# A Markovian approach to distributional semantics

Edouard Grave

Thèse

*presentée à*

L'université Paris VI - Pierre et Marie Curie
École doctorale informatique,
télécommunications et électronique

*par*

Édouard Grave

*pour obtenir*

le grade de docteur en sciences
specialité : informatique

# *A Markovian approach to distributional semantics*

*Directeurs de thèse :* Francis Bach et Guillaume Obozinski.

*Soutenue le dd/mm/yyyy, devant la comission d'examen formée de :*

| | | |
|---|---|---|
| Nom 1 | Université 1 | Fonction 1 |
| Nom 2 | Université 2 | Fonction 2 |
| Nom 3 | Université 3 | Fonction 3 |
| Nom 4 | Université 4 | Fonction 4 |
| Nom 5 | Université 5 | Fonction 5 |

# ABSTRACT

This thesis, which is organized in two independent parts, presents work on distributional semantics and on variable selection.

In the first part, we introduce a new method for learning good word representations using large quantities of unlabeled sentences. The method is based on a probabilistic model of sentence, using a hidden Markov model and a syntactic dependency tree. The latent variables, which correspond to the nodes of the dependency tree, aim at capturing the meanings of the words. We develop an efficient algorithm to perform inference and learning in those models, based on online EM and approximate message passing. We then evaluate our models on intrinsic tasks such as predicting human similarity judgements or word categorization, and on two extrinsic tasks: named entity recognition and supersense tagging.

In the second part, we introduce, in the context of linear models, a new penalty function to perform variable selection in the case of highly correlated predictors. This penalty, called the trace Lasso, uses the trace norm of the selected predictors, which is a convex surrogate of their rank, as the criterion of model complexity. The trace Lasso interpolates between the $\ell_1$-norm and $\ell_2$-norm. In particular, it is equal to the $\ell_1$-norm if all predictors are orthogonal and to the $\ell_2$-norm if all predictors are equal. We propose two algorithms to compute the solution of least-squares regression regularized by the trace Lasso, and perform experiments on synthetic datasets to illustrate the behavior of the trace Lasso.

KEYWORDS: distributional semantics; hidden Markov model; dependency tree; word representation; semantic class; variable selection; trace Lasso.

# RÉSUMÉ

Cette thèse, organisée en deux parties indépendantes, a pour objet la semantique distributionnelle et la sélection de variables.

Dans la première partie, nous introduisons une nouvelle méthode pour l'apprentissage de représentations de mots à partir de grandes quantités de texte brut. Cette méthode repose sur un modèle probabiliste de la phrase, utilisant modèle de Markov caché et arbre de dépendance. Nous présentons un algorithme efficace pour réaliser l'inférence et l'apprentissage dans un tel modèle, fondé sur l'algorithme EM en ligne et la propagation de message approchée. Nous évaluons les modèles obtenus sur des taches intrinsèques, telles que prédire des jugements de similarité humains ou catégoriser des mots et deux taches extrinsèques : la reconnaissance d'entités nommées et l'étiquetage en supersens.

Dans la seconde partie, nous introduisons, dans le contexte des modèles linéaires, une nouvelle pénalité pour la sélection de variables en présence de prédicteurs fortement corrélés. Cette pénalité, appelée *trace Lasso*, utilise la norme trace des prédicteurs sélectionnés, qui est une relaxation convexe de leur rang, comme critère de complexité. Le trace Lasso interpole les normes $\ell_1$ et $\ell_2$. En particulier, lorsque tous les prédicteurs sont orthogonaux, il est égal à la norme $\ell_1$, tandis que lorsque tous les prédicteurs sont égaux, il est égal à la norme $\ell_2$. Nous proposons deux algorithmes pour calculer la solution du problème de régression aux moindres carrés regularisé par le trace Lasso et réalisons des expériences sur des données synthétiques.

MOTS CLÉS : sémantique distributionnelle ; modèle de Markov caché ; arbre de dépendance ; représentation de mots ; classe sémantique ; sélection de variables ; trace Lasso.

# Contents

# INTRODUCTION

IN THE LAST twenty years, statistical machine learning has made tremendeous progress and enjoyed great success. This was possible thanks to different factors. First, the quantity of data, both labeled and unlabeled, has exploded during this period. Second, the computational power needed to analyze all of these data is now available and cheap. Finally, statistical and optimization methods were developed to tackle the challenges arising when dealing with that volume of data.

In this thesis, I describe two contributions to machine learning. The first one is an application to natural language processing. I developed a probabilistic model that automatically infers the meaning of words, given large quantities of unlabeled textual data. More particularly, using this model, it is possible to determine that the word *cat* is closer to *dog* than *banana*. The second contribution is about variable selection: the explosion of the amount of data also means that these data live in spaces of higher and higher dimensions. It is thus necessary to develop methods that can select the important variables to solve the problem.

## Semantic class induction

The first part of this thesis present work I carried out on word representation, and more specifically, on how to automatically learn good representations from large quantity of unlabeled texts. The contributions described in the first part of this thesis were previously published in (Grave et al., 2013b) and (Grave et al., 2013a).

### Motivations

Nowadays, most natural language processing systems are based on machine learning. The first step in designing such systems is to find a way to represent words as mathematical objects, often vectors, that can be fed into the machine learning algorithm. The simplest way to represent words is to associate a different integer to each word of the vocabulary, thus viewing words as discrete symbols. The word associated to the integer $i$ can also be represented by the high dimensional vector $\mathbf{e}_i$, which is equal to zero everywhere except the $i^{\text{th}}$ coefficient which is equal to one. For large vocabularies, the dimension of vectors associated to words is extremely high. Indeed, it is now common to deal with several hundreds of thousands of words, or even several millions of words. Unfortunately, statistical methods would need an unrealistic

quantity of labeled data to be trained with such high dimensional representations. A second serious limitation of this kind of representation is the fact that it is impossible to compare different words: computing the scalar product, or any other similarity measure, would give the same result for all the pairs of vectors representing two different words. In particular, this is a problem for words that were not seen in the training data: at test time, the algorithm has no way of determining how to model those words.

The first solution to these limitations that was proposed, even before the rise of statistical methods for natural language processing, is to design word features that capture the information needed to perform the underlying task. Examples of such features are prefixes and suffixes, such as *un-* for the adjective *unhappy* or *-ed* for the verb *wanted*, or shape features, such as features indicating if the word in capitalized or not, if it contains digits or hyphens, etc. This solution has proven to be quite effective for a large number of tasks such as part-of-speech tagging, syntactic parsing or named entity recognition. Unfortunately, for each new task, or each new domain or language, this set of features has to be redesigned by hand, which is time consuming and requires expert knowledge. For example, a very efficient feature for named entity recognition in English is whether a word is capitalized or not. This feature is completely useless for German, in which all nouns are capitalized, or Arabic, in which there are no capital letters.

