_c))^T,$$

for all spatial coordinates $\mathbf{x}_c$, and

$$\phi_r^i(x_r) = (C_r(x_{r,1} - x_r), C_r(x_{r,2} - x_r), ..., C_r(x_{r,n_1} - x_r))^T,$$

for all temporal coordinates x_r. This leads to the following matrix factorization problem:

$$\begin{aligned} \min_{\mathbf{W}_r \in \mathbb{R}^{n_1 \times k}, \mathbf{W}_c \in \mathbb{R}^{n_2 \times k}, \mathbf{V} \in \mathbb{R}^{n_1 \times n_2}} \quad & \|\mathbf{V} - \sum_{i=1}^{k} \phi_r^i(\mathbf{X}_r)\mathbf{w}_r^i(\phi_c^i(\mathbf{X}_c)\mathbf{w}_c^i)^T\|_F^2, \\ \text{s.t.} \quad & \mathcal{A}(\mathbf{V}) = \boldsymbol{\alpha}. \end{aligned}$$

In practice, the idea would be not to specify the feature maps (hence the covariance components) by hand, but let Algorithm 4 find appropriate ones. This can be done by specifying for example a class of kernel functions, and let the regression procedure find the window sizes by cross validation for each rank.

8.2 Matrix factorization and socio-demographic clustering

In Chapter 4, we proposed a hybrid method which has two components:

- a clustering component, which groups individuals into clusters, based on socio-demographic characteristics;
- a regression component, which proposes an estimation of the electricity consumption of all individuals of a cluster, with temporal covariates such as the temperature, the hour of the day, and the time of year.

These two components can easily be transformed into the nonnegative matrix factorization framework $\mathbf{V}^* = f_r(\mathbf{X}_r)f_c(\mathbf{X}_c)$, where

- $\mathbf{X}_r$ is the temporal covariates, f_r is the regression models of consumption, in the same way as in Chapter 7;
- $\mathbf{X}_c$ is the socio-demographic characteristics, and f_c is the classifier: that is, for an individual with socio-demographic features $\mathbf{x}_c$, $f_c(\mathbf{x}_c) = \mathbf{e}_i$ if this individual is in the i-th cluster, where $\mathbf{e}_i$ is the i-th canonical vector in $\mathbb{R}^k$. In the framework of Algorithm 4, this can be achieved by estimating f_c as a multinomial logistic regression.

The method proposed in Chapter 4 is thus a special case of Algorithm 4, where we only do one iteration.

8.3 Matrix factorization as feature generation

It is interesting to note that the analogy between matrix factorization and some of the more recent feature generation methods in machine learning, in particular in the neural network literature.

Matrix factorization is an extremely simple neural network: a depth-two auto-encoder with linear activation. One of the reasons why people are interested in convergence results of matrix factorization is because these analyses can shed light on the theoretical understanding of deeper neural networks [90].

Recent trends in machine learning showed that automatic feature generation methods, such as word2vec in natural language processing (which is also a depth-two neural network) and variational auto-encoders in image processing, can have very good empirical performance in applications. Similarly, in this chapter, we saw that the two more hand-tuned methods proposed in Chapters 3 and 4 can actually be written as special cases of the general approach proposed in Chapter 7.

Chapter

9 Implementation of nonnegative matrix factorization algorithms

Contents

During the thesis, an R package called meterModels has been developped to implement the algorithms proposed in the previous two chapters. In this chapter, we describe the main functionalities of meterModels. When it is necessary, we will also provide some simple code to illustrate how to use the package.

The meterModels package includes several matrix factorization algorithms, as well as many helper functions to perform experiments comparing different methods. Most of the package is written in R. Heavy computations are translated in C++ to increase the performance of the package, through the RCpp API. When available, openMP can be used to perform distributed computation in model estimation. On a multi-core machine, parallel estimation of multiple models can be used through the parallel package in R.

On a single machine, the package can be used on matrices with several hundred columns and 100,000 rows in less than ten minutes.

9.1 Data representation

Most matrix factorization software known to us can only deal with observations in the form of matrix entries, thus only adapted to matrix completion. Given the applications considered in this thesis, we developed meterModels to handle more general measurements. To achieve this, meterModels provides several data representation formats, for temporal aggregate data (Section 6) and general linear measurements. We describe these formats in this section.

This chapter is based on the vignette of meterModels.

9.1.1 incremental_data: A fully-observed data matrix

As indicated by the name, the object class `incremental_data` is particularly motivated by electricity metering. Basically, `incremental_data` is a full data matrix (in the field `incremental_data$data`), with a `string` in `incremental_data$data_id` as the name of the dataset, and optionally exogenous variables provided through `incremental_data$exo_variable`.

```
# create a rank-2 nonnegative 10-by-15 matrix as example
m = 10
n = 15
r = 2
example_matrix = matrix(rexp(m * r), ncol = r) %*%
  matrix(rexp(n * r), ncol = n)

# create an incremental_data object
incre_d = incremental_data(x = example_matrix, data_id = "exponential")
```

9.1.2 index_data: Temporal aggregates

`index_data` is to be used to provide temporal aggregate data to a model estimation algorithm. The class `index_data` is thus named because a meter reading is sometimes called an "index". To create an object of class `index_data`, there are two ways:

- We can provide an `incremental_data` object, and an object of the class `sampling_scheme` to tell the program how to sample the data matrix. For example, in the following code, we initiated a sampling scheme, `sampling_incre_d_at` which randomly takes 50% of the temporal aggregates:

```
seed = 42

# we will sample incre_d with a random sampling_scheme
# with a 50% sampling rate and a fixed seed
sample_incre_d_at = sampling_scheme(
  sampling_rate = 0.5, type = "random", seed = seed)

# create an index_data object
index_d = index_data(x = sample_incre_d_at, incremental_data = incre_d)
lapply(index_d,head)
```

```
## $sample_data
## [1] 21.409480 33.740012  5.029540  2.955382  7.894554  9.510452
##
## $sample_index
## [1] 151 154  47 135 104  84
##
## $data_id
## [1] "exponential"
##
## $dim_data
## [1] 10 15
```

- Or we can provide the dimension of the matrix to be estimated, the values of readings, and the position of the readings in the aggregated version of the corresponding `incremental_data` object. In the following code, we create the same sample as above, using directly the samples in `index_d`.

```
index_d2 = index_data(x = index_d$sample_index,
                      sample_data = index_d$sample_data,
                      m = index_d$dim_data[1],
                      n = index_d$dim_data[2],
                      data_id = index_d$data_id)
lapply(index_d2,head)
```

```
## $sample_data
## [1] 21.409480 33.740012  5.029540  2.955382  7.894554  9.510452
##
## $sample_index
## [1] 151 154  47 135 104  84
##
## $data_id
## [1] "exponential"
##
## $dim_data
## [1] 10 15
```

The created `index_data` object has four fields:

- `data_id` for the ID of the data as in `incremental_data`,
- `dim_data` for the dimension of the underlying data matrix,
- and `sample_index` and `sample_data` that indicate the index of the measurements and the readings in the cumulative matrix, obtained by calculating the cumulative sums of each column of the `incremental_data` matrix.

This is the primary data format used in Chapter 6. When this data format is used, most of the new algorithms in `meterModels` fully take advantage of it to estimate models efficiently.

9.1.3 `linear_measurement_data`: General linear measurements

A more general data format, `linear_measurement_data`, is available, for cases where the observations are linear measurements, but not necessarily temporal aggregates. The *masks* are provided through a `sparseMatrix` object (from a widely used `R` package called `Matrix`). To create a `linear_measurement_data` object, we provide the sparse matrix which tells the program how to measure the data, and an `incremental_data` object which is the matrix to measure.

```
# A measurement matrix that samples the first five entries of a matrix.
matrix_A = measurement_matrix(
  "matrix",
  matrix_A = Matrix::sparseMatrix(i = 1:5, j = 1:5, dims = c(5, m*n)))
lin_m_d = linear_measurement_data(
  x = matrix_A,
```

```
  data = incre_d)

print(cbind(incre_d$data[1:5], lin_m_d$sample_data))
```

```
## 5 x 2 Matrix of class "dgeMatrix"
##           [,1]      [,2]
## [1,] 1.8598281 1.8598281
## [2,] 4.8175843 4.8175843
## [3,] 1.4425791 1.4425791
## [4,] 0.2237663 0.2237663
## [5,] 3.1358933 3.1358933
```

9.1.4 exp_data

`exp_data` is a class to be used when running many algorithm runs, in an `experiment` (see below). It is created from an object of the class `incremental_data`, with some additional information as to which part of the data matrix to be used as training data and test data.

For example, the following code wraps `incre_d` into an object on which we will estimate models using the whole matrix as train data (the parameters `m_train` and `n_train` control this behavior).

```
# define the experiment data
exp_d = exp_data(x = incre_d, m_train = m, n_train = n)
```

9.2 matrix_model: Estimate a single model

`meterModels` has a number of matrix factorization algorithms included. Here we will use one of the proposed algorithms, `NeNMF`, which is an extension of [20].

To estimate a model, call the function `matrix_model` with the data object and the name of the algorithm, and a fixed rank. To see the estimated matrix, use `predict` on the estimated model.

```
nenmf_model = matrix_model(algorithm = "NeNMF", data = index_d, r = 2)
nenmf_pred = predict(nenmf_model, completion = TRUE)
```

We can plot the original and estimated model to see how well the matrix is recovered. In Figure 9.1, we can see that most of the original matrix is well estimated. There is some problem on the first column, probably due to the fact that only two temporal aggregates are observed on this column.

```
plot_matrices(list(
  original = example_matrix,
  nenmf_pred = nenmf_pred))
```

We can also call `matrix_model` with `incremental_data` objects, so that only to factorize the matrix. In this case, the estimated matrix is exactly the same as the original (Figure 9.2).

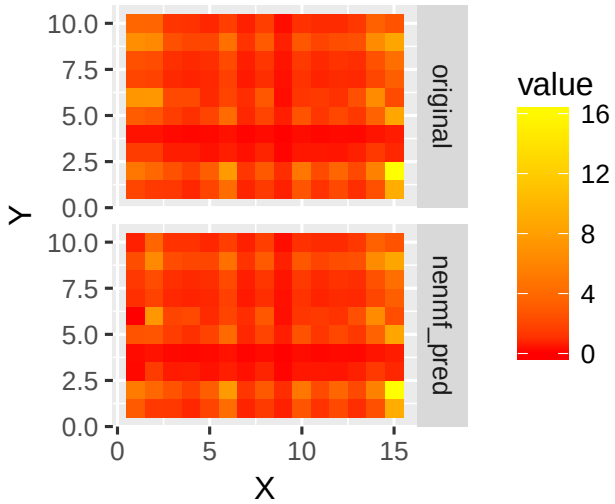


Figure 9.1: *The original matrix and the matrix estimated by* `NeNMF` *using 50% random temporal aggregates.*

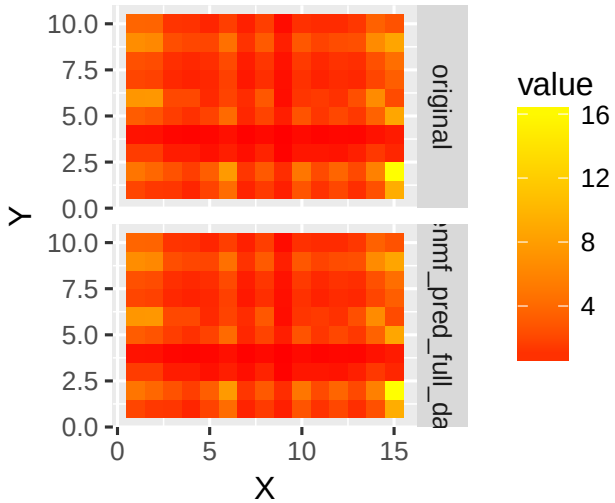


Figure 9.2: *The original matrix and the matrix estimated by* `NeNMF` *using full data.*

```
nenmf_model_full_data = matrix_model(algorithm = "NeNMF", data = incre_d, r = 2)
nenmf_factor_full_data = predict(nenmf_model_full_data, completion = FALSE)

plot_matrices(list(
  original = example_matrix,
  nenmf_pred_full_data = nenmf_factor_full_data))
```

In order to estimate an NMF with exogenous variables as side information (see Chapter 7), we use `HALS_model` as `algorithm`, and provide a `formula_params` object to specify which regression models to estimate:

```
regression_model_full_data = matrix_model(
  algorithm = "HALS_model", data = incre_d, r = 2,
  exo_variable = list(X = data.frame(variable = rnorm(dim(incre_d)[1]))),
  formula = list(X = formula_params(
    "s(variable)", # this is the right part of the formula
    HALS_model_params = list(reg_method = "gam"))))
```

Table 9.1: *Included algorithms in* `meterModels`

Algorithm	Type
HALS	Proposed method
NeNMF	Proposed method
HALSX1	Proposed method
HALS_model	Proposed method
softImpute	Reference method
HALSX2	Reference method
factor_gam	Reference method
grmf	Reference method
trmf	Reference method
individual_gam	Reference method
reduced_rank_regression	Reference method
empty_model	Reference method

Table 9.1 lists the algorithms included in `meterModels`. Among the reference methods, `softImpute`, `grmf`, `trmf`, and `reduced_rank_regression` are included by directly using the open-source programs of the respective authors. The other reference methods are in fact developed by us, although they are not the main propositions of this thesis. All of the proposed methods can be used jointly with the autocorrelation constraint, described in Chapter 6, even the ones proposed in Chapter 7.

9.3 Model evaluation

To evaluate obtained matrix factorization model, we can calculate the relative difference in a matrix norm (for example Frobenius norm), by using `relative_diff_norm`.

```
relative_diff_norm(matrix1 = incre_d$data, matrix2 = nenmf_pred, type = "F")
```

```
## [1] 0.258141
```

By using `evaluate_matrix_model`, we can evaluate the model using several metrics. The last two fields in the returned list below, `regression` and `prediction`, are reserved for regression models (Chapter 7), hence are `NA` here.

```
evaluate_matrix_model(nenmf_model, incre_d)
```

```
## $completion
##       Frob         ND      NRMSE       RMSE       MAPE
## 0.25814097 0.07393886 0.39200128 0.77130529 0.04642855
##
## $factorization
##       Frob         ND      NRMSE       RMSE       MAPE
## 0.25814137 0.07410952 0.39200188 0.77130647 0.04655324
##
```

```
## $regression
##  Frob    ND NRMSE  RMSE  MAPE
##    NA    NA    NA    NA    NA
##
## $prediction
##  Frob    ND NRMSE  RMSE  MAPE
##    NA    NA    NA    NA    NA
```

We can also check that the recovered matrix has almost the same measurements on the mask, by using `evaluate_on_mask`.

```
evaluate_on_mask(increData = incre_d$data, increEst = nenmf_pred,
                 known = index_d$sample_index)
```

```
## [1] 3.230599e-17
```

9.4 Run an experiment to estimate several models simultaneously

Many parameters are available for training a `matrix_model`. Therefore, we often need to try a number of combinations of the parameters, to find the best one. This is why we have the function `experiment`. To use `experiment`, we need to first define an `exp_data` object, and an experiment specification object (`exp_spec`).

```
# define the experiment specs
simple_spec = exp_spec(
  # try several ranks
  r = 2:5,
  # use NeNMF, HALS, and empty_model for constant interpolation
  algorithm = c("NeNMF", "HALS", "empty_model"),
  # save the models in a temporary directory
  save_model = "./exp_temp/",
  # use 3 parallel processes
  parallel_opts = list(mc.cores = 3),
  # we will only do nonnegative matrix factorization
  general_factorization = FALSE)
```

The estimated models are saved on disk at the path provided through `save_model` parameter in `simple_spec`. Here we are using 3 cores to run model training in parallel (`parallel_ops`). We are only going to do nonnegative matrix factorization (`general_factorization = FALSE`).