A second approach to word representation is to perform word clustering, in order to drastically reduce the size of the vocabulary. The idea is to use large quantity of unlabeled texts and to cluster words based on their usage patterns and their contexts. The seminal work of Brown et al. (1992) proposed such a clustering algorithm, known as Brown clustering, which is still used nowadays and has been successfully applied to tasks such as named entity recognition or syntactic parsing. In that case, each word belongs to exactly one cluster, and can be represented by its corresponding cluster symbol. This representation still has some limitations: words belonging to different clusters still cannot be compared, while words belonging to the same cluster are considered equivalent. Another important limitation is the fact that each word only belongs to one cluster, and polysemy is thus ignored.

Following the approach proposed by Brown et al. (1992), we propose a new method to learn word representations using large quantities of unlabeled texts. Our method take into account the syntax and polysemy, since they both are important characteristics of natural languages. Finally, we compare discrete and continuous representations and demonstrate that continuous ones work better since they allow to compute word similarities.

## Hidden Markov tree models for semantic class induction

In Chapter 2, I introduce a new method for learning word representations using large quantities of unlabeled sentences. This method is based on a probabilistic model of sentences, with latent classes which aim at capturing the meaning of words. According to our model, these latent classes are generated by a Markov process on a syntactic dependency tree (De Marneffe and

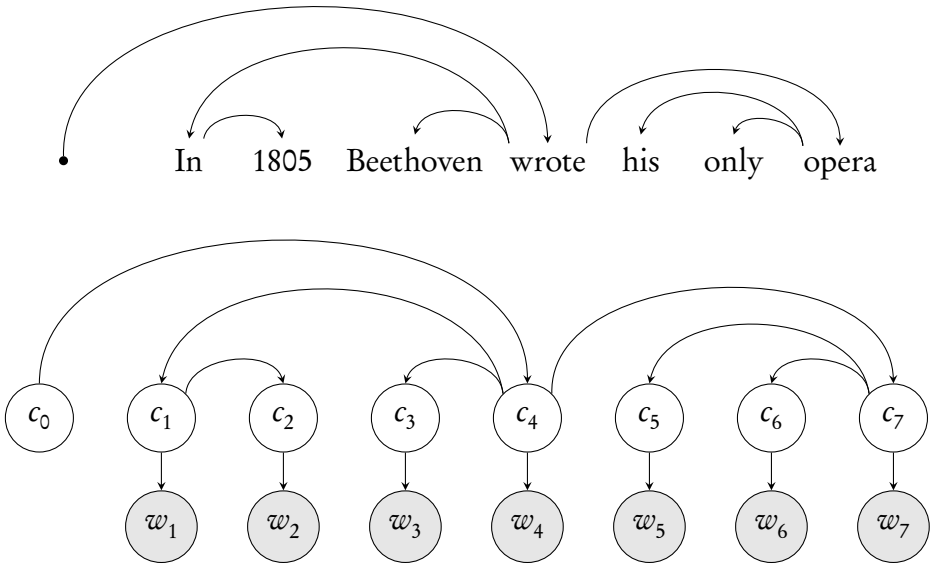



Figure 1: Example of a dependency tree and its corresponding graphical model.

Manning, 2008). Syntactic dependency trees capture the grammatical relations between words. Thus, using such trees allows our model to take the syntax into account. More formally, given a function $\pi$ such that $\pi(i)$ is the parent of the $i^{\text{th}}$ word of the sentence, the joint probability distribution on words $\mathbf{w} = (w_1, ..., w_K)$ and semantic classes $\mathbf{c} = (c_1, ..., c_K)$ can be factorized as

$$p(\mathbf{w},\ \mathbf{c}) = \prod_{k=1}^{K} p(c_k \mid c_{\pi(k)})\ p(w_k \mid c_k),$$

where both transition and emission probability distributions are multinomial distributions. Our model is thus a hidden Markov model, built on a dependency tree instead of a linear chain between the latent classes. See Figure 1 for a graphical representation of our model for the sentence

*In 1805 Beethoven wrote his only opera.*

In order to learn the parameters of our hidden Markov models, we consider the maximum likelihood estimator. Since we want to train our models on very large quantities of sentences, we use the online variant of the expectation-maximization algorithm, proposed by Cappé and Moulines (2009). We also want to learn models with large numbers of latent classes, of the order of 500, and since the complexity of the exact message passing algorithm is quadratic with respect to the number of latent classes, we propose to use an approximate variant which has a complexity of $O(n \log n)$, where $n$ is the number of classes.

## Intrinsic evaluations of our models

In Chapter 3, we evaluate our models on various classical tasks used for the evaluation of distributional models of semantics. The first one is predicting human similarity judgement. Pairs of words are presented to human subjects who are asked to rate their relatedness on a $0-10$ scale. Then, the vectorial representations obtained using our models are used to compute the similarity between the same pairs of words. The Pearson correlation coefficient between human similarity and model similarity scores is then computed to evaluate how well our models capture word similarity. We compare different similarity measures, such as the cosine, the Kullback-Leibler divergence or the Hellinger distance. We also evaluate our models on the BLESS dataset, a dataset introduced in order to determine what kind of semantic relations between words are favored by distributional models of semantics.

The second task we use to evaluate our models is a word categorization task. Given a set of words, the goal is to cluster them into groups that are semantically relevant. For example, given words refering to animals, vehicles and tools, the goal is to find those three clusters. We consider three datasets on which we evaluate our models, the first one is composed of concrete nouns, the second one is composed of concrete and abstract nouns and the third one is composed of verbs.

Finally, we evaluate our models on composition tasks. Natural languages verify the principle of compositionality: the meaning of a complex expression is a function of the meaning of its parts and the syntactic relations between them. It is thus believed that good word representations should follow the same principle, and in particular that it should be possible to compose representations of individual words in order to obtain representations of complex expressions. For this task, human subjects were presented pairs of phrases, and were asked to rate their semantic similarity. Our models are then used to compute the similarity between the same pairs of phrases. Again, the Pearson correlation coefficient is used to determine how well the models capture the semantic similarity of phrases. We evaluate our models on three datasets, comprising adjective-noun phrases, noun-noun phrases, verb-object phrases and subject-verb-object triples.

## Semi-supervised learning

In Chapter 4, we evaluate our models on two extrinsic tasks: semi-supervised named entity recognition and supersense tagging. As we said before, having a good word representation or good features is essential for supervised machine learning methods applied to natural language processing. It is thus common to first learn such a word representation on unlabeled data, and then use it as features for the supervised task.

We conduct experiments on two tasks. The first one, named entity recognition, is a part of information retrieval and consists in detecting and classifiying named entities, such as names of places, persons or organizations. Since there is a lot of different entities, many of them were not

seen in training data. It thus helps to learn a word representation to reduce the errors on those unknown words. The second task, supersense tagging, is a very coarse word sense disambiguation task. Building a system that can disambiguate all words is challenging because of the very large number of different word senses. Thus, it was proposed to reduce the number of word senses by clustering them. Supersenses, also known as lexicographer classes in WordNet, are an example of such coarse word senses. There is forty five supersenses, mainly for nouns and verbs. These experiments demonstrate that continuous vectorial representations, sometimes known as distributed representation, work better than atomic ones. We also show that context dependent representations usually outperform context independent representation. Finally, using the syntax slightly improves the results.

# Feature selection

The second part of this thesis present the work I did on feature selection. The contributions described in the second part of this thesis were previously published in (Grave et al., 2011).

## Motivations

In most statistical learning methods, the data are described by a set of features, or variables, such as word frequencies for text data or wavelet coefficients for image data. For some problems, it is desirable that the learnt or estimated model only depends on a small subset of those variables: this is the problem of variable or feature selection. Since models are often described by a vector of parameters, the problem of variable selection is closely related to the concept of parsimony: in particular, the problem of variable selection often reduces to estimating sparse parameter vectors. The first motivation for variable selection is interpretability: a model in which only a few variables are used is easier to understand. This is very important for applications such as medicine, economics, etc. Another important motivation is the fact that in many cases, the true model is sparse, or can be approximated by a sparse vector. It is thus more efficient, from a statistical perspective to estimate a sparse model.

The most natural way to perform variable selection is to constrain the estimator to use only $k$ variables. This problem, in the case of least squares regression, is known as best subset selection. Unfortunately, it is a hard combinatorial problem and is intractable in practice. Thus, greedy approximate algorithms such as matching pursuit were proposed to solve this problem. Another approach is to replace the constraint on the *number* of selected variables by its convex surrogate, which is the $\ell_1$-norm. In the case of least squares regression, this new estimator is known as the Lasso, or basis pursuit, and is defined by

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1.$$

It was proven that, under certain conditions on the design matrix $\mathbf{X}$ and the support of the true vector $\mathbf{w}^*$, this estimator can recover the exact support of $\mathbf{w}^*$, even if the dimension of the

problem is much larger than the number of observations.

Unfortunately, those conditions are not met in practice for certain problems, and in particular for problems where certain features are highly correlated. In that case, the Lasso can be unstable, selecting randomly one variable out of a group of highly correlated variables. In particular, this makes the interpretation of results difficult, or even misleading. We thus introduce a new estimator performing variable selection, in the case of highly correlated variables.

## Trace Lasso

In Chapter 7, we propose a new penalty function to perform variable selection in the case of highly correlated variables. It is based on the trace norm $||\mathbf{M}||_*$, which is equal to the sum of the singular values of the matrix $\mathbf{M}$. More precisely, this new penalty $\Omega_{\mathbf{X}}$ is called the trace Lasso and is equal to

$$\Omega_{\mathbf{X}}(\mathbf{w}) = ||\mathbf{X}\mathrm{Diag}(\mathbf{w})||_*,$$

where $\mathbf{X} \in \mathbb{R}^{n \times p}$ is the design matrix of the problem and $\mathrm{Diag}(\mathbf{w}) \in \mathbb{R}^{p \times p}$ is a matrix whose diagonal is equal to $\mathbf{w}$ and which is equal to zero everywhere else. Multiplying the design matrix $\mathbf{X}$ by $\mathrm{Diag}(\mathbf{w})$ is equivalent to multiply each column of $\mathbf{X}$ by the corresponding coefficient of $\mathbf{w}$. The idea behind the trace Lasso is the fact that another measure of complexity of a model, besides the *number* of selected variables, is the *rank* of the selected variables. In that case, adding a variable which is in the span of the already selected variables does not increase the complexity of the model. Since the trace norm is a convex surrogate of the rank, the trace Lasso is a convex surrogate of the rank of the selected variables.

The trace Lasso estimator has interesting properties that we now describe. First, the value of the trace Lasso regularizer only depends on the matrix $\mathbf{X}^\top\mathbf{X}$. If all the variables are orthogonal, then the trace Lasso regularizer is equal to the $\ell_1$-norm. On the other hand, if all the variables are equal, then the trace Lasso regularizer is equal to the $\ell_2$-norm. The group Lasso with non-overlapping groups can also be expressed as a special case of the trace Lasso regularizer. Moreover, the trace Lasso regularizer is a norm that is always comprised between the $\ell_1$ and the $\ell_2$-norms, meaning that it interpolates between those two norms, based on the correlation structure of the design matrix. In order to illustrate this behavior, we plotted in Figure 2 the unit balls for the trace Lasso norm for different values of $\mathbf{X}^\top\mathbf{X}$:

$$\mathbf{X}^\top\mathbf{X} = \begin{pmatrix} 1.0 & \rho & \rho \\ \rho & 1.0 & \rho \\ \rho & \rho & 1.0 \end{pmatrix},$$

where $\rho \in \{0.0,\ 0.5,\ 0.7,\ 0.9,\ 1.0\}$. We observe that the unit ball is smoothly deformed, from the $\ell_1$-ball to the $\ell_2$-ball.

Second, if the loss function $\ell : (\mathbf{x}_i^\top\mathbf{w},\ y_i) \mapsto \ell(\mathbf{x}_i^\top\mathbf{w},\ y_i)$ used with the trace Lasso regularizer is strongly convex with respect to its first argument, then, the minimum of the regularized

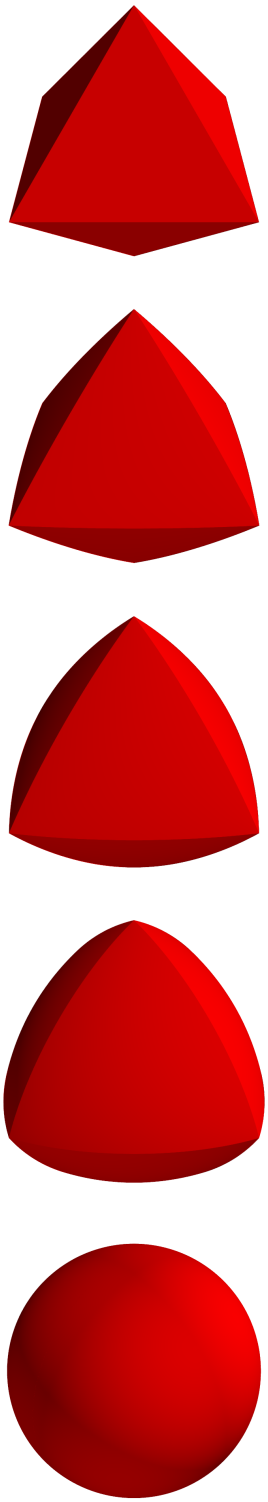


Figure 2: Unit balls of the trace Lasso norm for various value of $\mathbf{X}^\top\mathbf{X}$. See the text for details.

empirical risk minimization problem is unique. This is not the case for the Lasso, and we believe this is a first step towards stability. In particular, the trace Lasso regularizer is strongly convex in the flat directions of the loss function, that is, the directions belonging to the kernel of the design matrix $\mathbf{X}$.

We also introduce two algorithms to compute the optimum of least squares regression problems regularized by the trace Lasso norm. The first one is based on a variational formulation of the trace norm and belongs to the class of iteratively reweighted least squares algorithms. The second one is based on the alternating direction method of multipliers. We then perform experiments to compare those two optimization methods and to compare the trace Lasso with other estimators in the case of highly correlated designs.

# RÉSUMÉ

DURANT CES vingt dernières années, l'apprentissage statistique a fait des progrès considérables et a connu de grands succès. Cela fut possible grâce a différent facteurs. Premièrement, la quantité de données, à la fois étiquetées et non-étiquetées, a explosé pendant cette période. Deuxièmement, la capacité de calcul nécessaire pour analyser toutes ces données est maintenant disponible à bas coût. Enfin, de nouvelles méthodes statistiques et de nouveaux algorithmes pour l'optimisation ont été dévelopées afin de pouvoir traiter ces grands volumes de données.