We launch the experiment by calling `experiment`, with the experimental data, the sampling scheme, and the experimentation specification.

```
# then we are good to go
simple_exp = experiment(exp_data = exp_d,
                        sampling_scheme = sample_incre_d_at,
                        exp_spec = simple_spec)
```

Table 9.2: *Part of the evaluation table generated by* `experiment`.

r	algorithm	completion.Frob	factorization.Frob	time_spent
2	NeNMF	0.2499	0.2499	0.6408
3	NeNMF	0.3899	0.3899	0.8724
4	NeNMF	0.4723	0.4723	0.5726
5	NeNMF	0.5118	0.5118	0.5602
2	HALS	0.1961	0.1961	0.6340
3	HALS	0.3137	0.3137	0.8288
4	HALS	0.4015	0.4015	0.4019
5	HALS	0.4470	0.4470	0.3711
NA	empty_model	0.6415	NA	0.0089

Figure 9.3: *Comparison of the models estimated by calling* `experiment`

The `experiment` function returns a table with the evaluation metrics of the estimated models (Table 9.2). It looks like the `HALS` algorithm with rank fixed at 2 obtains the best model (smallest completion error). This is not surprising, since the data is a simulated matrix exactly of rank 2.

Using Table 9.2, we can generate plots similar to those reported in the last two chapters (Figure 9.3).

9.5 Distributed computation in model estimation

The NMF algorithms introduced in this thesis are intrinsically iterative[1]. Given the format of the observations (either temporal aggregates or general linear measurements), distributed stochastic gradient descent algorithms proposed in matrix completion and recommendation systems are not easily adaptable to the problem [131–134].

The parallel solution that we finally developped is based on the CCD++ algorithm [135]. The CCD++ algorithm is mostly a parallel HALS, which is discussed in previous chapters. When updating the columns of $\mathbf{F_r}$ and $\mathbf{F_c}$, this algorithm updates the coefficients in these columns in parallel. In order to combine this algorithm with the simplex projection and autocorrelation penalization discussed in Chapter 6, this algorithm was re-implemented in `Rcpp`. Using `openMP`, both the update of factors and the projection into the simplex satisfying the linear measurement

[1]The distributed computation in model estimation is implemented by Gustavo Castro during his internship at EDF Lab Paris-Saclay.

constraint are distributed through the multiple cores.

In `meterModels`, this parallelization can be used agnostically (not available with regression models). To choose to use this functionality, the user can use the `opts` parameter in `matrix_model`.

Part III

Perspectives

Chapter

10 Perspectives

In this thesis, we studied methods to model multivariate electricity consumption time series from partial data. We considered the problem from several aspects, with either temporally aggregated or individually aggregated data, either to estimate past consumption or to predict future consumption. We studied solutions from three contexts: spatio-temporal statistics, clustering of individuals using socio-demographic information, and NMF-based methods. In NMF, we extended the literature to have solutions that are adapted to our problem, and proposed improvements to better achieve the estimation goals. Particularly, we proposed to take into account additional information such as temporal autocorrelation and exogenous variables in NMF, and deduced algorithms to solve these extended models. The proposed methods were tested on synthetic and real datasets, and showed promising results.

Perspective applications in electricity

In this thesis, empirical studies were carried out to evaluate the estimation and prediction performance of the matrix factorization methods. In these empirical studies, the methods are evaluated by examining the distance between the prediction or recovery results with the original data, under certain metrics, typically the Frobenius norm. It will be interesting to carry out studies that are more directly connected to application cases in electricity generation and consumption. That is, there can be statistics that are more important for the electric network, and thus it will be more insightful to examine the methods using metrics that are directly linked to these important statistics.

For example, one potential application is to use the matrix factorization method to estimate the daily peak in power in local transformers from residential smart meter data. The setting is the following: smart meters can provide daily electricity consumptions to the distribution system operator (DSO) with individual offsets. The DSO can then run the algorithms proposed in this thesis to estimate the individual electricity consumption at a much finer scale (10-minute interval for example). By aggregating the estimated fine-scale consumption time series at each transformer, the DSO has an estimation to the moment of the daily peak and the level of the peak. Preliminary studies showed this approach to be promising. Perspectives in this application include fine tuning the evaluation metrics so as to reflect the real concern of the DSO. Moreover, to better adapt the methods of this thesis to this application, it can be interesting to change the objective function in the optimization problem to directly estimate the peak.

Methodological perspectives

Estimation with both spatial and temporal aggregates The empirical studies in this thesis used either spatial (in Chapters 3 and 4) or temporal aggregates (in Chapters 6 and 7).

However, the algorithm proposed in Chapter 7 can in fact use general linear measurements.

It would then be interesting to combine both partial observations together in one empirical study to evaluate the potential gain. In terms of electricity applications, we can potentially improve the individual consumption estimation, by using not only temporal aggregates but also temporally fine-scaled measures on the network, such as measured from the distribution network.

Given that the spatially and temporally aggregated data come from different sources, these two parts of data might be incoherent. It would be important to relax the linear equality constraint to deal with this situation.

Usage of social network data It is also possible to generalize the approach to social network data. The socio-demographic information for individuals considered in this thesis is in the form of individual features, such as client type, age, and occupation. It can be interesting to use a form of social networks, which encode the links between individuals. Social networks are important in recommendation systems, as people who are connected over a social network tend to have similar tastes. In electricity consumption, a similar mechanism might take place.

Correlation between the data and the measurement operator In most matrix sensing studies, it is supposed that the measurement operator is independent of the matrix. In matrix completion, the standard setting is that the observation pattern is uniformly random over all matrix entries. When the matrix to be completed is also considered to be a realization of a random distribution, it is often supposed to be independent from the observation.

However, this hypothesis is rarely verified. In a recommendation application, for example the MovieLens100k application considered in Chapter 7, it is often the case that users would give ratings to a number of correlated items. Moreover, users tend to try items on which they have an prior positive opinion. Therefore, the observation pattern should be correlated with the matrix to be completed. In the electricity consumption estimation problem, the correlation could be of following forms:

- Temporally aggregated data: particularly with periodic observations, if the observation period is a multiple of one of the intrinsic periods of the data, there might be interference. Variations of a frequency similar to that of the observations can be masked. Empirically, we observed in Chapter 6 that periodic observations have lower accuracy. This result could stem from the correlation between the observation patterns and the data.

- Spatially aggregated data: correlated individuals can be clustered spatially. If the observations are spatially aggregated consumption over similar individuals, and the estimation does not take this correlation into account, the total variance can be overestimated by the model.

Deep learning with partial data In the end of Chapter 8, we discussed briefly that nonnegative matrix factorization is a very shallow neural network with ReLU activation function. This fact thus raises the question whether the methods developed in this thesis can be generalized to more complex neural network architectures, or inversely, if deep learning can be modified to use partial data such as those considered in this thesis. There has been some interest in using deep learning jointly with compressive sensing techniques in the image processing community recently [136–138]. One important challenge in generalizing from matrix sensing using deep learning lies in finding a good representation for the observations which is compatible to the neural network architecture and computation methods.

Theoretical perspectives

We have proposed algorithms that are convergent to stationary points. As mentioned in Chapter 5, new results are emerging in global optimum guarantees for non-convex problems. In general, NMF is an NP-hard problem, and therefore, there can not be general guarantees. Recent results have shown that alternating direction algorithms can have global guarantees when one of the factors is initialized close to the ground truth [91, 98]. It would be interesting to see if for special classes of matrices, similar guarantees could be obtained without initialization conditions.

Appendices

Appendix

A Reference manual of package 'meterModels'

Type Package

Title Do nonnegative matrix factorization on meter data

Version 0.4.2

Date 2017-05-26

Description Nonnegative matrix factorization on meter data

License All rights reserved (EDF Lab).

LazyData true

VignetteBuilder knitr

RoxygenNote 6.0.1.9000

Suggests testthat,
R39Toolbox,
ggplot2,
dplyr,
knitr,
rmarkdown,
cowplot,
rARPACK,
recosystem,
softImpute,
caret,
gbm,
kernlab,
rrr,
RcppOctave,
mgcv,
KANN

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp,
reshape2,
Matrix,
MASS,
data.table(>= 1.10.4-3)

Depends parallel

R topics documented:

`aggregate_max` *Calculate the maximum at a certain time aggregation level and individual aggregation level*

Description

Calculate the maximum at a certain time aggregation level and individual aggregation level

Usage

```
aggregate_max(data_matrix, aggregate_vector = 1:nrow(data_matrix), timestep)

aggregate_which.max(data_matrix, aggregate_vector = 1:nrow(data_matrix),
  timestep)
```

Arguments

`data_matrix` the matrix data on which to do the computation

`aggregate_vector` an integer vector of same length as `nrow(data_matrix)`. Specifies the group in which each column is to be aggregated.

`timestep` an integer. If timestep = n, then n values in `data_matrix` is to be aggregated to one single value. For example, if `data_matrix` has hourly data on each row, and `timestep == 24`, the function calculates daily maximum for each aggregate group.

Functions

- `aggregate_which.max`: the same thing but with which.max

`aggregateData` *Aggregate data to desired time scale*

Description

Aggregate data to the desired time scale, using selected function

Usage

```
aggregateData(originalData, timestep, fun, ...)
```

Arguments

`originalData`	the original data to be aggregated
`timestep`	an integer. If timestep = n, then n values in `originalData` is to be aggregated to one single value.
`fun`	the function to be used to aggregate. For example, for averages, `mean` can be used.
`...`	additional parameters for `fun`

Value

the temporally aggregated data for one matrix

`as.sparseMatrix` *Cast an object into a sparse matrix of the* `Matrix` *package*

Description

Currently, `as.sparseMatrix.index_data` is implemented. The function casts an object into a sparse matrix which contains the cumulative sums. For an `index_data` object, it returns a matrix of dimension `c((x$dim_data)[1]+1, (x$dim_data)[2])`, which only has values at places where indexes are observed.

Usage

```
as.sparseMatrix(x, ...)

## S3 method for class 'index_data'
as.sparseMatrix(x, ...)
```

Arguments

`x`	an `index_data` object.
`...`	not used.

`boundary_close_exogeneous_variables` *Sample a non-negative separable matrix.*

Description

Functions to simulate factor matrices verifying uniqueness constraints of NMF.

Usage

```
boundary_close_exogeneous_variables(M, K)

separable_matrix(M, K, sparsity = 0)

boundary_close_matrix(M, K, sparsity = 0)
```

Arguments

`M, K`	the dimension of the matrix to simulate.
`sparsity`	the parameter that regulates the sparsity of the matrix. If `sparsity > 0`, `sparsity` fraction of coefficients in the non-separable/non-boundary close rows are set to 0.

Details

`boundary_close_exogeneous_variables` creates an exogeneous variable matrix for creating a boundary close matrix.

`separable_matrix` simulates a separable matrix.

`boundary_close_matrix` samples a matrix with boundary close columns.

Value

a `M`-by-K matrix separable.

References

For more details on the definition of these matrices, see:

- Laurberg, Hans, Mads Grasboll Christensen, Mark D. Plumbley, Lars Kai Hansen, and Soren Holdt Jensen. *Theorems on Positive Data: On the Uniqueness of NMF.* Computational Intelligence and Neuroscience 2008. doi:10.1155/2008/764206.
- Mei, Jiali, Yohann De Castro, Yannig Goude, Georges Hebrail, and Jean-Marc Azais. *Nonnegative Matrix Factorization with Side Information for Time Series Recovery and Prediction.* Submitted, 2017.

`calculate_linkage`	*Calculates the linkage vector used to* `increment_SPaggregate_evaluate`.

Description

Calculates the linkage vector used to `increment_SPaggregate_evaluate`.

Usage

```
calculate_linkage(entity_names, mode, sep = "_")
```

Arguments

`entity_names`	the entity names to be split.
`mode`	either 1 or 2. If `mode = 1`, the entity names are split at `sep`. If `mode = 2`, the first five characters are used as aggregation entities.
`sep`	the separator character in `entity_names`.

Details

The entity names are supposed to be

- either in the form `"xxxSyyy"` where `"xxx"` the higher level entity name, `"S"` is a separator character (parameter `sep`) and `"yyy"` is the lower level entity name.
- or in the form `"xxxxxyyy"` where the first five characters are the higher level entity name.

Value

an integer vector of the same length as `entity_names` specifying how to aggregate the entities.

This can be used either in aggregated evaluation functions such as `increment_SPaggregate_evaluate` or as an intrinsic property of a dataset so defined while creating an `incremental_data` object.

`compare_two_matrix` *Calculate the deviation between the data matrix and the estimation matrix*

Description

Calculate the deviation between the data matrix and the estimation matrix

Usage

```
compare_two_matrix(data, est, metrics)

implemented_matrix_comparison_metrics()
```

Arguments

`data, est`	The two matrices to be compared
`metrics`	A character vector specifying the metrics to be used. Should be a subset of `implemented_matrix_comparison_metrics`.

Value

A named vector of the distance/deviation between the two matrices.

Examples

```
compare_two_matrix(example_matrix, jitter(example_matrix),
                   implemented_matrix_comparison_metrics())
```

`create_formula_string` *Create character formula from a vector of variable names*

Description

This function creates a string which can be later transformed to the right part of an `R formula`. It is primarily used to create `formula_params` objects to be used in `regression_models`.

Usage

```
create_formula_string(var_names, type = "gam", k = 15, numeric)
```

Arguments

`var_names`	The vector of variables names to be put in the formula
`type`	Either `"gam"` for smooths or anything else for a `lm` kind of formula.
`k`	The dimension of the spline basis when using `"gam"`.
`numeric`	A vector of the same length as `var_names` specifying if the variables are `numeric`. If not supplied, all variables are supposed to be `numeric`. If this is not the case, then there will be problems with smooth terms in estimation.

See Also

`formula_params` and `regresion_model`

Examples

```
create_formula_string(LETTERS[1:5])
create_formula_string(LETTERS[1:5], numeric = c(FALSE, FALSE, TRUE, TRUE, TRUE))
create_formula_string(LETTERS[1:5], type = "lm")
```

`empty_formula` *Create an empty formula to be used as a placeholder*

Description

This object is to be used when no regression componnent is to be estimated. It is basically an empty `formula_params`.

Usage

```
empty_formula()

is.empty_formula(x)
```

Value

a `formula_params` object that is also an `empty_formula` object.

`empty_model` *A naive matrix recovery model*

Description

Basically, this is the most naive matrix recovery model, without the low rank hypothesis.

Usage

```
empty_model(x, rho = NULL, lambda = 0, ...)

is.empty_model(x)
```

Arguments

`x`	an object of the class `index_data`, `incremental_data`, or `linear_measurement_data`.
`rho`	either `NULL` or a vector of length `n`. When it is provided, it is the vector of temporal autocorrelation threshold to be imposed on each of the columns.
`lambda`	either a number which is 0 or positive, or `"optimal"`. This is the penalization parameter to be used to impose autocorrelation constraint.
`...`	additional parameters to pass on to the estimation routine.

Details

When the data are temporal aggregates (`index_data`), constant interpolation is returned. If `rho` and `lambda` is provided, the projection of 0 matrix with autocorrelation penalization is returned.

When the data are general linear measurements, the orthogonal projection of 0 matrix into the linear subspace of matrices satisfying the data constraints is returned. For example, for matrix completion, this is equivalent to replacing every missing value with 0.

Value

an `empty_model` which is a `matrix_model` with minimal fields.

Examples

```
model = empty_model(x = example_index_data)
model$M
model = empty_model(x = example_linear_measurement_data)
model$M

pred_empty_model = predict(model, completion = TRUE)

# empty_model cannot predict with completion=FALSE, since it's not
# a factorization_model
is.null(predict(model, completion = FALSE))
```

`evaluate_matrix_model` *Evaluate a* `matrix_model`

Description

Given `matrix_model`, this function calculates a number of performance metrics on the precision of matrix recovery and prediction.

Usage

```
evaluate_matrix_model(model, incre_data, aggreg = FALSE,
  prediction_type = implemented_prediction_types(),
  metrics = implemented_matrix_comparison_metrics(), newdata = NULL)
```

Arguments

`model`	a `matrix_model` object
`incre_data`	the `incremental_data` object on which the to evaluate the performance of the model.
`aggreg`	should the data be aggregated to be evaluated?
`prediction_type`	Which `predict` methods to use in the evaluation? This character vector should be a subset of `implemented_prediction_types()`.
`metrics`	Which metrics to use in the evaluation? This character vector should be a subset of `implemented_matrix_comparison_metrics()`.
`newdata`	When evaluating a `regression_model`, provide new data through this variable to test the prediction of the `regression_model`. As in `predict.regression_model`, if supplied, this should be a named list with fields `X` and/or `Y`.

Value

a named list. Each element of the list is a named vector of the evaluation `metrics` for one of the `prediction_type`.

See Also

`implemented_prediction_types` and `implemented_matrix_comparison_metrics` for details of the different types of evaluation value possible.

Examples

```
fac_mod = factorization_model(example_index_data, r = 2, algorithm = "HALS")
# For a factorization_model, regression and prediction errors are NA
evaluate_matrix_model(fac_mod, example_incremental_data)
```

```
reg_mod = regression_model(x = example_index_data,
                           exo_variable = example_exogenous_variables,
                           formula = example_formula,
                           r = 2,
                           algorithm = "HALS_model")
# For a regression_model, is no newdata is supplied,
# the prediction error is calculated on original data, hence equal to
# regression error.
evaluate_matrix_model(reg_mod, example_incremental_data)

evaluate_matrix_model(reg_mod, example_incremental_data,
                      newdata = example_exogenous_variables)
```

`example_matrix` *Example data structures*

Description

Example data structures

Usage

```
example_matrix

example_sampling_scheme

example_index_data

example_incremental_data

example_linear_measurement_data

example_exogenous_variables

example_formula

example_exp_data
```

Format

An object of class `matrix` with 13 rows and 11 columns.

`exp_data` *Create a set of data to be used in an experiment*

Description

The object of the class `exp_data` is to be used in experiments with many parameters. Therefore, specific information concerning the data is to be passed through this function.

Usage

```
exp_data(x, m_rho, m_train, m_test, n_train, n_test, formula)

is.exp_data(x)

## S3 method for class 'incremental_data'
exp_data(x, m_rho = 0, m_train, m_test, n_train,
  n_test, formula = NULL)

## S3 method for class 'linear_measurement_data'
exp_data(x, m_rho = 0, m_train, m_test,
  n_train, n_test, formula = NULL)
```

Arguments

`x`	the data object to be wrapped into an `exp_data`. Either of class `incremental_data` or `linear_measurement_data`.
`m_rho`	parameter to specify which parts of the data to be used as "historical data". The rows `1:m_rho` for calculating temporal autocorrelation and excluded for other use (model training and testing).
`m_train, m_test, n_train, n_test`	The rows `1:m_train + m_rho` of the data is to be used as training data. The rows `1:m_test + m_rho+m_train` is to be used as test data (when regression models are estimated). The same for `n_train` and `n_test` for the columns.
`formula`	a list of formula and additional parameters to pass on to specify regression models. Several pairs of formulae can be specified simultaneously and will result in separate model estimations by `experiment`. Each element of `formula` is of the format `list(X = formula_params(""), Y = formula_params(""))`. The fields `X` and `Y` are for row and column features respectively.

Value

an `exp_data` object.

See Also

`formula_params` for specification of the regression formulae, `experiment` for using an `exp_data` in action.

Examples

```
# create exp_data with incremental_data
example_exp_data = exp_data(example_incremental_data, m_rho = 0,
                            m_train = 2, n_train = 2,
                            formula = example_formula)

# create exp_data with linear_measurement_data
example_exp_data = exp_data(example_linear_measurement_data, m_rho = 0,
                            m_train = 2, n_train = 2,
                            formula = example_formula)
```

`exp_spec` *Create an object specifying parameters of an experiment*

Description

Create an object specifying parameters of an experiment

Usage

```
exp_spec(r, algorithm = implemented_algorithms("all"), random_start = 3,
  lambda = c(0, 1, "optimal"), general_factorization = c(TRUE, FALSE),
  parallel_opts = list(mc.cores = 1, mc.preschedule = FALSE), opts = list(),
  save_model = file.path(getwd(), "exp_tmp"), eval_only = FALSE)

is.exp_spec(x)
```

Arguments

`r, algorithm, lambda, general_factorization`
: The same parameters as in `factorization_model`, except they can be vectors so that a combination of all parameters are tested.

`random_start`
: an integer to specify the number of random runs to turn for each experiment spec. When using `parallel` to parallelize, these random runs with one single experiment spec will be run on the same core.

`parallel_opts`
: a list to specify how to parallelize the experiments. It has two fields. `mc.cores`: the number of cores to use and `mc.preschedule` should the parallelization be prescheduled. See `parallel::mclapply` for more details.

`opts`
: a named list to set the parameters for the estimation algorithm with the following fields:

- `tol`: tolerance for deciding whether to stop.
- `maxit`: the maximal number of iterations.
- `minit`: the minimal number of iterations.
- `verbose`: whether to provide more reporting in the computation or not.
- `fixed_X`, `fixed_Y`: whether `X` or `Y` should be fixed in the iterations.
- `maxtime`: maximal computing time allocated in seconds.
- `ITER_MAX`, `ITER_MIN`: the maximal and minimal number of iterations for NeNMF.
- `X0`,`Y0`: initial values for `X`, `Y`
- `FuninR`: If `FuninR = "R"` use the R version of subroutines. If `FuninR = "cpp"` use the C++ version. Default is `"cpp"`.
- `ccdppitmax`,`ccdppthreads`,`ccdpplambda`: the maximal number of iterations, the number of threads, and the penalization parameter for CCDPP algorithm.

`save_model` a character string specifying where to save the estimated models.

`eval_only` Should we estimate (potentially) new models or only evaluate old models saved in `save_model`.

Value

An object of class `exp_spec` which wraps the provided parameters AND an `exp_grid` in which each row is a model to be estimated. Incompatible parameter combinations are removed by an internal function `simplify_exp_grid`.

See Also

`experiment` to run experiments.

Examples

```
ex_sp = exp_spec(r = 2:10)
```

`experiment` *The main experiment function*

Description

This function is the main function to use to run a series of experiments.

Usage

```
experiment(exp_data, sampling_scheme, exp_spec, verbose = TRUE)

evaluate_past_experiement(exp_data, sampling_scheme, exp_spec,
  verbose = FALSE)
```

Arguments

`exp_data` An `exp_data` object.

`sampling_scheme` A `sampling_scheme` object.

`exp_spec` An `exp_spec` object.

`verbose` Whether messages reporting the start and the end of each model estimation should be printed in stdout.

Value

In a typical experiment run, this function saves all estimated models and the model estimation result tables on the disk.

It also returns a data.frame of standard model evaluation metrics. Apart from information on the models that are estimated, the output table contains outputs of the internal function `evaluate_output_list` applied on the train data, the column test data, the row test data, and the column-row test data. The latter three are present whenever they are applicable: with `m_test`, `n_test` strictly larger than 0, in presence of exogenous variables, and with algorithms which estimate regression_models.

See Also

`evaluate_output_list` to know more about the output evaluation metrics.

Examples

```
ex_sp = exp_spec(r = 2:3, algorithm = c("NeNMF", "HALS", "HALS_model"),
                 lambda = 0,
                 random_start = 2, general_factorization = FALSE)
## Not run:
example_exp = experiment(example_exp_data, example_sampling_scheme, ex_sp)
example_exp = find_optimal_rank(example_exp)

## End(Not run)
```

`factorization_model` *Fit a factorization model*

Description

Fit a factorization model with index data or incremental data

Usage

```
factorization_model(x, r, algorithm = implemented_algorithms("factorization"),
  opts = list(), rho = NULL, lambda = 0, ground_truth = NULL,
  NNDSVD = FALSE, save_iterations = FALSE, general_factorization = FALSE,
  ...)

is.factorization_model(x)

## S3 method for class 'index_data'
factorization_model(x, r,
  algorithm = implemented_algorithms("factorization"), opts = list(),
  rho = NULL, lambda = 0, ground_truth = NULL, NNDSVD = FALSE,
  save_iterations = FALSE, general_factorization = FALSE, ...)

## S3 method for class 'incremental_data'
factorization_model(x, r,
  algorithm = implemented_algorithms("factorization"), opts = list(),
  rho = NULL, lambda = 0, ground_truth = NULL, NNDSVD = FALSE,
  save_iterations = FALSE, general_factorization = FALSE, ...)

## S3 method for class 'linear_measurement_data'
factorization_model(x, r,
  algorithm = implemented_algorithms("linear_measurement_data"),
  opts = list(), rho = NULL, lambda = 0, ground_truth = NULL,
  NNDSVD = FALSE, save_iterations = FALSE, general_factorization = FALSE,
  formula = NULL, ...)

## S3 method for class 'factorization_model'
dim(x)
```

Arguments

`x` an object of the class `index_data`, `incremental_data`, or `linear_measurement_data`.

`r` integer. the target rank of the matrix.

`algorithm` a character string with the name of the algorithm to be used. See `implemented_algorithms` for the complete list.

`opts` a named list to set the parameters for the estimation algorithm with the following fields:

- `tol`: tolerance for deciding whether to stop.
- `maxit`: the maximal number of iterations.
- `minit`: the minimal number of iterations.
- `verbose`: whether to provide more reporting in the computation or not.
- `fixed_X`, `fixed_Y`: whether `X` or `Y` should be fixed in the iterations.
- `maxtime`: maximal computing time allocated in seconds.
- `ITER_MAX`, `ITER_MIN`: the maximal and minimal number of iterations for NeNMF.
- `X0`,`Y0`: initial values for `X`, `Y`

- `FuninR`: If `FuninR = "R"` use the R version of subroutines. If `FuninR = "cpp"` use the C++ version. Default is `"cpp"`.
- `ccdppitmax`,`ccdppthreads`,`ccdpplambda`: the maximal number of iterations, the number of threads, and the penalization parameter for CCDPP algorithm.

`rho` either `NULL` or a vector of length `n`. When it is provided, it is the vector of temporal autocorrelation threshold to be imposed on each of the columns.

`lambda` either a number which is 0 or positive, or `"optimal"`. This is the penalization parameter to be used to impose autocorrelation constraint.

`ground_truth` the ground truth matrix of incremental data.

`NNDSVD` a logical to specify whether use SVD to initialize.

`save_iterations` whether to save the factorizations during iterations.

`general_factorization` If `TRUE` does a general factorization without nonnegativity. If `FALSE`, which is the default value, does an NMF. IF is the string `"random"`, randomly allow half of the iterations to explore negative values.

`...` additional parameters to pass on to the estimation routine.

`formula` A list of two fields named `X` and `Y`. Both are to be objects of the class `formula_params`.

Value

a `factorization_model` object. It has all fields of `matrix_model`, and also the following additional ones:

- `X`,`Y`: the two factors.
- `hist_obj`, `hist_prjg`, `hist_error`, `hist_time`: reporting values throughout the iterations.
- `iter`: number of iterations used.
- `niter`: number of inner iterations using NeNMF algorithm.
- `convergence`: whether the algorithm arrived at convergence or the maximal number of iterations is attained.
- `X_iterations`, `Y_iterations`: the factors throughout the iterations. They are `NULL` if `save_iterations = FALSE`.
- other fields which are only applicable to `regression_models`.

Examples

```
fac_mod = factorization_model(example_index_data, r = 2, algorithm = "HALS")

# predict with completion = TRUE or FALSE
predict(fac_mod, completion = TRUE)
predict(fac_mod, completion = FALSE)

# plot the model
plot(fac_mod)
```

`find_optimal_rank` *Post-processing of experiment outputs*

Description

These functions help post process experiment outputs.

Usage

```
find_optimal_rank(output_table)

duplicate_complete_sampling_results(output_table)

transform_formula_to_factors(output_table)
```

Arguments

`output_table`	an output_table returned by `experiement`, or obtained by merging several these outputs.

Details

`find_optimal_rank` finds the optimal rank for each combination of all other experiment parameters.

`duplicate_complete_sampling_results` duplicates the rows which are estimated using complete matrix observations in the table returned by `experiment` and changes the `sampling_type` value to `"periodic"` and `"random"`. This is to help us plot error rates depending on the sampling rate.

`transform_formula_to_factors` Transform formulas in the output table into factors so that it would be easy to produce graphics.

Value

a post-processed output table.

See Also

`experiment` for examples.

finds_permutation *Evaluate a list of* matrix_models

Description

Internal functions used in `experiment` to report estimated model performance.

Usage

```
finds_permutation(correlation_matrix)

compare_nmf_factors(X, X_prime,
  metrics = implemented_matrix_comparison_metrics()[1])

compare_two_outputs(output1, output2, known = NULL, perm = FALSE,
  matrix_A = NULL)

how_many_stationary_points(in_between, rank)

evaluate_output_list(these_models, increData, aggreg, verbose = FALSE,
  prediction_type = implemented_prediction_types(),
  metrics = implemented_matrix_comparison_metrics(), newdata = NULL)
```

Arguments

`correlation_matrix`	the correlation matrix between two factors matrices of the same size used to find the permutation of factors which minimizes their distance.
`X`	the first factor matrix.
`X_prime`	the second factor matrix.
`metrics`	Which metrics to use in the evaluation? This character vector should be a subset of `implemented_matrix_comparison_metrics()`.
`output1, output2`	the two outputs to compare
`known`	extra set of observation index, if we want to evaluate the two on mask with a different one than that included in `output1`.
`perm`	whether to output the found permutation or not.
`in_between`	the pairwise distance between the models.
`rank`	the rank of the models
`these_models`	the list of models.
`increData`	the `incremental_data` against with to evaluate model predictions.
`aggreg`	whether to calculate on an aggregation scale.
`verbose`	wheter to print the results.

`prediction_type`
: Which `predict` methods to use in the evaluation? This character vector should be a subset of `implemented_prediction_types()`.

`newdata`
: When evaluating a `regression_model`, provide new data through this variable to test the prediction of the `regression_model`. As in `predict.regression_model`, if supplied, this should be a named list with fields `X` and/or `Y`.

Details

For a list of model estimated with the same parameters, `evaluate_output_list` from different random initializations, this function outputs the average performance metrics for matrix recovery and prediction. `evaluation_matrix_model` uses this function to actually calculate the error rates.

When evaluating on train data, `evaluate_output_list` also calculates several metrics which evaluates the distance between the models estimated from different random initialization. This is done by calling `compare_two_outputs` and `how_many_stationary_points`, which compare both the matrix recovery and the factors, and estimate the number of distinct stationary points in the list. Even more internally, they call `finds_permutation` and `compare_nmf_factors` to permute the factors and actually compare them.

`generate_exo_mat` *Generate some random features*

Description

This is mostly used in development and for creating simulations.

Usage

```
generate_exo_mat(m, d, k = 3)
```

Arguments

`m` number of rows in the exogenous variable table.

`d` number of variables in the exogenous variable table.

`k` dimension of the spline basis to be used to generate the features.

Value

A named list with

- `exo_matrix`: the variables
- `features`: the transformed variables
- `formula_alone`: the formula used to transform variables
- `formula_caret`: `formula_alone` without the splines

`get_data_test` *Subsample or aggregate the data matrix to desired frequency and dimension*

Description

This function subsamples or aggregates a data matrix whose columns are time series to a desired frequency. It can also calculates the cumulative sums of the columns and/or normalizes the columns to have mean 1.

Usage

```
get_data_test(data_matrix, base_freq = 1, row_number = NULL,
  normalize = TRUE, cumul = TRUE, fun = mean)
```

Arguments

`data_matrix`	a numeric matrix.
`base_freq`	the frequency desired.
`row_number`	the intended row_number.
`normalize`	whether the data is to be normalized
`cumul`	whether the cumulative index is to be returned
`fun`	the aggregation function to be used. Default is `mean`.

Details

NAs are removed in calculating the aggregation values.

Value

a matrix of dimension `row_number`-by-`ncol(data_matrix)`.

Examples

```
## Not run:
m = 10000
n = 10000
r = 2
example_matrix = matrix(rexp(m * r), ncol = r) %*%
  matrix(rexp(n * r), ncol = n)


system.time(get_data_test(example_matrix, 100))
system.time(aggregateData(example_matrix, 100, mean))

## End(Not run)
```

get_formula_param *Retrieves parameters from a* formula_params

Description

Retrieves parameters from a `formula_params`

Usage

```
get_formula_param(x, what)
```

Arguments

`x`	the `formula_params` from which to get parameters.
`what`	the name of the parameters to be retrieved.