Dans cette thèse, je présente deux contributions que j'ai realisées dans le domaine de l'apprentissage automatique. La première est appliquée au traitement automatique de la langue naturelle. J'ai développé un modèle probabiliste permettant d'inférer le sens des mots automatiquement, à partir de grandes quantités de texte brut. Plus précisément, il est possible, grâce à ce modèle, de déterminer que le mot *chat* est plus proche de *chien* que de *banane*. La deuxième contribution concerne la sélection de variables : en effet, le nombre de variables utilisées pour décrire les données a explosé, parallélement à la quantité des données. Il est donc important de développer des méthodes qui sélectionnent automatiqument les variables importantes pour résoudre un problème.

## Induction de classes sémantiques

La première partie de cette thèse présente le travail que j'ai effectué sur la représentation de mots, et plus précisément, sur l'apprentissage automatique de telles représentation à partir de grandes quantités de texte brut.

### Motivations

De nos jours, la plupart des systèmes de traitement automatique des langues reposent sur l'apprentissage automatique. La première étape dans la création de tels systèmes est de trouver un moyen de représenter les mots comme des objets mathématiques qui peuvent être traités par un algorithme d'apprentissage automatique. La manière la plus simple de représenter les mots est d'associer un entier différent a chacun des mots du vocabulaire. Il est alors possible de représenter le mot associé à l'entier $i$ par le vecteur de la base canonique $e_i$. La dimension d'une telle représentation est extrêmement élevée pour de grands vocabulaire. En effet, il n'est pas

rare d'avoir des centaines de millier, voir des millions, de mots différents dans un vocabulaire. Malheureusement, la quantité de données nécessaire pour entrainer un système fondé sur l'apprentissage statistique est beaucoup trop importante lorsque la dimension des données est aussi grande. Une autre limite importante de ce type de représentation est le fait qu'il est impossible de comparer différents mots : en effet, calculer le produit scalaire, ou tout autre mesure de similarité, donne le même résultat quelque soit les mots considérés. C'est un problème important pour les mots non observés dans les données utilisées pour l'apprentissage. Il est en effet impossible pour l'algorithme de savoir comment traiter de tels mots inconnus.

La première solution qui fut considérée, est de créer des représentations pour les mots à la main, ces représentations encodant l'information nécessaire pour résoudre la tache considérée. Par exemple, de telles représentations peuvent être le préfixe ou le suffixe du mot, comme *in-* pour l'adjectif *invariable* ou *-ment* pour l'adverbe *automatiquement*. Elles peuvent aussi encoder la forme du mot, comme le fait que le mot commence par une majuscule, contienne des chiffres ou non, etc. Cette solution fut utilisée avec succès pour des taches telles que l'étiquetage morpho-syntaxique, l'analyse syntaxique ou encore la reconnaissance d'entités nommées. Malheureusement, pour chaque nouvelle tache, pour chaque nouveau domaine ou pour chaque nouvelle langue, cet ensemble de traits doit être redéfini à la main, ce qui demande du temps et de l'expertise. Par exemple, l'un des traits les plus efficaces pour la reconnaissance d'entités nommées en anglais est le fait qu'un mot commence par une majuscule ou non. Ce trait est complétement inutile en allemand car tous les noms ont une majuscule, ou en arabe, car aucun mot n'a de majuscule.

Une deuxième approche pour représenter les données textuelles consiste à regrouper les mots similaires au sein de classes, afin de fortement réduire la taille du vocabulaire. De grandes quantités de texte brut sont utilisées pour déterminer quels mots doivent être regroupés, en fonction des contextes dans lesquels ils apparaissent. Le travail de Brown et al. (1992) propose un tel algorithme, connu sous le nom de *Brown clustering*, qui est encore largement utilisé aujourd'hui. Dans ce cas, chaque mot appartient à exactement un groupe qui peut être utilisé pour représenter le mot. Cette représentation souffre encore de certaines limites : deux mots appartenant à des groupes différents ne peuvent toujours pas être comparés tandis que deux mots du même groupe sont considérés égaux. Une autre limite importante est le fait que la polysemie soit ignorée, car chaque mot appartient à un seul groupe.

Inspiré par l'approche suivie par Brown et al. (1992), nous proposons une nouvelle méthode permettant d'apprendre automatiquement à représenter les mots à partir de grandes quantités de texte brut. Cette méthode tient compte de caractéristiques importantes des langues naturelles, telles que la syntaxe et la polysémie. Enfin, nous comparons des représentations discrètes et des représentations continues et montrons que ces dernières donnent de meilleurs résultats. Cela est dû au fait qu'il est possible de comparer différents mots avec des représentations continues et non avec des représentations discrètes.

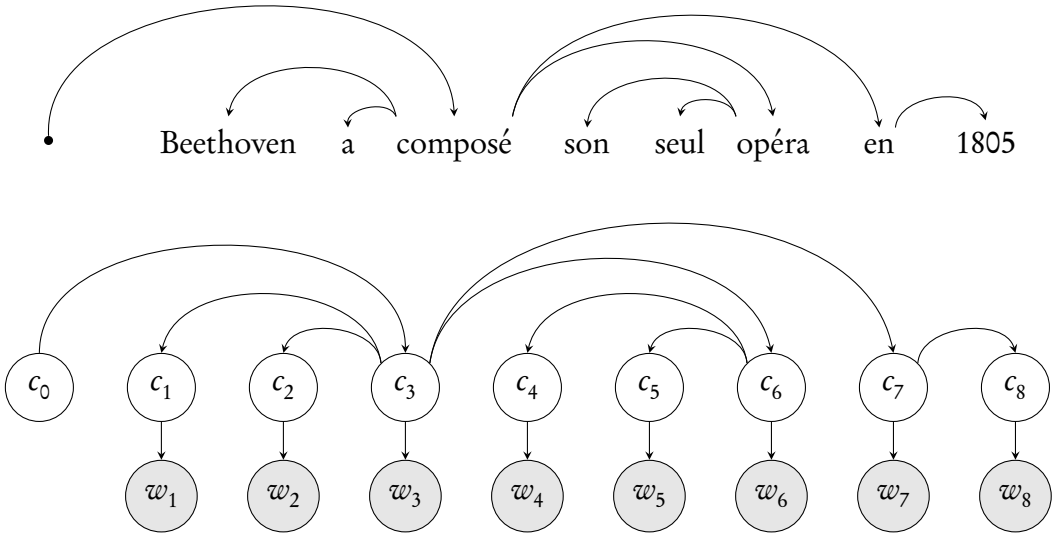



FIGURE 3 – Exemple d'arbre de dépendance et notre modèle graphique associé.