Value

a vector of parameters which match `what`. `NULL` is returned if nothing is found.

implemented_algorithms *The implemented algorithms*

Description

Lists the algorithms using keywords.

Usage

```
implemented_algorithms(...)
```

Details

As parameters, provide a subset of comma-separated character strings of the following keywords:

- `"octave"`: imported methods from octave (`"grmf"` and `"trmf"`)
- `"factorization"`: matrix factorization methods without regression
- `"autocorrelation"`: algorithms that can be used jointly with autocorrelation.
- `"regression"`: regression-enhenced algorithms
- `"benchmark"`: benchmarks

- "partial_data": algorithms that can be used with partial data.
- "completion": algorithms that can be used to do matrix completion.
- "factorized": all matrix factorization algorithms, including regression-enhenced ones.
- "linear_measurement_data": algorithms that can be used with objects of the class linear_measurement_data.
- "index_data": algorithms that can be used with objects of the class index_data.
- "proposed": algorithms proposed by the thesis.
- "reference": everything except "proposed"
- "all": everything.

When multiple parameters are put in, the intersection of the sets are returned. Default is "all".

Examples

```
implemented_algorithms()
implemented_algorithms("reference","factorized")
```

---

implemented_prediction_types

The prediction types

Description

The prediction types

Usage

```
implemented_prediction_types()
```

Details

For object of class matrix_model, whenever applicable, the four implemented types are

- "factorization": object$X %*% object$Y.
- "completion": object$X %*% object$Y projected into the subspace satisfying the data constraint.
- "regression": predict(object$models$X) %*% predict(object$models$Y).
- "factorization": same as "regression", but with new data.

Examples

```
implemented_prediction_types()
```

implemented_sampling_type

The implemented sampling types

Description

Lists the implemented sampling types for each type of data

Usage

```
implemented_sampling_type(type = "index_data")
```

Arguments

`type`	either `"index_data"` or `"linear_measurement_data"`.

Value

a vector of the sampling types.

Examples

```
implemented_sampling_type("index_data")
```

increment_evaluate

Evaluates a matrix estimation against the data matrix.

Description

Evaluates a matrix estimation against the data matrix.

Usage

```
increment_evaluate(increData, increEst, period = 1, lissage = NULL,
  cumData = NULL, cumEst = NULL,
  metrics = implemented_matrix_comparison_metrics())

increment_SPaggregate_evaluate(linkage_vector, increData, increEst,
  period = 1, lissage = NULL, cumData = NULL, cumEst = NULL,
  metrics = implemented_matrix_comparison_metrics())

evaluate_on_mask(increEst, cumData = NULL, increData = NULL, obs = NULL,
  known, matrix_A = NULL)
```

Arguments

`increData, obs`
: either the incremental data matrix `increData` or the observations on the data matrix `obs`. One of `cumData`, `increData`, and `obs` has to be supplied for `evaluate_on_mask`.

`period`
: interger. he aggregation scale.

`lissage`
: a number between 0 and 1. The smoothing parameter.

`cumData, increData`
: the matrix of cumulative or incremental data. One of these must be supplied. If both supplied, will use `increData`.

`cumEst, increEst`
: the matrix of cumulative or incremental estimation. One of these must be supplied. If both supplied, will use `increEst`.

`metrics`
: a character vector specifying the comparison metrics to be calculated.

`linkage_vector`
: integer vector specifying how to aggregate columns of estimation and data matrix.

`known`
: the index of the observations on the matrix. Used in `evaluate_on_mask`.

`cumEst`
: the matrix of cumulative estimated consumption

`incremental_data` *Put the data matrix in form*

Description

Put the data matrix in form

Usage

```
incremental_data(x, exo_variable = NULL, aggreg = NULL, data_id)

## S3 method for class 'incremental_data'
dim(x)
```

Arguments

`x`
: the `numeraic` data matrix.

`exo_variable`
: exogeneous variables associated with the data matrix. Supposed to be a named list with (optional) fields `X` and `Y`.

`aggreg`
: a numeric vector specifying how to aggregate the data matrix. Must be of length `ncol{x}`. This is optional.

Examples

```
incre_data_without_exo_variable = incremental_data(
    x = example_matrix,
    data_id = "exponential")
incre_data_with_exo_variable = incremental_data(
    x = example_matrix,
    data_id = "exponential",
    exo_variable = example_exogenous_variables)
```

---

ind2sub — *Translate between indexes and subscripts*

Description

Translate between indexes and subscripts

Usage

```
ind2sub(m, ind)

sub2ind(m, r, c)
```

Arguments

`m`	integer. The number of rows in the matrix
`ind`	vector of integers. The indexes to be translated.
`r, c`	the integer vectors of row and column subscripts.

Details

An *index* is a single-integer representation of position of an entry in a matrix (example: `M[index]`).

A *subscript* is a pair-of-integer representation (example: `M[i,j]` where `(i,j)` is the subscript).

`ind2sub` translates indexes to subscripts.

`sub2ind` translates subscripts to indexes.

Examples

```
subs = ind2sub(10, 1:20)
inds = sub2ind(10, subs[,1], subs[,2])
```

`index_data` *Create an object of index data*

Description

Create an object of index data, either with incremental data and a sampling scheme, or directly with index data.

Usage

```
index_data(x, ...)

is.index_data(x)

## S3 method for class 'index_data'
dim(x)

## S3 method for class 'sampling_scheme'
index_data(x, incremental_data, ...)

## S3 method for class 'numeric'
index_data(x, sample_data, m, n, data_id, ...)
```

Arguments

`x`	either a `sampling_scheme` object or a numeric vector of the index of the sampled data. If `x` is a `sampling_scheme`, the matrix to be sampled should be supplied by `incremental_data`. If `x` is a numeric vector, `sampled_data` should be supplied.
`incremental_data`	an objecto of the `incremental_data` class to be sampled.
`sample_data`	the observations of indexes (cumulative values). When supplied, should match the numeric vector `x`.
`m, n`	the dimension of the incremental data matrix to be estimated. In other words, they are the dimensions of `diff(index_data)`, not of the sampled accumulated index matrix.
`data_id`	the ID of the data.

Value

an object of `index_data` class.

Examples

```
m = 5
n = 3
```

```
r = 2
example_matrix = matrix(rexp(m * r), ncol = r) %*%
    matrix(rexp(n * r), ncol = n)
# create an incremental_data object
incre_d = incremental_data(x = example_matrix, data_id = "exponential")

seed = 42
# we will sample incre_d with a random sampling_scheme
# with a 50% sampling rate and a fixed seed
sample_incre_d_at = sampling_scheme(
    sampling_rate = 0.5, type = "random", seed = seed)

# create an index_data object
index_d = index_data(x = sample_incre_d_at, incremental_data = incre_d)
lapply(index_d,head)
index_d2 = index_data(x = index_d$sample_index,
    sample_data = index_d$sample_data,
    m = index_d$dim_data[1],
    n = index_d$dim_data[2],
    data_id = index_d$data_id)
```

`index_data_to_matrix_measurement`
Convert `index_data` *to a measurement matrix and the observations*

Description

Convert `index_data` to a measurement matrix and the observations

Usage

```
index_data_to_matrix_measurement(measurements_index_data)

sparse_matrix_to_matrix_measurement(sparse_matrix)
```

Arguments

`measurements_index_data`
: an `index_data` object.

`sparse_matrix`
: a sparse matrix where the zero entries are unobserved.

Value

a list with two fields:

- `matrix_A`: a sparse matrix representing the measurement operator
- `measurements`: a vector of length `dim{matrix_A}[1]`.

Examples

```
measurements_m = index_data_to_matrix_measurement(example_index_data)
```

---

is.incremental_data

Reports whether x is an incremental_data object

Description

Reports whether x is an incremental_data object

Usage

```
is.incremental_data(x)
```

Arguments

x	An object to test

---

linear_measurement_data

Create a data object with general linear measurements

Description

Create a data object with general linear measurements

Usage

```
linear_measurement_data(x, ...)

## S3 method for class 'numeric'
linear_measurement_data(x, matrix_A, dim_data,
  exo_variable = NULL, data_id)

## S3 method for class 'sampling_scheme'
linear_measurement_data(x, data,
  exo_variable = NULL, ...)

## S3 method for class 'measurement_matrix'
linear_measurement_data(x, data, ...)

## S3 method for class 'sparseMatrix'
```

```
linear_measurement_data(x, exo_variable = NULL,
  data_id = "")

is.linear_measurement_data(x)

## S3 method for class 'linear_measurement_data'
dim(x)
```

Arguments

`x`	a numeric vector, a `sampling_scheme`, a `measurement_matrix`, or a `Matrix::sparseMatrix`.
`matrix_A`	the measurement matrix used to generate the measures. Should be an object of class `Matrix::sparseMatrix` of dimension `length(x)`-by-`prod(dim_data)`.
`dim_data`	a length-2 vector with the dimension of the measured matrix
`exo_variable`	optional exogenous variables.
`data_id`	the ID of the dataset.
`data`	either an `incremental_data` or a `linear_measurement_data` object. The data is to be sampled with `x`.

Details

- If `x` is a numeric vector, `matrix_A` and `dim_data` should be supplied.
- If `x` is a `sampling_scheme`, `data` should be supplied, and it shoud be either `incremental_data` or `linear_measurement_data`.
- If `x` is a `measurement_matrix`, `data` should be of the class `incremental_data`.
- If `x` is a `Matrix::sparseMatrix`, it is supposed that it is a matrix with missing entries. The linear measurement operator is one which samples all observed entries in `x`. More specifically, `x` should not be a `Matrix::nsparseMatrix` (a binary pattern matrix), but one which actually contains the value of the observations. Otherwise there will be a warning.

Value

a `linear_measurement_data` object which includes the following fields:

- `sample_data`: the observations in the form of a numeric vector.
- `measurement_matrix`: a `measurement_matrix` of dimension `length(sample_data)`-by-`prod(dim_data)`
- `dim_data`: the dimension of the matrix to be estimated.
- `exo_variable`: the exogenous variables. Named list with fields called `X` and `Y`. Should verify `nrow(exo_variable$X) == dim_data[1]` and `nrow(exo_variable$Y) == dim_data[2]`.
- `data_id`: a character string. The name of the dataset.

See Also

`measurement_matrix`

Examples

```
m = 5
n = 3
r = 2
example_matrix = matrix(rexp(m * r), ncol = r) %*%
    matrix(rexp(n * r), ncol = n)
# create an incremental_data object
incre_d = incremental_data(x = example_matrix, data_id = "exponential")

# A measurement matrix that samples the first five entries of a matrix.
matrix_A = measurement_matrix(
   "matrix",
   matrix_A = Matrix::sparseMatrix(i = 1:5, j = 1:5, dims = c(5, m*n)))
lin_m_d = linear_measurement_data(
  x = matrix_A,
  data = incre_d)


## examples of linear_measurement_data.sparseMatrix
completion_data = Matrix::sparseMatrix(i = 1,j = 1, x = 0, dims=c(5,5))
linear_measurement_data = linear_measurement_data(x = completion_data)
## Not run:
completion_data = Matrix::sparseMatrix(i = 1,j = 1, dims=c(5,5))
# this generates a warning because  we are not supposed to provide a
# sparse "pattern" matrix to linear_measurement_data.sparseMatrix
linear_measurement_data = linear_measurement_data(x = completion_data)

## End(Not run)
```

---

lissageV *Exponential smoothing*

Description

Standard exponential smoothing of a numeric vector.

Usage

```
lissageV(vect, theta)
```

Arguments

`vect`	the numeric vector to smooth.
`theta`	the smoothing parameter. Should be between 0 and 1.

Value

A numeric vector of the same length as `vect`.

References

the package `R39Toolbox` developped by R39.

Examples

```
lissageV(1:10, 0.1)
```

---

match_aggregate_max

Evaluation functions concerning aggregated (daily) peaks

Description

These functions compare the periodic peaks (for example daily or hourly or weekly) of the individual aggregated ground truth with that of an estimation. This is to evalute if the NMF algorithms can be applied to estimate periodic peaks from temporally aggregated individual data.

Usage

```
match_aggregate_max(matrix1, matrix2, timestep, aggregate_vector)

diff_puissance_moment_pmax(matrix1, matrix2, timestep, aggregate_vector)

diff_aggregate_max(matrix1, matrix2, timestep, aggregate_vector)

diff_aggregate_max_positive(matrix1, matrix2, timestep, aggregate_vector)

diff_aggregate_max_negative(matrix1, matrix2, timestep, aggregate_vector)
```

Arguments

`matrix1, matrix2`
: The ground truth and the estimation.

`timestep`
: The temporal aggregation step at which to calculate the peaks. If the each row of `matrix1` and `matrix2` are half hourly datat, and we are interested in daily peaks, `timestep` should be 48.

`aggregate_vector`
: The aggregation vector to aggregate individuals.

Details

`match_aggregate_max` calculate the percentage of peak moments that are correctly estimated.

`diff_puissance_moment_pmax` calculates the average deviation between the estimation and ground truth at the peak moments (both estimated and real).

`diff_aggregate_max` calculates the average deviation between the estimaed peaks and the real peaks.

`diff_aggregate_max_positive` calculates the average positive deviation between the estimaed peaks and the real peaks.

`diff_aggregate_max_negative` calculates the average negative deviation between the estimaed peaks and the real peaks.

Value

a single value representing an evaluation metric.

`matrix_model` *General matrix model*

Description

Create a general matrxi model. An object of `matrix_model` class has a `predict` method. There are two types of matrix models, `factorization_model` and `regression_model`. The difference between them is that `predict.regression_model` allows new data. This is where the dispatching between factorization and regression models happens based on the data and the algorithm.

Usage

```
matrix_model(algorithm = implemented_algorithms(), data, ...)

is.matrix_model(x)
```

Arguments

`algorithm`	the algorithm used to create the model.
`data`	an object of class `incremental_data`, `index_data` or `linear_measurement_data`.
`...`	other parameters to be passed on to `factorization_model` or `regression_model`.

Value

an object of class `matrix_model` which has at least the following fields:

- `M`: the recovered matrix.
- `algorithm`: the algorithm.
- `obs`: a named list of the observations provided for model estimation.
- `args_list`: a named list of the other arguments.

If `algorithm` is not compatible with the data and other parameters that are provided, this function will return an `empty_model`, which is a naive model.

See Also

regression_model, factorization_model, empty_model.

Examples

```
model0 = matrix_model(data = example_index_data, algorithm = "empty_model")
model1 = matrix_model(data = example_index_data, r = 2, algorithm = "NeNMF")

predict(model0, completion = TRUE)
predict(model0, completion = FALSE)
```

---

measurement_matrix *Create a* measurement_matrix *to represent the measurement operator*

Description

Create a measurement_matrix to represent the measurement operator

Usage

```
measurement_matrix(type, ...)
```

Arguments

type	a character string specifying the type of representation used in the other parameters.
pattern_mat, matrix_A, list_mat, list_subs, list_ind, vec_ind	parameters to be used to pass one representation of the measurement operator. See Details for more explanation.

Details

The parameter type can be of the following values:

- "completion" a binary matrix of the same dimension as the matrices to be measured. The "pattern" matrix should be supplied under the parameter name pattern_mat.
- "matrix" a matrix which is the measurement operator in the matrix form. The parameter matrix_A should be supplied. It should be a sparseMatrix (class from the Matrix package).
- "list_mat" a list of matrices, each of the same dimension as the matrices to be measured. Each matrix is a "mask". The parameter list_mat should be supplied.
- "list_sub", "list_ind", "vec_ind". All three are used for completion data. "list_ind" is a list of length of nrow of the matrix to be measured, in which each element is a vector of the row indexes of the observations. "list_sub" is a list of vectors of the "sub" (row column pair) of the observations (the parameter name is list_subs). "vec_ind" is a vector of the indexes (by seeing the matrix as a very long vector) of the observations.

meterModels *meterModels*

Description

meterModels

NMF_increment *NMF on increment data*

Description

Algortihm for the incremental NMF problem. Different algorithms are called depending on the `algorithm` argument. Supposed to be internal.