## Modèles de Markov cachés pour l'induction de classes sémantiques

Dans le chapitre 2, j'introduis une nouvelle méthode pour apprendre une représentation des mots à partir de grandes quantités de texte brut. Cette méthode repose sur un modèle probabiliste de la phrase, dans lequel des variables latentes représentent le sens des mots. Dans notre modèle, ces variables latentes sont générées à l'aide d'un processus de Markov sur un arbre de dépendance syntaxique. Les arbres de dépendance représentent les relations grammaticales existant entre les mots. L'utilisation de tels arbres permet de prendre en compte la syntaxe. Plus formellement, étant donné une fonction $\pi$ telle que $\pi(i)$ est le parent du $i$ème mot de la phrase, la probabilité jointe des mots $\mathbf{w} = (w_1, ..., w_K)$ et des classes sémantiques $\mathbf{c} = (c_1, ..., c_K)$ se factorise en :

$$p(\mathbf{w},\ \mathbf{c}) = \prod_{k=1}^{K} p(c_k \mid c_{\pi(k)})\ p(w_k \mid c_k),$$

où les probabilités de transition et d'émission suivent toutes deux une distribution multinomiale. Notre modèle est donc juste un modèle de Markov caché, dans lequel la chaîne entre les variables latentes est remplacée par un arbre de dépendance. Voir la figure 3 pour une représentation graphique de notre modèle pour la phrase

*Beethoven a composé son seul opéra en 1805.*

Nous utilisons l'estimateur du maximum de vraisemblance afin d'apprendre les paramètres de notre modèle de Markov caché. Nous utilisons la variante en ligne de l'algorithme espérance-maximisation, proposé par Cappé and Moulines (2009), afin de pouvoir entrainer notre modèle en utilisant un grand nombre de phrases. Nous souhaitons aussi entrainer des modèles avec un

grand nombre d'états cachés, tel que 512, et comme la complexité de l'algorithme pour l'inférence exacte est quadratique par rapport au nombre d'états cachés, nous proposons d'utiliser un algorithme approché, qui a une complexité de $O(n \log n)$, où $n$ est le nombre d'états cachés.

## Evaluations intrinsèques de notre modèle

Dans le chapitre 3, nous évaluons notre modèle sur diverses taches classiques, utilisées pour évaluer les modèles de sémantique distributionnelle. La première consiste à prédire des jugements de similarité humains. Des paires de mots sont présentées à des sujets humains, qui doivent noter leur similarité sur une échelle de 0 à 10. La représentation obtenue à l'aide de notre modèle est alors utilisée pour calculer la similarité entre les mêmes paires de mots. Le coefficient de corrélation de Pearson est alors calculé entre les similarités humaines et les similarités obtenues à l'aide du modèle, afin de déterminer si la similarité obtenue fait sens. Nous comparons différentes mesures de similarité, telle que le cosinus, la divergence de Kullback-Leibler ou encore la distance d'Hellinger. Nous évaluons aussi notre modèle sur le jeu de donnée BLESS, afin de déterminer quel type de relations entre mots est favorisé par notre modèle.

La deuxième tache que nous utilisons pour évaluer notre modèle est une tache de catégorisation. Étant donné un ensemble de mots, le but est de trouver une partition de ces mots qui soit sémantiquement cohérente. Par exemple, étant donnés des mots faisant référence à des animaux, des véhicules et des outils, le but est de retrouver ces trois groupes sémantiques. Nous considérons trois jeux de données, le premier comprenant des noms faisant référence à des entités concrètes, le deuxième comprenant des noms faisant référence à des entités concrètes et abstraites et le dernier comprenant des verbes.

Enfin, nous evaluons notre modèle sur des taches utilisant la composition. Les langues naturelles vérifient le principe de compositionnalité : le sens d'une expression complexe peut être obtenu en fonction du sens de ses parties et des relations syntaxique entre elles. Il est donc communément admis qu'une bonne représentation des mots doivent suivre le même principe, et donc qu'il soit possible de composer les représentations des mots afin d'obtenir des représentation d'expressions complexes. Pour cette tache, des paires de groupes nominaux ou verbaux ont été présentées à des sujets humains qui ont noté leur similarités. Notre modèle est alors utilisé pour comparer les mêmes expressions. De nouveau, le coefficient de corrélation de Pearson est utilisé pour déterminer la qualité de la mesure de similarité obtenue. Nous évaluons notre modèle sur trois jeux de données comprenant : des groups nominaux et verbaux et des triplets sujet-verbe-objet.

## Apprentissage semi-supervisé

Dans le chapitre 4, nous évaluons notre modèle sur deux tache extrinsèques : la reconnaissance d'entités nommées et l'étiquetage en super-sens. Comme nous l'avons souligné précédemment, avoir une bonne représentation des mots ou de bons traits est essentiel pour l'apprentissage supervisé appliqué au traitement des langues naturelles. Il est donc commun d'apprendre une

telle représentation de manière non-supervisée dans un premier temps et de se servir de cette représentation dans l'algorithme supervisé dans un second temps.

La première tache considérée, la reconnaissance d'entités nommées, fait partie du domaine de la recherche d'information. Elle consiste à détecter et classifier les entités nommées telles que noms de lieux, de personnes ou d'organisations. Un grand nombre de ces entités nommées ne sont pas observées dans les données d'apprentissage et il est donc très utile d'apprendre une représentation des mots pour réduire les erreurs commises sur ces mots inconnus. La seconde tache, l'étiquetage en super-sens, est une tache de désambiguation très grossière. Créer un système pouvant désambiguer tous les mots est compliqué car il y a un très grand nombre de sens différents. Il fut donc proposé de réduire le nombre de sens par regroupement. Les super-sens sont un exemple de tel regroupement. Il y a quarante-cinq super-sens différents, principalement pour les noms et les verbes. Nous montrons que les représentations continues obtiennent de meilleurs résultats que les représentations atomiques. De même, dans la plupart des cas, l'utilisation d'un arbre de dépendance améliore les performances par rapport à l'utilisation d'une chaîne.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Paris | art | président | an | blanc | publier | dire |
| Londres | histoire | directeur | mois | noir | écrire | penser |
| Rome | économie | membre | jours | rouge | sortir | savoir |
| Berlin | science | professeur | heure | petit | jouer | noter |
| Lyon | recherche | ministre | minute | vert | enregistrer | estimer |
| Marseille | musique | secrétaire | semaine | bleu | apparaître | considérer |
| Montréal | cinéma | député | année | jaune | composer | affirmer |

TABLE 1 – Exemples de classes sémantiques obtenues. (Corpus : Wikipedia français).

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Roma | filosofia | presidente | anno | conquistare | bianco | scrivere |
| Milano | arte | direttore | giorno | lasciare | nero | interpretare |
| Venezia | storia | membro | mese | occupare | colore | suonare |
| Torino | scienze | capo | tempo | sconfiggere | rosso | pubblicare |
| Firenze | medicina | professore | ora | attaccare | blu | cantare |
| Parigi | teologia | segretario | minuto | distruggere | scuro | realizzare |
| Napoli | arti | generale | episodio | abbandonare | verde | fare |

TABLE 2 – Exemples de classes sémantiques obtenues. (Corpus : Wikipedia italien).