Usage

```
NMF_increment(data, known = NULL, m, n, r, matrix_A = NULL,
  algorithm = implemented_algorithms("completion"), opts = list(),
  rho = NULL, lambda = 0, ground_truth = NULL, NNDSVD = FALSE,
  save_iterations = FALSE, general_factorization = FALSE,
  exo_variable = list(X = NULL, Y = NULL), formula = list(X =
  empty_formula(), Y = empty_formula()), ...)
```

Arguments

`data`	a numeric vector with the observed values (cumulative data).
`known`	an integer vector witht the index of the observed values. Has to be the same length as `data`.
`m, n`	the dimension of the matrix to be estimated.
`r`	integer. the target rank of the matrix.
`matrix_A`	a sparse matrix of dimension `length(data)`-by-`m` * `n`. Used when estimating models with general linear measurements. Either `known` or `matrix_A` has to be supplied.
`algorithm`	a character string with the name of the algorithm to be used. See `implemented_algorith` for the complete list.
`opts`	a named list to set the parameters for the estimation algorithm with the following fields: • `tol`: tolerance for deciding whether to stop. • `maxit`: the maximal number of iterations. • `minit`: the minimal number of iterations. • `verbose`: whether to provide more reporting in the computation or not.

- `fixed_X`, `fixed_Y`: whether `X` or `Y` should be fixed in the iterations.
- `maxtime`: maximal computing time allocated in seconds.
- `ITER_MAX`, `ITER_MIN`: the maximal and minimal number of iterations for NeNMF.
- `X0`,`Y0`: initial values for `X`, `Y`
- `FuninR`: If `FuninR = "R"` use the R version of subroutines. If `FuninR = "cpp"` use the C++ version. Default is `"cpp"`.
- `ccdppitmax`,`ccdppthreads`,`ccdpplambda`: the maximal number of iterations, the number of threads, and the penalization parameter for CCDPP algorithm.

`rho` either `NULL` or a vector of length `n`. When it is provided, it is the vector of temporal autocorrelation threshold to be imposed on each of the columns.

`lambda` either a number which is 0 or positive, or `"optimal"`. This is the penalization parameter to be used to impose autocorrelation constraint.

`ground_truth` the ground truth matrix of incremental data.

`NNDSVD` a logical to specify whether use SVD to initialize.

`save_iterations` whether to save the factorizations during iterations.

`general_factorization` If `TRUE` does a general factorization without nonnegativity. If `FALSE`, which is the default value, does an NMF. IF is the string `"random"`, randomly allow half of the iterations to explore negative values.

`exo_variable` A list of two fields named `X` and `Y`. Both are to be data.frames, of `m` or `n` rows, corresponding to the row individuals or the column individuals of the matrix to be recovered.

`formula` A list of two fields named `X` and `Y`. Both are to be objects of the class `formula_params`.

`...` additional parameters to be passed to `mgcv::gam` for initializing the models when using HALS_models and gam.

`normalize_factors` *Normalize the factors in a* `factorization_model`

Description

This function rescales `output$X` to have columns which have averages of one. `output$Y` is modified accordingly to keep the model equivalent.

Usage

```
normalize_factors(output)
```

Arguments

`output` a `factorization_model`

Value

A `factorization_model` with normalized factors.

`periodic_sampling` *Helper functions to subsample the matrices*

Description

These functions are returned wrapped inside `sampling_schemes`.

Usage

```
periodic_sampling(m, n, sr, seed = NULL)

random_sampling(m, n, sr, seed = NULL)

evolution_sampling(m, n, sr, seed = NULL)

block_sampling(m, n, sr, seed = NULL)
```

Arguments

`m, n`	integers. The number of rows and columns in the matrix.
`sr`	the sampling rate (0.2 for sampling 20% of data). For every sampling type except `evolution_sampling`, this should be a single number between 0 and 1. For `evoluation_sampling`, this should be a pair of sampling rates. `sr[1]` is the (periodic) sampling rate applied on regular users. `sr[2]` is the percentage of regular users. The rest of the users are observed completely (supposed to be users with Linky meters).
`seed`	the seed for sampling the phase.

Details

`periodic_sampling` proposes sampling entries which are at regular intervals (with random offset for each column).

`random_sampling` proposes sampling entries which are chosen uniformly at random amongst all entries in the matrix.

`evolution_sampling` proposes sampling entries at regular intervals for a part of the users, and complete observations for the rest of the users.

`block_sampling` supposes that a block of data is missing, and returns the entries that are not missing.

Value

The sampled (observed) entries' indexes. For `mat`, a matrix of dimension `m`-by-`n`, call `mat[xxxxx_sampling(m, n, sr)]` to retrieve the observed entries.

plot.factorization_model

Plot a `factorization_model`

Description

Plots the factors and the weights of a `factorization_model`, using `plot_factors` and `plot_weights`.

Usage

```
plot.factorization_model(x, ...)
```

Arguments

`x`	an object of class `factorization_model`, aka, has the fields `X` and `Y`.
`...`	parameters passed to `plot_factors` and `plot_weights`.

See Also

See `factorization_model` for examples. See `plot_factors` and `plot_weights` for additional parameters.

plot.regression_model

Plot a regression_model

Description

This function plots the regression models (the models between factors and exogenous variables).

Usage

```
plot.regression_model(x)
```

Arguments

`x`	an object of class `regression_model`. `x$algorithm` should be either "HALSX1" or "factor_gam".

See Also

See `regression_model` for examples.

plot_a_matrix *Plot a matrix*

Description

Plot a matrix using `geom_tile` in `ggplot2` package.

Usage

```
plot_a_matrix(M)

plot_matrices(list_matrices)
```

Arguments

`M`	a numeric matrix.
`list_matrices`	a (potentially named) list of matrices.

Examples

```
plot_a_matrix(example_matrix)
plot_matrices(list(example_matrix,
                   cbind(jitter(example_matrix), example_matrix)))
```

plot_factors *Plot factors*

Description

Plot factors obtained in a `factorization model`. This function takes a numeric matrix and plot each column as a curve. Underlying this function is the idea that factors that are obtained are time series.

Usage

```
plot_factors(W, start = NULL, interval = NULL, ...)
```

Arguments

`W`	a numeric matrix (factors obtained in NMF typically).
`start, interval`	options to put time x-axis values for factors. `start` is a `POSIXct`. `interval` is a `difftime`.

See Also

`plot_weights`

Examples

```
plot_factors(example_matrix)

# We can also specify the periods covered by the factors
plot_factors(example_matrix, start = as.POSIXct(Sys.Date()),
             interval = as.difftime(1, units = "days"))
```

`plot_weights` *Plot weights*

Description

Plot a numeric matrix by interpreting each column as weights of factors for one individual. Each column of `H` is renormalized to sum 1.

Usage

```
plot_weights(H, order_by_factor = 1, ...)
```

Arguments

`H`	a numeric matrix (weights obtained in NMF).
`order_by_factor`	The factor by which to order the individuals. For the graphic to be more informative, we reorder the individuals according to one of the factors.

See Also

`plot_factors`

Examples

```
plot_weights(example_matrix)
```

`predict.matrix_model`
Predict method for `matrix_model`*s*

Description

Predict method for `matrix_model`s

Usage

```
predict.matrix_model(object, newdata = NULL, completion = NULL, ...)

predict.factorization_model(object, completion, ...)

predict.regression_model(object, newdata = NULL, len = NULL, ...)

predict.empty_model(object, completion, ...)
```

Arguments

`object`	the `matrix_model` to use in prediction.
`newdata`	When `object` is a `regression_model`, provide new values for the exogeneous variables for prediction in the form of `list(X = newdataX, Y = newdataY)` where `newdataX` and `newdataY` are `data.frame`s corresponding to new rows and columns.
`completion`	a `logical` value. Should the returned prediction of a `factorization_model` be projected into the subspace imposed by the data?
`...`	in `predict.matrix_model` used to pass additional arguments to `predict.regression_m` In the other two methods, it is unused.
`len`	used if `object$algorithm == "trmf"`. Since `trmf` algorithm does not use `newdata`, we have to tell it how many periods to predict.

Details

When not specified explicitly, predict treats a regression_model that is also a factorization_model as a regression_model.

Value

If `newdata` is supplied and `object` is a `regression_model` with appropriate `$algorithm`, a matrix of dimension `nrow(newdata$X)`-by-`nrow(newdata$Y)`. If only one of the newdata `data.frame`s is provided, the missing dimension is replaced by that of the data used to estimate `object`.

If no `newdata` is supplied, the matrix recovery result is returned, either projected or not, depending on the value of `completion`.

If the parameters are not compatible, `NULL` is returned.

See Also

For examples, see `matrix_model`, `regression_model`, `factorization_model`, and `empty_model`.

`print.formula_params` *Create a* `formula_params` *object which controls what regression_model to use*

Description

This function creates a `formula_params` object. See below for details.

Usage

```
## S3 method for class 'formula_params'
print(x, ...)

formula_params(formula_alone, HALS_model_params = list(),
  HALSX1_params = list(), trmf_params = list())

is.formula_params(x)

## S3 method for class 'formula_params'
as.character(x)
```

Arguments

`formula_alone`
: a string that looks like the right-hand side of a typical R `formula`

`HALS_model_params`
: A named list of parameters for the `"HALS_model"` algorithm. Should include
  - `reg_model`: a character string specifying which regression method to use. Supported values are `"gam"` or regression methods supported by `caret`. If is `"gam"`, the `mgcv` package will be used. If is anything else, the `caret` package will be used for estimation of the regression models.
  - Other parameters are passed in the estimation iterations to either `mgcv::gam` or `caret::train`.

`HALSX1_params`
: A named list of parameters for the `"HALSX1"` algorithm. Can use this to pass on a parameter called `sp` to specify penalization parameter in spline regression.

`trmf_params`
: A named list of parameters for the `"trmf"` algorithm. The parameters are optional:
  - `lag_idx`: The number of periods considered in the autoregressive models of the factors.
  - `maxit`: The maximal number of iterations.
  - `lambdas`: A vector of three numbers: penalization parameters on `X`, `Y` and `M` (the matrix to be recovered).

Details

A `formula_params` object has a `formula_alone` which is a string that looks like the right-hand side of a typical R `formula`. Other fields are used to store algorithm-specific parameters, so `HALS_model_params` is for the `"HALS_model"` algorithm etc..

Value

a `formula_params` object. Basically a wrapper of the parameters.

References

https://topepo.github.io/caret/available-models.html for the list of available regression methods in `caret`.

Examples

```
formula_params(create_formula_string(colnames(
               example_exogenous_variables$X), type = "gam"),
               HALS_model_params = list(reg_method = "gam"))
formula_params(create_formula_string(colnames(
               example_exogenous_variables$X), type = "lm"),
               HALS_model_params = list(reg_method = "xgbTree",
                                        # All tuning parameters to pass to
                                        # caret is to be supplied here
                                        subsample = 0.4))
```

`regression_model` *Fit a regression model*

Description

Fit a regression model with index data or incremental data, with exogeneous variables

Usage

```
regression_model(x, exo_variable, formula, r = NULL,
  algorithm = implemented_algorithms("regression"), opts = list(),
  rho = NULL, lambda = 0, ground_truth = NULL, NNDSVD = FALSE,
  save_iterations = FALSE, general_factorization = FALSE, ...)

is.regression_model(x)

## S3 method for class 'index_data'
regression_model(x, exo_variable, formula, r = NULL,
  algorithm = implemented_algorithms("regression", "partial_data"),
  opts = list(), rho = NULL, lambda = 0, ground_truth = NULL,
  NNDSVD = FALSE, save_iterations = FALSE, general_factorization = FALSE,
```

```
  ...)

## S3 method for class 'linear_measurement_data'
regression_model(x, exo_variable, formula,
  r = NULL, algorithm = implemented_algorithms("regression",
  "linear_measurement_data"), opts = list(), rho = NULL, lambda = 0,
  ground_truth = NULL, NNDSVD = FALSE, save_iterations = FALSE,
  general_factorization = FALSE, ...)

## S3 method for class 'factorization_model'
regression_model(x, exo_variable, formula, ...)

## S3 method for class 'incremental_data'
regression_model(x, exo_variable = NULL, formula,
  r = NULL, algorithm = implemented_algorithms("regression"),
  opts = list(), rho = NULL, lambda = 0, ground_truth = NULL,
  NNDSVD = FALSE, save_iterations = FALSE, general_factorization = FALSE,
  ...)
```

Arguments

`x` an object of class `index_data`, `incremental_data`, `linear_measurement_data`, or `factorization_model`.

`exo_variable` A list of two fields named `X` and `Y`. Both are to be data.frames, of `m` or `n` rows, corresponding to the row individuals or the column individuals of the matrix to be recovered.

`formula` A list of two fields named `X` and `Y`. Both are to be objects of the class `formula_params`.

`r` integer. the target rank of the matrix.

`algorithm` a character string with the name of the algorithm to be used. See `implemented_algorithms` for the complete list.

`opts` a named list to set the parameters for the estimation algorithm with the following fields:

- `tol`: tolerance for deciding whether to stop.
- `maxit`: the maximal number of iterations.
- `minit`: the minimal number of iterations.
- `verbose`: whether to provide more reporting in the computation or not.
- `fixed_X`, `fixed_Y`: whether `X` or `Y` should be fixed in the iterations.
- `maxtime`: maximal computing time allocated in seconds.
- `ITER_MAX`, `ITER_MIN`: the maximal and minimal number of iterations for NeNMF.
- `X0`,`Y0`: initial values for `X`, `Y`
- `FuninR`: If `FuninR = "R"` use the R version of subroutines. If `FuninR = "cpp"` use the C++ version. Default is `"cpp"`.
- `ccdppitmax`,`ccdppthreads`,`ccdpplambda`: the maximal number of iterations, the number of threads, and the penalization parameter for CCDPP algorithm.

`rho` either `NULL` or a vector of length `n`. When it is provided, it is the vector of temporal autocorrelation threshold to be imposed on each of the columns.

`lambda` either a number which is 0 or positive, or `"optimal"`. This is the penalization parameter to be used to impose autocorrelation constraint.

`ground_truth` the ground truth matrix of incremental data.

`NNDSVD` a logical to specify whether use SVD to initialize.

`save_iterations` whether to save the factorizations during iterations.

`general_factorization` If `TRUE` does a general factorization without nonnegativity. If `FALSE`, which is the default value, does an NMF. IF is the string `"random"`, randomly allow half of the iterations to explore negative values.

`...` additional parameters to passe on to NMF_increment.

Value

A `regression_model` object. Always has a field called `models` which consequently can be used in prediction. When the object returned is also a `matrix_model` or a `factorization_model`, corresponding fields are included.

See Also

`formula_params` to see details on how to provide parameters for the regression algorithms.

Examples

```
reg_mod = regression_model(x = example_index_data,
                           exo_variable = example_exogenous_variables,
                           formula = example_formula,
                           r = 2,
                           algorithm = "HALS_model")

# When not specified explicitly,
# predict treats a regression_model that is also a factorization_model as
# a regression_model.
# Predict a regression_model without newdata to see the prediction
# values of the regression models.
predict(reg_mod)

# Call predict.factorization_model explicitly on regression_model to see
# prediction by the factorization_model
predict.factorization_model(reg_mod, completion = TRUE)
predict.factorization_model(reg_mod, completion = FALSE)

# Predict using newdata
predict(reg_mod, newdata = list(X = example_exogenous_variables$X[1:2,]))
predict(reg_mod, newdata = list(Y = example_exogenous_variables$Y[1,,drop=FALSE]))
predict(reg_mod, newdata = list(X = example_exogenous_variables$X[1:2,],
                                Y = example_exogenous_variables$Y[1,,drop=FALSE]))

# When not specified explicitly, it is the regression_model part that is
# plotted.
```

```
plot(reg_mod)
# We can force it to plot factorization_model.
plot.factorization_model(reg_mod)
```

`relative_diff_norm` *Relative difference in norm between two matrices*

Description

Relative difference in norm between two matrices

Usage

```
relative_diff_norm(matrix1, matrix2, type = "F")

normalized_deviation(matrix1, matrix2)

normalized_RMSE(matrix1, matrix2)

RMSE(matrix1, matrix2)

MAPE(matrix1, matrix2)
```

Arguments

`matrix1,`	matrix2 the two matrices to compare.
`type`	currently unused. For future development can be used to inplement other metrix than Frobenius norm.