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| membrane | cell | association | treatment | analyze | increase |
| nucleus | lymphocyte | relationship | exposure | evaluate | change |
| surface | fibroblast | correlation | stimulation | assess | reduction |
| cytoplasm | macrophage | difference | injection | examine | decrease |
| tissue | progenitor | interaction | administration | determine | difference |
| structure | xenograft | relation | transfection | investigate | improvement |
| matrix | hepatocyte | link | incubation | measure | variation |

TABLE 3 – Exemples de classes sémantiques obtenues. (Corpus : articles biomedicaux).

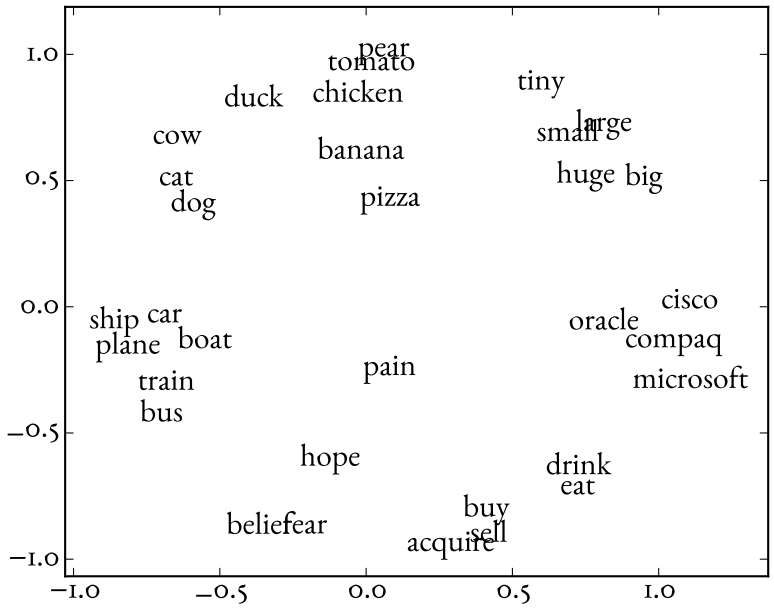



FIGURE 4 – Représentation vectorielle des mots obtenue à l'aide de notre modèle.

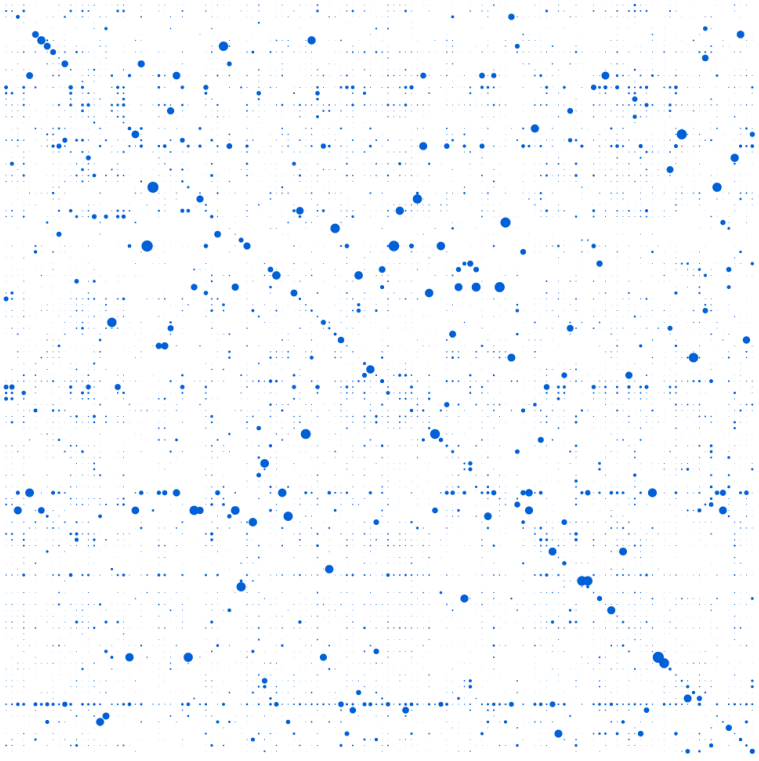


FIGURE 5 – Matrice de transition entre les classes sémantiques.

# Sélection de variables

La seconde partie de cette thèse présente le travail que j'ai effectué sur la sélection de variables.

## Motivations

Dans la plupart des algorithmes d'apprentissage statistique, les données sont représentées par un ensemble de variables, ou traits, telles que la fréquence des mots pour les données textuelles ou les coefficients de la transformée en ondelettes pour les images. Pour certains problèmes, il est désirable que le modèle appris ne depende que d'un petit sous-ensemble de ces variables : c'est le problème de sélection de variables. Comme les modèles sont souvent décrits par un vecteur de paramètres, le problème de selection de variables est souvent équivalent à un problème d'estimation avec une contrainte de parcimonie sur le vecteur de paramètres. Une motivation importante pour la sélection de variables est l'interprétabilité : un modèle dans lequel seulement un petit nombre de variables sont utilisées est plus facile à interpréter et à comprendre. Cela est très important pour des applications telles que l'économie ou la médecine. Une autre motivation importante est le fait que dans de nombreux cas, le vrai modèle est effectivement parcimonieux, ou peut être approché par un modèle parcimonieux. Dans ce cas, il est plus efficace, d'un point de vue statistique, d'estimer un modèle parcimonieux.

La méthode la plus naturelle pour la sélection de variables est de contraindre le modèle à n'utiliser que $k$ variables. Dans le cas de la régression aux moindres carrés, ce problème s'appelle *best subset selection*. Malheureusement, en général, c'est un problème combinatoire difficile qu'il est impossible de résoudre en pratique. C'est pourquoi des algorithmes approchés gloutons, tels que *matching pursuit*, ont été proposés pour résoudre ce problème. Une autre approche consiste à relacher la contrainte sur le nombre de variables sélectionnées en utilisant une contrainte convexe faisant intervenir la norme $\ell_1$ du vecteur de paramètres. Dans le cas de la régression aux moindres carrés, ce nouvel estimateur est connu sous le nom de *Lasso* ou *basis pursuit* et est défini par :

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1.$$

Il est possible de démontrer que, sous certaines conditions sur la matrice de design $\mathbf{X}$ et le support du vrai vecteur $\mathbf{w}^*$, cet estimateur peut retrouver le support exact de $\mathbf{w}^*$, même dans le cas où le nombre d'observations est beaucoup plus petit que la dimension du problème.

Malheureusement, ces conditions ne sont pas vérifiées en pratique pour certains problèmes, et en particulier pour les problèmes dans lesquels certaines variables sont fortement corrélées. Dans ce cas, le Lasso peut être instable, selectionnant au hasard une variable d'un groupe de variables fortement corrélées. En particulier, cela rend l'interprétation des modèles obtenus difficile voir même dangereuse. C'est pourquoi, nous introduisons un nouvel estimateur faisant de la sélection de variables en présence de prédicteurs fortement corrélés.

## Trace Lasso

Dans le chapitre 7, j'introduis une nouvelle pénalité qui sélectionne les variables en présence de forte corrélations entre les prédicteurs. Cette pénalité repose sur la norme trace $||\mathbf{M}||_*$, qui est égale à la somme des valeurs singulières de la matrice $\mathbf{M}$. Plus précisement, cette nouvelle pénalité $\Omega_X$ s'appelle *trace Lasso* et est définie par :

$$\Omega_{\mathbf{X}}(\mathbf{w}) = ||\mathbf{X}\mathrm{Diag}(\mathbf{w})||_*,$$

où $\mathbf{X} \in \mathbb{R}^{n \times p}$ est la matrice de design et $\mathrm{Diag}(\mathbf{w}) \in \mathbb{R}^{p \times p}$ est une matrice dont la diagonale est égale à $\mathbf{w}$. Multiplier $\mathbf{X}$ par $\mathrm{Diag}(\mathbf{w})$ est équivalent à multiplier chaque colonne de $\mathbf{X}$ par le coefficient de $\mathbf{w}$ correspondant. L'intuition derrière cette pénalité est la suivante : une mesure de complexité d'un modèle autre que le nombre de variables sélectionnées est la dimension du sous-espace engendré par ces variables, autrement dit, leur rang. Or la norme trace est une relaxation convexe du rang.

Le trace Lasso a des propriétés intéressantes que nous décrivons maintenant. Premièrement, la valeur de cette pénalité dépend uniquement de la matrice $\mathbf{X}^\top\mathbf{X}$ (et de $\mathbf{w}$ évidemment). Si toutes les variables sont orthogonales, alors le trace Lasso est égal à la norme $\ell_1$. Au contraire, si toutes les variables sont égales, alors le trace Lasso est égal à la norme $\ell_2$. Le *group Lasso* peut aussi s'écrire comme un cas particulier du trace Lasso. De plus, le trace Lasso est toujours compris entre la norme $\ell_1$ et la norme $\ell_2$. Cela signifie qu'il interpole ces deux normes, en fonction de la structure de corrélation des variables. Afin d'illustrer ce comportement, j'ai représenté les boules unités du trace Lasso pour différentes valeurs de $\mathbf{X}\top\mathbf{X}$ :

$$\mathbf{X}^\top\mathbf{X} = \begin{pmatrix} 1.0 & \rho & \rho \\ \rho & 1.0 & \rho \\ \rho & \rho & 1.0 \end{pmatrix},$$

où $\rho \in \{0.0,\ 0.5,\ 0.7,\ 0.9,\ 1.0\}$, à la figure 6. Nous observons que la boule unité est continuement déformée, de la boule $\ell_1$ à la boule $\ell_2$.

Deuxièmement, si la fonction de perte $\ell : (\mathbf{x}_i^\top\mathbf{w},\ y_i) \mapsto \ell(\mathbf{x}_i^\top\mathbf{w},\ y_i)$ utilisée avec le trace Lasso est fortement convexe, alors le minimum du risque empirique regularisé est unique. Cela n'est pas le cas pour la norme $\ell_1$, et je pense que c'est une première étape vers la stabilité. En particulier, le trace Lasso est fortement convexe dans les directions plates de la fonction de perte, c'est à dire, les directions appartenant au noyau de la matrice de design $\mathbf{X}$.

Finalement, je présente deux algorithmes pour calculer le minimum du problème de régression aux moindres carrés regularisé par le trace Lasso. Le premier repose sur une formulation variationnelle de la norme trace et appartient à la classe d'algorithmes connus sous le nom d'*iteratively reweighted least squares*. Le deuxième repose sur la méthode *alternating direction method of multipliers*. J'ai réalisé des expériences sur des données synthétiques, afin de comparer les deux algorithmes d'optimisation et le trace Lasso avec d'autres estimateurs classiques.

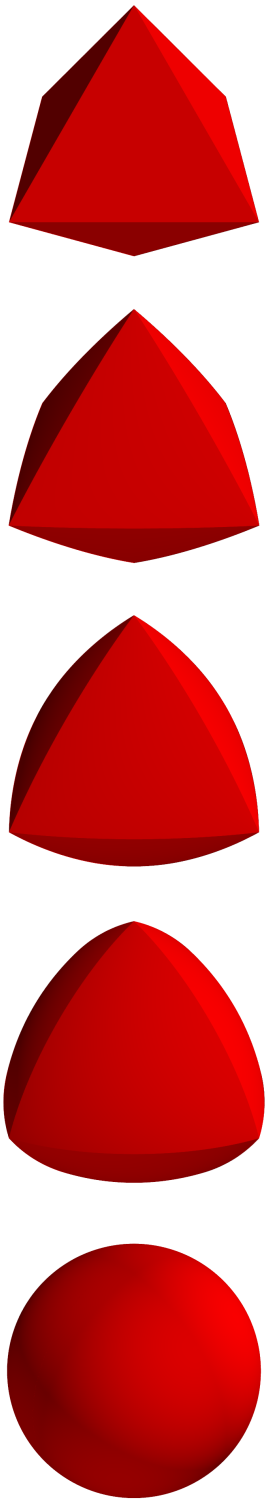


FIGURE 6 – Boules unité du trace Lasso pour diverses valeurs de $\mathbf{X}^\top\mathbf{X}$. Voir le texte pour les détails.

# Part I

# Distributional semantics

# Chapter 1



# A brief introduction to natural language processing

Languages (both natural and artificial) can be modelled as being formed by combining a lexicon and a grammar. A lexicon is a set of words and the specific information associated to each word, such as its irregular forms. Sometimes, for natural languages, it also includes idiomatic expressions. A grammar, on the other hand, can be thought as the set of rules that describe how words can be assembled together to form complex units such as phrases or sentences. Depending on the formalism being used, the quantity of information belonging to the lexicon or the grammar can be more or less important. Lexicons, grammars and their associated concepts are essential tools for designing natural language processing systems, and we thus describe them in greater details in the following. We also discuss a remarkable property of natural languages: while the lexicon and the grammar can be considered finite, it is possible to express a potentially infinite number of ideas. Finally, we introduce the field of distributional semantics, which is the study of methods that aim at learning automatically the meaning of natural language expressions, based on large quantities of textual data. We refer the reader to Jurafsky and Martin (2000) or Manning and Schütze (1999) for more detailed introductions to natural language processing.

## 1.1 What is a word?

This question might seem rather naive, but properly defining what is a word, and other related concepts, is essential to develop natural language processing systems. The smallest grammatical element of a natural language is called a morpheme, while a word, which is made up of one or multiple morphemes is the smallest grammatical element of a language that can be used alone. For example, the word *seeking* is made up of two morphemes: *seek* and *-ing*. The morpheme *seek*, which can be used independently as a word, is called a free morpheme, while the morpheme *-ing*, which cannot be used independently as a word, is called a bound morpheme. Finally, the stem is the main morpheme of a word, the one from which the meaning is derived.

The stem of the word *seeking* is the morpheme *seek*. Stems are not necessarily words: for example, the stem of the French verb *supprimer* is the morpheme *supprim-*, which is not a word. Given a word, determining its corresponding stem is called stemming.

### 1.1.1 Word, form and lemma

Now, a new question arises: are *bird* and *birds* the same word? What about *be*, *is* and *was*? In fact, *bird* and *birds* are different surface forms of the same lexeme: a lexeme is a set of words, which are inflected variants of the same word, and thus share the same meaning. In a dictionary, a lexeme corresponds to an entry of the dictionary: for example each verb only have one entry, and not one for each of its inflected form. Each lexeme is represented by a particular form, called the lemma, for example the infinitive for verbs. Given a form, determining its corresponding lemma is called lemmatization.