`sampling_scheme` *A wrapper class for the sampling schemes*

Description

Specify a sampling scheme with its type and parameters

Usage

```
sampling_scheme(sampling_rate, type = c("periodic", "random", "evolution",
  "block", "completion", "all"), seed = NULL)

is.sampling_scheme(object)
```

Arguments

`sampling_rate`
: a number strictly between 0 and 1, or a numeric vector. If `type == "evolution"`, it is two numbers between 0 and 1. If `type == "all"`, this is ignored.

`type`
: a character string specifying the sampling type. See `implemented_sampling_type` for more details.

`seed`
: the seed used in random number generation.

Value

an object of class `sampling_scheme`. Apart from the information passed on from the parameters, this object has a `$sampling_fun` field which contains a function that we can then use to sample data. See `xxx_sampling` for more details about these functions.

Examples

```
ss = sampling_scheme(0.2, "periodic")
ss$sampling_fun(m = 10, n = 20)
```

`set_missing_parameters_to_default`

Internal function to set missing estimation options to default values

---

Description

Internal function to set missing estimation options to default values

Usage

```
set_missing_parameters_to_default(opts, algorithm)
```

Arguments

`opts`
: a named list to set the parameters for the estimation algorithm with the following fields:

- `tol`: tolerance for deciding whether to stop.
- `maxit`: the maximal number of iterations.
- `minit`: the minimal number of iterations.
- `verbose`: whether to provide more reporting in the computation or not.
- `fixed_X`, `fixed_Y`: whether `X` or `Y` should be fixed in the iterations.
- `maxtime`: maximal computing time allocated in seconds.
- `ITER_MAX`, `ITER_MIN`: the maximal and minimal number of iterations for NeNMF.
- `X0`,`Y0`: initial values for `X`, `Y`

- `FuninR`: If `FuninR = "R"` use the R version of subroutines. If `FuninR = "cpp"` use the C++ version. Default is `"cpp"`.
- `ccdppitmax`,`ccdppthreads`,`ccdpplambda`: the maximal number of iterations, the number of threads, and the penalization parameter for CCDPP algorithm.

Bibliography

References for Chapter 1: Résumé en français

[4] Daniel D. Lee and H. Sebastian Seung. "Learning the Parts of Objects by Non-Negative Matrix Factorization". In: *Nature* 401.6755 (1999), pp. 788–791 *Cited on pages 5, 67.*

[5] David Donoho and Victoria Stodden. "When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts?" In: *Advances in Neural Information Processing Systems*. 2003, None *Cited on pages 5, 6, 55, 58, 87.*

[6] Nicolas Gillis. "The Why and How of Nonnegative Matrix Factorization". In: *Regularization, Optimization, Kernels, and Support Vector Machines* 12 (2014), p. 257 *Cited on pages 5, 69, 70.*

[7] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 8 (2009), pp. 30–37 *Cited on page 5.*

[8] Wei Xu, Xin Liu, and Yihong Gong. "Document Clustering Based on Non-Negative Matrix Factorization". In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM, 2003, pp. 267–273 *Cited on page 5.*

[9] Cédric Févotte. "Itakura-Saito Nonnegative Factorizations of the Power Spectrogram for Music Signal Decomposition". In: *Machine Audition: Principles, Algorithms and Systems* (2010), pp. 266–296 *Cited on page 5.*

[10] RTE. *Balance Responsible Entity System*. http://clients.rte-france.com/lang/an/clients_producteurs/services_clients/dispositif_re.jsp. May 2014 *Cited on pages 5, 67.*

[11] Stephen A. Vavasis. "On the Complexity of Nonnegative Matrix Factorization". In: *SIAM Journal on Optimization* 20.3 (2009), pp. 1364–1377 *Cited on pages 6, 59.*

[12] Hans Laurberg, Mads Græsbøll Christensen, Mark D. Plumbley, Lars Kai Hansen, and Søren Holdt Jensen. "Theorems on Positive Data: On the Uniqueness of NMF". en. In: *Computational Intelligence and Neuroscience* 2008 (2008), pp. 1–9. DOI: 10.1155/2008/764206 *Cited on pages 6, 87, 88.*

[13] Kejun Huang, Nicholas D. Sidiropoulos, and Ananthram Swami. "Non-Negative Matrix Factorization Revisited: Uniqueness and Algorithm for Symmetric Decomposition". In: *IEEE Transactions on Signal Processing* 62.1 (Jan. 2014), pp. 211–224. DOI: 10.1109/TSP.2013.2285514 *Cited on page 6.*

[14] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. "Computing a Nonnegative Matrix Factorization–provably". In: *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*. ACM, 2012, pp. 145–162 *Cited on page 6.*

[15] Nicolas Gillis. "Sparse and Unique Nonnegative Matrix Factorization through Data Preprocessing". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 3349–3386 *Cited on page 6.*

[16] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. "Escaping From Saddle Points—Online Stochastic Gradient for Tensor Decomposition". In: *arXiv preprint arXiv:1503.02101* (2015) *Cited on pages 6, 59.*

[17] Ben Recht, Christopher Re, Joel Tropp, and Victor Bittorf. "Factoring Nonnegative Matrices with Linear Programs". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1214–1222 *Cited on pages 6, 58.*

[18] Jingu Kim, Yunlong He, and Haesun Park. "Algorithms for Nonnegative Matrix and Tensor Factorizations: A Unified View Based on Block Coordinate Descent Framework". In: *Journal of Global Optimization* 58.2 (2014), pp. 285–319 *Cited on pages 6, 70, 71, 90, 91, 94.*

[19] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. "Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization". In: *Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 169–176 *Cited on pages 7, 70, 90, 92.*

[20] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. "NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization". In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2882–2898. DOI: 10.1109/TSP.2012.2190406 *Cited on pages 7, 61, 70, 71, 116.*

[21] Yunmei Chen and Xiaojing Ye. "Projection Onto A Simplex". In: *arXiv preprint arXiv:1101.6081* (2011) *Cited on pages 7, 70.*

[22] Zhe Chen and Andrzej Cichocki. "Nonnegative Matrix Factorization with Temporal Smoothness and/or Spatial Decorrelation Constraints". In: *Laboratory for Advanced Brain Signal Processing, RIKEN, Tech. Rep* 68 (2005) *Cited on pages 7, 69.*

[23] Cédric Févotte and Jérôme Idier. "Algorithms for Nonnegative Matrix Factorization with the Beta-Divergence". In: *Neural Computation* 23.9 (2011), pp. 2421–2456 *Cited on pages 7, 69.*

[24] Paris Smaragdis, Cédric Févotte, Gautham J. Mysore, Nasser Mohammadiha, and Matthew Hoffman. "Static and Dynamic Source Separation Using Nonnegative Factorizations: A Unified View". In: *IEEE Signal Processing Magazine* 31.3 (2014), pp. 66–75. DOI: 10.1109/MSP.2013.2297715 *Cited on pages 7, 69.*

[25] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd. "Generalized Low Rank Models". In: *Foundations and Trends in Machine Learning* 9.1 (2016). DOI: 10.1561/2200000055 *Cited on pages 7, 69.*

[26] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S. Dhillon. "High-Dimensional Time Series Prediction with Missing Values". In: *arXiv preprint arXiv:1509.08333* (2015) *Cited on pages 7, 69, 101, 104.*

[27] Andrzej Cichocki, ed. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. Chichester, U.K: John Wiley, 2009. ISBN: 978-0-470-74666-0 *Cited on pages 8, 61.*

[28] Artur Trindade. *UCI Maching Learning Repository - ElectricityLoadDiagrams20112014 Data Set*. http://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014. 2016 *Cited on pages 9, 75, 100.*

[29] Commission for Energy Regulation, Dublin. "Electricity Smart Metering Customer Behaviour Trials Findings Report." In: (2011) *Cited on pages 9, 75.*

[30] Commission for Energy Regulation, Dublin. "Results of Electricity Coast-Benefit Analysis, Customer Behaviour Trials and Technology Trials Commission for Energy Regulation." In: (2011) *Cited on pages 9, 75.*

References for Chapter 2: Introduction

[1] Jiali Mei, Yannig Goude, Georges Hebrail, and Nicholas Kong. "Spatial Estimation of Electricity Consumption Using Socio-Demographic Information". In: *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*. Oct. 2016, pp. 753–757. DOI: 10.1109/APPEEC.2016.7779596 *Cited on pages 12, 43.*

[31] H. Lee Willis. *Spatial electric load forecasting*. CRC Press, 2002 *Cited on pages 12, 44.*

[32] Heiko Hahn, Silja Meyer-Nieberg, and Stefan Pickl. "Electric load forecasting methods: Tools for decision making". In: *European Journal of Operational Research* 199.3 (Dec. 2009), pp. 902–907. ISSN: 03772217. DOI: 10.1016/j.ejor.2009.01.062 *Cited on pages 12, 44.*

[33] J.D. Melo, A. Padilha-Feltrin, and E.M. Carreno. "Data issues in spatial electric load forecasting". In: *2014 IEEE PES General Meeting | Conference Exposition*. 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939848 *Cited on pages 12, 44.*

[34] X. Sun, P. B. Luh, K. W. Cheung, W. Guan, L. D. Michel, S. S. Venkata, and M. T. Miller. "An Efficient Approach to Short-Term Load Forecasting at the Distribution Level". In: *IEEE Transactions on Power Systems* 31.4 (2016), pp. 2526–2537. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2015.2489679 *Cited on pages 12, 44.*

[35] R. Li, C. Gu, F. Li, G. Shaddick, and M. Dale. "Development of Low Voltage Network Templates - Part I: Substation Clustering and Classification". In: *IEEE Transactions on Power Systems* 30.6 (2015), pp. 3036–3044. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2014.2371474 *Cited on pages 12, 44.*

[36] Yannig Goude, Raphael Nedellec, and Ning Kong. "Local Short and Middle Term Electricity Load Forecasting with Semi-Parametric Additive Models". In: *Smart Grid, IEEE Transactions on* 5.1 (2014), pp. 440–446 *Cited on page 12.*

[37] Souhaib Ben Taieb, James W. Taylor, and Rob J. Hyndman. "Coherent Probabilistic Forecasts for Hierarchical Time Series". en. In: *PMLR*. July 2017, pp. 3348–3357 *Cited on page 12.*

[38] *GEFCom2014*. http://www.drhongtao.com/gefcom/2014 *Cited on page 12.*

[39] Pierre Gaillard, Yannig Goude, and Raphaël Nedellec. "Additive Models and Robust Aggregation for GEFCom2014 Probabilistic Electric Load and Electricity Price Forecasting". In: *International Journal of Forecasting* 32.3 (2016), pp. 1038–1050 *Cited on page 12.*

[40] *Datascience.Net - Prévision de La Consommation Électrique - RTE*. https://www.datascience.net/fr/challenge/32/details# *Cited on page 12.*

References for Chapter 3: Spatio-temporal change of support

[41] N Cressie. *Statistics for Spatial Data: Wiley Series in Probability and Statistics*. Wiley: New York, NY, USA, 1993 *Cited on pages 20, 21, 23, 26.*

[42] Jean-Marc Azaïs and Mario Wschebor. *Level sets and extrema of random processes and fields*. John Wiley & Sons, 2009 *Cited on page 21.*

[43] Robert J Adler. *The geometry of random fields*. Vol. 62. SIAM, 1981 *Cited on page 21.*

[44] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. the MIT Press, 2006 *Cited on pages 22, 109.*

[45] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer-Verlag, 1999 *Cited on page 22.*

[46] Georges Matheron. *Traité de géostatistique appliquée*. Editions Technip, 1962 *Cited on page 22.*

[47] Geoff Bohling. *Data Analysis in Engineering and Natural Science*. Kansas Geological Survey, University of Kansas, 2005 *Cited on page 23.*

[48] Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2011 *Cited on pages 26, 29, 30.*

[49] Sudipto Banerjee, Bradley P Carlin, and Alan E Gelfand. *Hierarchical modeling and analysis for spatial data*. Crc Press, 2014 *Cited on pages 26, 27.*

[50] Michael F Goodchild, Nina Siu Ngan Lam, and University of Western Ontario. Dept. of Geography. *Areal interpolation: a variant of the traditional spatial problem*. London, Ont.: Department of Geography, University of Western Ontario, 1980 *Cited on page 26.*

[51] Robin Flowerdew and Mick Green. "Statistical methods for inference between incompatible zonal systems". In: *Accuracy of spatial databases* (1989), pp. 239–247 *Cited on page 26.*

[52] Andrew S Mugglin, Bradley P Carlin, and Alan E Gelfand. "Fully model-based approaches for spatially misaligned data". In: *Journal of the American Statistical Association* 95.451 (2000), pp. 877–887 *Cited on page 26.*

[53] Carol A Gotway and Linda J Young. "Combining incompatible spatial data". In: *Journal of the American Statistical Association* 97.458 (2002), pp. 632–648 *Cited on page 26.*

[54] Roger S Bivand, Edzer J Pebesma, Virgilio Gomez-Rubio, and Edzer Jan Pebesma. *Applied spatial data analysis with R*. Vol. 747248717. Springer, 2008 *Cited on page 26.*

[55] Jean-Paul Chiles and Pierre Delfiner. *Geostatistics: modeling spatial uncertainty*. Vol. 497. John Wiley & Sons, 2009 *Cited on page 26.*

[56] Hsin-Cheng Huang, Noel Cressie, and John Gabrosek. "Fast, resolution-consistent spatial prediction of global processes from satellite data". In: *Journal of Computational and Graphical Statistics* 11.1 (2002), pp. 63–88 *Cited on page 26.*

[57] Eric D Kolaczyk and Haiying Huang. "Multiscale statistical models for hierarchical spatial aggregation". In: *Geographical Analysis* 33.2 (2001), pp. 95–118 *Cited on page 26.*

[58] Michel Goulard and Marc Voltz. "Linear coregionalization model: tools for estimation and choice of cross-variogram matrix". In: *Mathematical Geology* 24.3 (1992), pp. 269–286 *Cited on page 27.*

[59] Andrew O. Finley, Sudipto Banerjee, and Bradley P. Carlin. "spBayes: An R Package for Univariate and Multivariate Hierarchical Point-Referenced Spatial Models". In: *Journal of Statistical Software* 19.4 (2007), pp. 1–24 *Cited on page 27.*

[60] Donald E Myers. "Pseudo-cross variograms, positive-definiteness, and cokriging". In: *Mathematical Geology* 23.6 (1991), pp. 805–816 *Cited on page 28.*

[61] Noel Cressie and Hsin-Cheng Huang. "Classes of nonseparable, spatio-temporal stationary covariance functions". In: *Journal of the American Statistical Association* 94.448 (1999), pp. 1330–1339 *Cited on page 29.*

[62] Edzer J Pebesma. "Multivariable geostatistics in S: the gstat package". In: *Computers & Geosciences* 30.7 (2004), pp. 683–691 *Cited on page 30.*

[63] Roger Bivand. *CRAN Task View: Analysis of Spatial Data.* 2015. (Visited on 10/30/2015) *Cited on page 30.*

[64] *Regles relatives a la Programmation, au Mecanisme d'Ajustement et au dispositif de Responsable d'Equilibre, Regles chapitre F.* Accessed: 2016-07-08 *Cited on pages 35, 47.*

References for Chapter 4: Spatial estimation of electricity consumption using socio-demographic information

[1] Jiali Mei, Yannig Goude, Georges Hebrail, and Nicholas Kong. "Spatial Estimation of Electricity Consumption Using Socio-Demographic Information". In: *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC).* Oct. 2016, pp. 753–757. DOI: 10.1109/APPEEC.2016.7779596 *Cited on pages 12, 43.*

[31] H. Lee Willis. *Spatial electric load forecasting.* CRC Press, 2002 *Cited on pages 12, 44.*

[32] Heiko Hahn, Silja Meyer-Nieberg, and Stefan Pickl. "Electric load forecasting methods: Tools for decision making". In: *European Journal of Operational Research* 199.3 (Dec. 2009), pp. 902–907. ISSN: 03772217. DOI: 10.1016/j.ejor.2009.01.062 *Cited on pages 12, 44.*

[33] J.D. Melo, A. Padilha-Feltrin, and E.M. Carreno. "Data issues in spatial electric load forecasting". In: *2014 IEEE PES General Meeting | Conference Exposition.* 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939848 *Cited on pages 12, 44.*

[34] X. Sun, P. B. Luh, K. W. Cheung, W. Guan, L. D. Michel, S. S. Venkata, and M. T. Miller. "An Efficient Approach to Short-Term Load Forecasting at the Distribution Level". In: *IEEE Transactions on Power Systems* 31.4 (2016), pp. 2526–2537. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2015.2489679 *Cited on pages 12, 44.*

[35] R. Li, C. Gu, F. Li, G. Shaddick, and M. Dale. "Development of Low Voltage Network Templates - Part I: Substation Clustering and Classification". In: *IEEE Transactions on Power Systems* 30.6 (2015), pp. 3036–3044. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2014.2371474 *Cited on pages 12, 44.*

[64] *Regles relatives a la Programmation, au Mecanisme d'Ajustement et au dispositif de Responsable d'Equilibre, Regles chapitre F.* Accessed: 2016-07-08 *Cited on pages 35, 47.*

[65] Shu Fan and Rob J. Hyndman. "Short-term load forecasting based on a semi-parametric additive model". In: *Power Systems, IEEE Transactions on* 27.1 (2012), pp. 134–141 *Cited on page 44.*

[66] Amandine Pierrot and Yannig Goude. "Short-term electricity load forecasting with generalized additive models". In: *Proceedings of ISAP power.* 2011, pp. 593–600 *Cited on page 44.*

[67] Piotr Mirowski, Sining Chen, Tin Kam Ho, and Chun-Nam Yu. "Demand Forecasting in Smart Grids". In: *Bell Labs Technical Journal* 18.4 (2014), pp. 135–158. ISSN: 1538-7305. DOI: 10.1002/bltj.21650 *Cited on pages 44, 50, 51.*

[68] R.A. Sevlian and R. Rajagopal. "A model for the effect of aggregation on short term load forecasting". In: *PES General Meeting | Conference Exposition, 2014 IEEE.* 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6938899 *Cited on page 44.*

[69] Carlos Alzate and Mathieu Sinn. "Improved electricity load forecasting via kernel spectral clustering of smart meters". In: *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 943–948 *Cited on page 44.*

[70] Joshua D. Rhodes, Wesley J. Cole, Charles R. Upshaw, Thomas F. Edgar, and Michael E. Webber. "Clustering analysis of residential electricity demand profiles". In: *Applied Energy* 135 (2014), pp. 461–471 *Cited on page 44.*

[71] Franklin L. Quilumba, Wei-Jen Lee, Heng Huang, David Y. Wang, and Robert L. Szabados. "Using Smart Meter Data to Improve the Accuracy of Intraday Load Forecasting Considering Customer Behavior Similarities". In: *Smart Grid, IEEE Transactions on* 6.2 (2015), pp. 911–918 *Cited on page 44.*

[72] *United States Census Bureau*. Accessed: 2016-02-10 *Cited on page 45.*

[73] *2011 census for England and Wales - Office for National Statistics*. Accessed: 2016-02-10 *Cited on page 45.*

[74] *INSEE - Database - Population census results*. Accessed: 2016-02-10 *Cited on pages 45, 48.*

[75] Alexander Bruhns, Gilles Deurveilher, and Jean-Sébastien Roy. "A non linear regression model for mid-term load forecasting and improvements in seasonality". In: *Proceedings of the 15th Power Systems Computation Conference*. 2005, pp. 22–26 *Cited on page 45.*

[76] Simon Wood. *Generalized additive models: an introduction with R*. CRC press, 2006 *Cited on pages 45, 46.*

[77] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001 *Cited on page 46.*

References for Chapter 5: Introduction to nonnegative matrix factorization

[5] David Donoho and Victoria Stodden. "When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts?" In: *Advances in Neural Information Processing Systems*. 2003, None *Cited on pages 5, 6, 55, 58, 87.*

[11] Stephen A. Vavasis. "On the Complexity of Nonnegative Matrix Factorization". In: *SIAM Journal on Optimization* 20.3 (2009), pp. 1364–1377 *Cited on pages 6, 59.*

[16] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. "Escaping From Saddle Points—Online Stochastic Gradient for Tensor Decomposition". In: *arXiv preprint arXiv:1503.02101* (2015) *Cited on pages 6, 59.*

[17] Ben Recht, Christopher Re, Joel Tropp, and Victor Bittorf. "Factoring Nonnegative Matrices with Linear Programs". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1214–1222 *Cited on pages 6, 58.*

[20] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. "NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization". In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2882–2898. DOI: 10.1109/TSP.2012.2190406 *Cited on pages 7, 61, 70, 71, 116.*

[27] Andrzej Cichocki, ed. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. Chichester, U.K: John Wiley, 2009. ISBN: 978-0-470-74666-0 *Cited on pages 8, 61.*

[78] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization". In: *SIAM review* 52.3 (2010), pp. 471–501 *Cited on pages 58, 59, 69, 83, 85, 86.*

[79] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu. "Spatio-Temporal Compressive Sensing and Internet Traffic Matrices (Extended Version)". In: *IEEE/ACM Transactions on Networking* 20.3 (June 2012), pp. 662–676. DOI: 10.1109/TNET.2011.2169424 *Cited on pages 58, 59, 69, 70, 85, 86.*

[80] Chris Ding, Tao Li, Wei Peng, and Haesun Park. "Orthogonal Nonnegative Matrix T-Factorizations for Clustering". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2006, pp. 126–135 *Cited on page 58.*

[81] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. "Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems". In: *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 765–774 *Cited on page 58.*

[82] Mark A. Davenport and Justin Romberg. "An Overview of Low-Rank Matrix Recovery from Incomplete Observations". In: *IEEE Journal of Selected Topics in Signal Processing* 10.4 (2016), pp. 608–622 *Cited on page 59.*

[83] Eftychios A Pnevmatikakis and Liam Paninski. "Sparse Nonnegative Deconvolution for Compressive Calcium Imaging: Algorithms and Phase Transitions". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 1250–1258 *Cited on pages 59, 86.*

[84] T. Tony Cai, Anru Zhang, et al. "ROP: Matrix Recovery via Rank-One Projections". In: *The Annals of Statistics* 43.1 (2015), pp. 102–138 *Cited on pages 59, 84, 85.*

[85] Or Zuk and Avishai Wagner. "Low-Rank Matrix Recovery from Row-and-Column Affine Measurements". In: *Proceedings of The 32nd International Conference on Machine Learning*. 2015, pp. 2012–2020 *Cited on pages 59, 69, 84–86.*

[86] Emmanuel J. Candès and Benjamin Recht. "Exact Matrix Completion via Convex Optimization". en. In: *Foundations of Computational Mathematics* 9.6 (2009), pp. 717–772. DOI: 10.1007/s10208-009-9045-5 *Cited on pages 59, 68, 85, 86.*

[87] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. "Low-Rank Matrix Completion Using Alternating Minimization". In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '13. New York, NY, USA: ACM, 2013, pp. 665–674. ISBN: 978-1-4503-2029-0. DOI: 10.1145/2488608.2488693 *Cited on page 59.*

[88] Ruoyu Sun and Zhi-Quan Luo. "Guaranteed Matrix Completion via Nonconvex Factorization". In: *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium On*. IEEE, 2015, pp. 270–289 *Cited on page 59.*

[89] Rong Ge, Jason D. Lee, and Tengyu Ma. "Matrix Completion Has No Spurious Local Minimum". In: *arXiv:1605.07272 [cs, stat]* (May 2016). arXiv: 1605.07272 [cs, stat] *Cited on page 59.*

[90] Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. "Global Optimality of Local Search for Low Rank Matrix Recovery". In: *arXiv:1605.07221 [cs, math, stat]* (May 2016). arXiv: 1605.07221 [cs, math, stat] *Cited on pages 59, 60, 65, 66, 86, 112.*

[91] Yuanzhi Li, Yingyu Liang, and Andrej Risteski. "Recovery Guarantee of Non-Negative Matrix Factorization via Alternating Updates". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4987–4995 *Cited on pages 59, 127.*

[92] Dohyung Park, Anastasios Kyrillidis, Constantine Carmanis, and Sujay Sanghavi. "Non-Square Matrix Sensing without Spurious Local Minima via the Burer-Monteiro Approach". en. In: *PMLR*. Apr. 2017, pp. 65–74 *Cited on page 59.*

[93] Xingguo Li, Zhaoran Wang, Junwei Lu, Raman Arora, Jarvis Haupt, Han Liu, and Tuo Zhao. "Symmetry, Saddle Points, and Global Geometry of Nonconvex Matrix Factorization". In: *arXiv:1612.09296v2 [cs]* (Jan. 2017) *Cited on page 59.*

[94] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. "The Loss Surfaces of Multilayer Networks". In: *arXiv:1412.0233 [cs]* (Nov. 2014). arXiv: 1412.0233 [cs] *Cited on page 59.*

[95] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. "Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization". In: *arXiv:1406.2572 [cs, math, stat]* (June 2014). arXiv: 1406.2572 [cs, math, stat] *Cited on page 59.*

[96] Afonso S. Bandeira, Nicolas Boumal, and Vladislav Voroninski. "On the Low-Rank Approach for Semidefinite Programs Arising in Synchronization and Community Detection". In: *arXiv:1602.04426 [math]* (Feb. 2016). arXiv: 1602.04426 [math] *Cited on page 59.*

[97] Song Mei, Yu Bai, and Andrea Montanari. "The Landscape of Empirical Risk for Non-Convex Losses". In: *arXiv preprint arXiv:1607.06534* (2016) *Cited on page 59.*

[98] Yuanzhi Li and Yingyu Liang. "Provable Alternating Gradient Descent for Non-Negative Matrix Factorization with Strong Correlations". en. In: *PMLR*. July 2017, pp. 2062–2070 *Cited on pages 59, 127.*

[99] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena scientific Belmont, 1999 *Cited on page 60.*

[100] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Science & Business Media, 2011 *Cited on page 64.*

References for Chapter 6: Nonnegative matrix factorization for time series recovery from a few temporal aggregates

[2] Jiali Mei, Yohann De Castro, Yannig Goude, and Georges Hébrail. "Nonnegative Matrix Factorization for Time Series Recovery From a Few Temporal Aggregates". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 6–11 Aug 2017, pp. 2382–2390 *Cited on pages 67, 82, 85, 86, 95.*

[4] Daniel D. Lee and H. Sebastian Seung. "Learning the Parts of Objects by Non-Negative Matrix Factorization". In: *Nature* 401.6755 (1999), pp. 788–791 *Cited on pages 5, 67.*

[6] Nicolas Gillis. "The Why and How of Nonnegative Matrix Factorization". In: *Regularization, Optimization, Kernels, and Support Vector Machines* 12 (2014), p. 257 *Cited on pages 5, 69, 70.*

[10] RTE. *Balance Responsible Entity System*. http://clients.rte-france.com/lang/an/clients_producteurs/services_clients/dispositif_re.jsp. May 2014 *Cited on pages 5, 67.*

[18] Jingu Kim, Yunlong He, and Haesun Park. "Algorithms for Nonnegative Matrix and Tensor Factorizations: A Unified View Based on Block Coordinate Descent Framework". In: *Journal of Global Optimization* 58.2 (2014), pp. 285–319 *Cited on pages 6, 70, 71, 90, 91, 94.*

[19] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. "Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization". In: *Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 169–176 *Cited on pages 7, 70, 90, 92.*

[20] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. "NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization". In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2882–2898. DOI: 10.1109/TSP.2012.2190406 *Cited on pages 7, 61, 70, 71, 116.*

[21] Yunmei Chen and Xiaojing Ye. "Projection Onto A Simplex". In: *arXiv preprint arXiv:1101.6081* (2011) *Cited on pages 7, 70.*

[22] Zhe Chen and Andrzej Cichocki. "Nonnegative Matrix Factorization with Temporal Smoothness and/or Spatial Decorrelation Constraints". In: *Laboratory for Advanced Brain Signal Processing, RIKEN, Tech. Rep* 68 (2005) *Cited on pages 7, 69.*

[23] Cédric Févotte and Jérôme Idier. "Algorithms for Nonnegative Matrix Factorization with the Beta-Divergence". In: *Neural Computation* 23.9 (2011), pp. 2421–2456 *Cited on pages 7, 69.*

[24] Paris Smaragdis, Cédric Févotte, Gautham J. Mysore, Nasser Mohammadiha, and Matthew Hoffman. "Static and Dynamic Source Separation Using Nonnegative Factorizations: A Unified View". In: *IEEE Signal Processing Magazine* 31.3 (2014), pp. 66–75. DOI: 10.1109/MSP.2013.2297715 *Cited on pages 7, 69.*

[25] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd. "Generalized Low Rank Models". In: *Foundations and Trends in Machine Learning* 9.1 (2016). DOI: 10.1561/2200000055 *Cited on pages 7, 69.*

[26] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S. Dhillon. "High-Dimensional Time Series Prediction with Missing Values". In: *arXiv preprint arXiv:1509.08333* (2015) *Cited on pages 7, 69, 101, 104.*

[28] Artur Trindade. *UCI Maching Learning Repository - ElectricityLoadDiagrams20112014 Data Set*. http://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014. 2016 *Cited on pages 9, 75, 100.*

[29] Commission for Energy Regulation, Dublin. "Electricity Smart Metering Customer Behaviour Trials Findings Report." In: (2011) *Cited on pages 9, 75.*

[30] Commission for Energy Regulation, Dublin. "Results of Electricity Coast-Benefit Analysis, Customer Behaviour Trials and Technology Trials Commission for Energy Regulation." In: (2011) *Cited on pages 9, 75.*

[78] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization". In: *SIAM review* 52.3 (2010), pp. 471–501 *Cited on pages 58, 59, 69, 83, 85, 86.*

[79] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu. "Spatio-Temporal Compressive Sensing and Internet Traffic Matrices (Extended Version)". In: *IEEE/ACM Transactions on Networking* 20.3 (June 2012), pp. 662–676. DOI: 10.1109/TNET.2011.2169424 *Cited on pages 58, 59, 69, 70, 85, 86.*

[85] Or Zuk and Avishai Wagner. "Low-Rank Matrix Recovery from Row-and-Column Affine Measurements". In: *Proceedings of The 32nd International Conference on Machine Learning*. 2015, pp. 2012–2020 *Cited on pages 59, 69, 84–86.*

[86] Emmanuel J. Candès and Benjamin Recht. "Exact Matrix Completion via Convex Optimization". en. In: *Foundations of Computational Mathematics* 9.6 (2009), pp. 717–772. DOI: 10.1007/s10208-009-9045-5 *Cited on pages 59, 68, 85, 86.*

[101] SVK. *Balance Responsibility*. http://www.svk.se/en/stakeholder-portal/Electricity-market/Balance-responsibility/. Oct. 2016 *Cited on page 67.*

[102] REE. *Balance Responsible Party*. https://www.esios.ree.es/en/glossary#letterB. 2016 *Cited on page 67.*

[103] Aharon Ben-Tal and Dick den Hertog. "Hidden Conic Quadratic Representation of Some Nonconvex Quadratic Optimization Problems". In: *Mathematical Programming* 143.1-2 (2013), pp. 1–29. DOI: 10.1007/s10107-013-0710-8 *Cited on pages 69, 72.*

[104] Angelika Rohde and Alexandre B. Tsybakov. "Estimation of High-Dimensional Low-Rank Matrices". en. In: *The Annals of Statistics* 39.2 (2011), pp. 887–930. DOI: 10.1214/10-AOS860 *Cited on pages 69, 86.*

[105] Emmanuel J. Candès and Yaniv Plan. "Tight Oracle Inequalities for Low-Rank Matrix Recovery From a Minimal Number of Noisy Random Measurements". In: *IEEE Transactions on Information Theory* 57.4 (2011), pp. 2342–2359. DOI: 10.1109/TIT.2011.2111771 *Cited on pages 69, 85.*

[106] Pierre Alquier and Benjamin Guedj. "An Oracle Inequality for Quasi-Bayesian Non-Negative Matrix Factorization". In: *axXiv preprint arXiv:1601.01345* (2016) *Cited on page 69.*

[107] Nicolas Gillis and François Glineur. "Low-Rank Matrix Approximation with Weights or Missing Data Is NP-Hard". In: *SIAM Journal on Matrix Analysis and Applications* 32.4 (2011), pp. 1149–1165 *Cited on page 69.*

[108] Yangyang Xu and Wotao Yin. "A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion". en. In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1758–1789. DOI: 10.1137/120887795 *Cited on page 69.*

[109] Eftychios A. Pnevmatikakis and Liam Paninski. "Sparse Nonnegative Deconvolution for Compressive Calcium Imaging: Algorithms and Phase Transitions". In: *Advances in Neural Information Processing Systems*. 2013, pp. 1250–1258 *Cited on page 69.*

[110] Luigi Grippo and Marco Sciandrone. "On the Convergence of the Block Nonlinear Gauss–Seidel Method under Convex Constraints". In: *Operations Research Letters* 26.3 (2000), pp. 127–136 *Cited on pages 71, 90, 91.*

[111] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. "Spectral Regularization Algorithms for Learning Large Incomplete Matrices". In: *Journal of machine learning research* 11.Aug (2010), pp. 2287–2322 *Cited on pages 76, 104.*

References for Chapter 7: Time series recovery and prediction using NMF with side information

[2] Jiali Mei, Yohann De Castro, Yannig Goude, and Georges Hébrail. "Nonnegative Matrix Factorization for Time Series Recovery From a Few Temporal Aggregates". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 6–11 Aug 2017, pp. 2382–2390 *Cited on pages 67, 82, 85, 86, 95.*

[3] Jiali Mei, Yohann De Castro, Yannig Goude, Georges Hébrail, and Jean-Marc Azaïs. "Nonnegative Matrix Factorization with Side Information for Time Series Recovery and Prediction". In: *submitted* (2017) *Cited on page 81.*

[5] David Donoho and Victoria Stodden. "When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts?" In: *Advances in Neural Information Processing Systems*. 2003, None *Cited on pages 5, 6, 55, 58, 87.*

[12] Hans Laurberg, Mads Græsbøll Christensen, Mark D. Plumbley, Lars Kai Hansen, and Søren Holdt Jensen. "Theorems on Positive Data: On the Uniqueness of NMF". en. In: *Computational Intelligence and Neuroscience* 2008 (2008), pp. 1–9. DOI: 10.1155/2008/764206 *Cited on pages 6, 87, 88.*

[18] Jingu Kim, Yunlong He, and Haesun Park. "Algorithms for Nonnegative Matrix and Tensor Factorizations: A Unified View Based on Block Coordinate Descent Framework". In: *Journal of Global Optimization* 58.2 (2014), pp. 285–319 *Cited on pages 6, 70, 71, 90, 91, 94.*

[19] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. "Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization". In: *Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 169–176 *Cited on pages 7, 70, 90, 92.*

[26] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S. Dhillon. "High-Dimensional Time Series Prediction with Missing Values". In: *arXiv preprint arXiv:1509.08333* (2015) *Cited on pages 7, 69, 101, 104.*

[28] Artur Trindade. *UCI Maching Learning Repository - ElectricityLoadDiagrams20112014 Data Set*. http://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014. 2016 *Cited on pages 9, 75, 100.*

[78] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization". In: *SIAM review* 52.3 (2010), pp. 471–501 *Cited on pages 58, 59, 69, 83, 85, 86.*

[79] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu. "Spatio-Temporal Compressive Sensing and Internet Traffic Matrices (Extended Version)". In: *IEEE/ACM Transactions on Networking* 20.3 (June 2012), pp. 662–676. DOI: 10.1109/TNET.2011.2169424 *Cited on pages 58, 59, 69, 70, 85, 86.*

[83] Eftychios A Pnevmatikakis and Liam Paninski. "Sparse Nonnegative Deconvolution for Compressive Calcium Imaging: Algorithms and Phase Transitions". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 1250–1258 *Cited on pages 59, 86.*

[84] T. Tony Cai, Anru Zhang, et al. "ROP: Matrix Recovery via Rank-One Projections". In: *The Annals of Statistics* 43.1 (2015), pp. 102–138 *Cited on pages 59, 84, 85.*

[85] Or Zuk and Avishai Wagner. "Low-Rank Matrix Recovery from Row-and-Column Affine Measurements". In: *Proceedings of The 32nd International Conference on Machine Learning*. 2015, pp. 2012–2020 *Cited on pages 59, 69, 84–86.*

[86] Emmanuel J. Candès and Benjamin Recht. "Exact Matrix Completion via Convex Optimization". en. In: *Foundations of Computational Mathematics* 9.6 (2009), pp. 717–772. DOI: 10.1007/s10208-009-9045-5 *Cited on pages 59, 68, 85, 86.*

[90] Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. "Global Optimality of Local Search for Low Rank Matrix Recovery". In: *arXiv:1605.07221 [cs, math, stat]* (May 2016). arXiv: 1605.07221 [cs, math, stat] *Cited on pages 59, 60, 65, 66, 86, 112.*

[104] Angelika Rohde and Alexandre B. Tsybakov. "Estimation of High-Dimensional Low-Rank Matrices". en. In: *The Annals of Statistics* 39.2 (2011), pp. 887–930. DOI: 10.1214/10-AOS860 *Cited on pages 69, 86.*

[105] Emmanuel J. Candès and Yaniv Plan. "Tight Oracle Inequalities for Low-Rank Matrix Recovery From a Minimal Number of Noisy Random Measurements". In: *IEEE Transactions on Information Theory* 57.4 (2011), pp. 2342–2359. DOI: 10.1109/TIT.2011.2111771 *Cited on pages 69, 85.*

[110] Luigi Grippo and Marco Sciandrone. "On the Convergence of the Block Nonlinear Gauss–Seidel Method under Convex Constraints". In: *Operations Research Letters* 26.3 (2000), pp. 127–136 *Cited on pages 71, 90, 91.*

[111] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. "Spectral Regularization Algorithms for Learning Large Incomplete Matrices". In: *Journal of machine learning research* 11.Aug (2010), pp. 2287–2322 *Cited on pages 76, 104.*

[112] Prateek Jain and Inderjit S. Dhillon. "Provable Inductive Matrix Completion". In: *arXiv preprint arXiv:1306.0626* (2013) *Cited on pages 81, 85, 86.*

[113] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. "Collaborative Filtering with Graph Information: Consistency and Scalable Methods". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 2107–2115 *Cited on pages 81, 85, 86, 101, 105.*

[114] Si Si, Kai-Yang Chiang, Cho-Jui Hsieh, Nikhil Rao, and Inderjit S. Dhillon. "Goal-Directed Inductive Matrix Completion". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. Aug. 2016 *Cited on pages 81, 85, 86.*

[115] Raja Velu and Gregory C. Reinsel. *Multivariate Reduced-Rank Regression: Theory and Applications*. en. Google-Books-ID: dsfSBwAAQBAJ. Springer Science & Business Media, Apr. 2013. ISBN: 978-1-4757-2853-8 *Cited on pages 85, 86.*

[116] Florentina Bunea, Yiyuan She, and Marten H. Wegkamp. "Joint Variable and Rank Selection for Parsimonious Estimation of High-Dimensional Matrices". EN. In: *The Annals of Statistics* 40.5 (Oct. 2012), pp. 2359–2388. DOI: 10.1214/12-AOS1039 *Cited on pages 85, 86.*

[117] Lisha Chen and Jianhua Z. Huang. "Sparse Reduced-Rank Regression for Simultaneous Dimension Reduction and Variable Selection". In: *Journal of the American Statistical Association* 107.500 (2012), pp. 1533–1545 *Cited on page 85.*

[118] V. Kekatos, Y. Zhang, and G. B. Giannakis. "Electricity Market Forecasting via Low-Rank Multi-Kernel Learning". In: *IEEE Journal of Selected Topics in Signal Processing* 8.6 (Dec. 2014), pp. 1182–1193. DOI: 10.1109/JSTSP.2014.2336611 *Cited on pages 85, 86.*

[119] Rina Foygel, Michael Horrell, Mathias Drton, and John D. Lafferty. "Nonparametric Reduced Rank Regression". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1628–1636 *Cited on pages 85, 86.*

[120] Deepak Agarwal and Bee-Chung Chen. "Regression-Based Latent Factor Models". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2009, pp. 19–28 *Cited on pages 85, 86.*

[121] Miao Xu, Rong Jin, and Zhi-Hua Zhou. "Speedup Matrix Completion with Side Information: Application to Multi-Label Learning". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 2301–2309 *Cited on pages 85, 86.*

[122] Kai-Yang Chiang, Cho-Jui Hsieh, and Inderjit S Dhillon. "Matrix Completion with Noisy Side Information". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 3447–3455 *Cited on pages 85, 86.*

[123] Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. "A New Approach to Collaborative Filtering: Operator Estimation with Spectral Regularization". In: *The Journal of Machine Learning Research* 10 (2009), pp. 803–826 *Cited on pages 85, 86.*

[124] Kun Chen, Hongbo Dong, and Kung-Sik Chan. "Reduced Rank Regression via Adaptive Nuclear Norm Penalization". In: *Biometrika* 100.4 (Dec. 2013), pp. 901–920. DOI: 10.1093/biomet/ast036 *Cited on page 86.*

[125] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Vol. 23. Prentice hall Englewood Cliffs, NJ, 1989 *Cited on page 91.*

[126] Simon Wood. *Generalized Additive Models: An Introduction with R*. CRC press, 2006 *Cited on pages 96, 97.*

[127] Max Kuhn. "Building Predictive Models in R Using the Caret Package". In: *Journal of Statistical Software* 28.5 (2008), pp. 1–26 *Cited on page 98.*

[128] *MovieLens 100K Dataset*. https://grouplens.org/datasets/movielens/100k/. 2015-09-23T15:02:16+00:00 *Cited on page 100.*

[129] Chris Addy. "Reduced-Rank Regression [R Package Rrr Version 1.0.0]". In: () *Cited on pages 101, 104.*

[130] Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. "Kernlab-an S4 Package for Kernel Methods in R". In: (2004) *Cited on page 101.*

References for Chapter 8: Comparing matrix factorization with traditional methods

[44] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. the MIT Press, 2006 *Cited on pages 22, 109.*

[90] Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. "Global Optimality of Local Search for Low Rank Matrix Recovery". In: *arXiv:1605.07221 [cs, math, stat]* (May 2016). arXiv: 1605.07221 [cs, math, stat] *Cited on pages 59, 60, 65, 66, 86, 112.*

References for Chapter 9: Implementation of nonnegative matrix factorization algorithms

[20] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. "NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization". In: *IEEE Transactions on Signal Processing* 60.6 (2012), pp. 2882–2898. DOI: 10.1109/TSP.2012.2190406 *Cited on pages 7, 61, 70, 71, 116.*

[131] Benjamin Recht and Christopher Ré. "Parallel Stochastic Gradient Algorithms for Large-Scale Matrix Completion". In: *Mathematical Programming Computation* 5.2 (2013), pp. 201–226 *Cited on page 120.*

[132] Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, S. V. N. Vishwanathan, and Inderjit Dhillon. "NOMAD: Non-Locking, stOchastic Multi-Machine Algorithm for Asynchronous and Decentralized Matrix Completion". In: *Proceedings of the VLDB Endowment* 7.11 (2014), pp. 975–986 *Cited on page 120.*

[133] Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. "A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 6.1 (2015), p. 2 *Cited on page 120.*

[134] Wei-Sheng Chin, Bo-Wen Yuan, Meng-Yuan Yang, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. "LIBMF: A Library for Parallel Matrix Factorization in Shared-Memory Systems". In: *Journal of Machine Learning Research* 17.86 (2016), pp. 1–5 *Cited on page 120.*

[135] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. "Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems". In: *Data Mining (ICDM), 2012 IEEE 12th International Conference On*. IEEE, 2012, pp. 765–774 *Cited on page 120.*

References for Chapter 10: Perspectives

[91] Yuanzhi Li, Yingyu Liang, and Andrej Risteski. "Recovery Guarantee of Non-Negative Matrix Factorization via Alternating Updates". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4987–4995 *Cited on pages 59, 127.*

[98] Yuanzhi Li and Yingyu Liang. "Provable Alternating Gradient Descent for Non-Negative Matrix Factorization with Strong Correlations". en. In: *PMLR*. July 2017, pp. 2062–2070 *Cited on pages 59, 127.*

[136] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. "Accurate Image Super-Resolution Using Very Deep Convolutional Networks". In: *arXiv:1511.04587 [cs]* (Nov. 2015). arXiv: 1511.04587 [cs] *Cited on page 126.*

[137] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. "Image Super-Resolution Using Deep Convolutional Networks". In: *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2016), pp. 295–307 *Cited on page 126.*

[138] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. "Compressed Sensing Using Generative Models". en. In: *PMLR*. July 2017, pp. 537–546 *Cited on page 126.*

université PARIS-SACLAY
École doctorale de mathématiques Hadamard (EDMH)

Reconstitution et prédiction de séries temporelles avec la factorisation de matrice nonnégative augmentée de régression appliquée à la consommation électrique

Mots clés Analyse spatiale, Séries chronologiques, Consommation électrique, Séparation de sources, Factorisation de matrice nonnégative

Résumé Nous sommes intéressés par la reconstitution et la prédiction des séries temporelles multi-variées à partir des données partiellement observées et/ou agrégées. La motivation du problème vient des applications dans la gestion du réseau électrique.

Nous envisageons des outils capables de résoudre le problème d'estimation de plusieurs domaines. Après investiguer le krigeage, qui est une méthode de la littérature de la statistique spatio-temporelle, et une méthode hybride basée sur le clustering des individus, nous proposons un cadre général de reconstitution et de prédiction basé sur la factorisation de matrice nonnégative. Ce cadre prend en compte de manière intrinsèque la corrélation entre les séries temporelles pour réduire drastiquement la dimension de l'espace de paramètres. Une fois que le problématique est formalisé dans ce cadre, nous proposons deux extensions par rapport à l'approche standard.

La première extension prend en compte l'autocorrélation temporelle des individus. Cette information supplémentaire permet d'améliorer la précision de la reconstitution. La deuxième extension ajoute une composante de régression dans la factorisation de matrice nonnégative. Celle-ci nous permet d'utiliser dans l'estimation du modèle des variables exogènes liées avec la consommation électrique, ainsi de produire des facteurs plus interprétables, et aussi améliorer la reconstitution. De plus, cette méthode nous donne la possibilité d'utiliser la factorisation de matrice nonnégative pour produire des prédictions.

Sur le côté théorique, nous nous intéressons à l'identifiabilité du modèle, ainsi qu'à la propriété de la convergence des algorithmes que nous proposons. La performance des méthodes proposées en reconstitution et en prédiction est testée sur plusieurs jeux de données de consommation électrique à niveaux d'agrégation différents.

Time series recovery and prediction with regression-enhanced nonnegative matrix factorization applied to electricity consumption

Keywords Spatial analysis, Time series, Electricity consumption, Source separation, Nonnegative matrix factorization

Abstract We are interested in the recovery and prediction of multiple time series from partially observed and/or aggregate data. Motivated by applications in electricity network management, we investigate tools from multiple fields that are able to deal with such data issues.

After examining kriging from spatio-temporal statistics and a hybrid method based on the clustering of individuals, we propose a general framework based on nonnegative matrix factorization. This framework takes advantage of the intrinsic correlation between the multivariate time series to greatly reduce the dimension of the parameter space. Once the estimation problem is formalized in the nonnegative matrix factorization framework, two extensions are proposed to improve the standard approach.

The first extension takes into account the individual temporal autocorrelation of each of the time series. This increases the precision of the time series recovery. The second extension adds a regression layer into nonnegative matrix factorization. This allows exogenous variables that are known to be linked with electricity consumption to be used in estimation, hence makes the factors obtained by the method to be more interpretable, and also increases the recovery precision. Moreover, this method makes the method applicable to prediction.

We produce a theoretical analysis on the framework which concerns the identifiability of the model and the convergence of the algorithms that are proposed. The performance of proposed methods to recover and forecast time series is tested on several multivariate electricity consumption datasets at different aggregation level.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France