It is important to note that lemmatization and stemming are two different operations, even if they are often used for the same purpose: reducing the size of the dictionary. For example *was* and *are* have the same lemma, *be*, but different stems. On the other hand, *unhappy* and *happiness* have the same stem, *happy*, but different lemmas.

### 1.1.2 From morphemes to words: a bit of morphology

Studying how morphemes assemble together to form words is called morphology, and is part of the grammar of a language. There are four main ways to combine morphemes to obtain words: inflection, derivation, compounding and cliticization. We will now present each of these four constructions.

Inflection occurs when a word is modified in order to express a grammatical category such as the number, the gender, the case, the tense or the voice. Inflection for verbs is known as conjugation, while inflection for nouns is known as declension. For example in english, the verb *to look* can be inflected as *looks* to express the third person singular or as *looked* to express the past. These two examples follow rules: the third person singular is obtained by using the suffix *-s*, while the past is obtained by using the suffix *-ed*. Sometimes, inflection does not follow rules, and is known as irregular inflection. An example of irregular inflection is the past of the verb *to drink* which is *drank*. Some words, such as adjectives in english, are never inflected: these words are known as invariant.

Derivation occurs when a word, a verb for example, is used to construct a new word, a noun for example, whose meaning is different but related to the meaning of the initial word. Examples of derivation is constructing the noun *computation* from the verb *to compute* or the french adverb *invariablement* from the adjective *invariable*. The adjective *invariable* is itself derived from the adjective *variable*, and thus derivation can happen between words of the same class. Derivation often implies adding a suffix, such as *-ation* or *-ment* in the previous examples or a prefix, such as *in-*.

Using two stems, such as *cat* and *walk*, to construct a new word, *catwalk*, is known as compounding. It is opposed to derivation by the fact that a compound is obtained by composing two stems (which are free morphemes), while derivation involves bound morphemes, such as *-ly* to form adverbs or *un-* to form antonyms. Finally cliticization is obtained by combining a word with a clitic. A clitic is a morpheme that has the syntactic role of a word: for example in the English sentence *he's tall*, the verb *is* is reduced to the clitic *'s*. Or in the French sentence *J'ai un frère*, the subject *je* is reduced to the clitic *j'*.

### 1.1.3 The importance of morphology for NLP

The extent to which different languages use inflection, derivation and compounding vary a lot. For example, in modern Chinese, words are almost never inflected, but a lot of them are compounds. English is weakly inflected: only nouns and verbs are inflected, and they do not have more than five different inflected forms. Latin languages, such as French, Italian or Spanish, are a bit more inflected: adjectives are also inflected and each verb can have more than forty different inflected forms. Finally, Slavic languages, such as Russian, Czech or Serbian are highly inflected. Each noun or adjective have six or seven cases and as much as three genders.

Thus, for moderately or highly inflected languages, the number of surface forms is greatly superior to the number of lemmas. Treating each surface form as a different word type leads to an explosion in the number of parameters and increase the problem of data sparsity. That is why most natural language processing pipelines start by doing some sort of morphological analysis of the words, that is, given a surface form, try to identify the corresponding lemma and the different grammatical categories, such as the gender, the case, the tense or the part-of-speech. We introduce parts-of-speech in the next section.

### 1.1.4 Parts-of-speech

Parts-of-speech, also known as word classes are grammatical categories of words based on syntactic functions played by the words and their morphological behavior. Example of parts-of-speech are noun, adjective, article, verb, adverb, pronoun, conjunction and participle. Some of this classes are called open classes, since new words can appear in those classes. For example, in English or French, the noun and verb classes are open. The other classes are called closed classes, since no new words can appear. For example, the class of coordinating conjunctions in French is closed (*mais, ou, et, donc, or, ni, car*). Finally, words with little semantic meaning and which mainly serve a grammatical purpose are called function words, as opposed to content words, which carry most of the meaning. Nouns, verbs or adjectives are examples of content words.

Given a sentence, finding the parts-of-speech associated to the words forming that sentence is called part-of-speech tagging. Looking up the words in a lexicon or a dictionnary is usually not sufficient because many words belong to different grammatical classes. For example, the English word *dive* can be either a noun or a verb, depending on the context. Similarly, the

French word *beau* can be either a noun or an adjective. Resolving such ambiguities cannot be done for each word independently since parts-of-speech depend on the syntax of the sentence. Morphological analysis and part-of-speech tagging are sometimes performed together, since the possible morphological interpretations of a word depends on its part-of-speech.

## 1.2 From words to sentences: syntax

In the previous section, we introduced various concepts applicable to words, taken as independent units of the language. Assembling words together to form valid sentences follows some rules, known as the syntax of the language. We will now discuss the two main formalisms used to describe the syntax that are widely used in natural language processing: constituency grammars and dependency grammars.

### 1.2.1 Constituency grammars

Constituency grammars are based on the notion of constituent, and how these constituents assemble together to form larger constituents. For example, the sentence

*My cat is chasing a mouse.*

is formed by assembling the noun phrase *My cat* and the verb phrase *is chasing a mouse*. Both these constituents can in turn be decomposed into simpler constituents: for example, the noun phrase *My cat* can be decomposed into the preposition *My* and the noun *cat*. Constituency grammars thus define rules on how this constituents might assemble together. For example, in English, a noun phrase might be formed by assembling a preposition and a noun, such as in the previous example or by assembling a determiner, an adjective and a noun such as in the noun phrase *a red apple*. On the other hand, a noun phrase cannot be formed by assembling a noun and a determiner. Thus, *man the* is not a valid noun phrase.

One way to formalize constituency grammars is by using context free grammars, that where first introduced by Chomsky (1956) for natural languages and rediscovered by Backus (1959) in the field of programming languages design. In the following definition, we formally introduce context free grammars.

**Definition 1.** *A context free grammar $\mathcal{G}$ is quadruple $\mathcal{G} = (N, \Sigma, R, S)$ where:*

- *$N$ is a set of non-terminal symbols,*
- *$\Sigma$ is a set of terminal symbols,*
- *$R \subset N \times (N \cup \Sigma)^*$ is a set of rewriting rules of the form $X \to Y_1 ... Y_k$, where $X \in N$ and each $Y_i \in N \cup \Sigma$,*
- *$\mathtt{S} \in N$ is a special symbol called the start symbol.*

In the case of natural language, the set of non-terminal symbols are symbols representing the various constituents, such as `S` for sentence, `NP` for noun phrase, `VP` for verb phrase or `NN` for common noun, while the set of terminal symbols $\Sigma$ are the words in the vocabulary of the language. Then, the rewriting rules encode what are the valid way to assemble constituents together. For example, the fact that noun phrases can be formed by a determiner and a noun corresponds to the rule

$$\texttt{NP} \rightarrow \texttt{DET NN}$$

and the fact that the word *cat* is a noun corresponds to the rule

$$\texttt{NN} \rightarrow \textit{cat}.$$

Then, the set of strings that can be obtained by using the context free grammar defines the valid sentences of the language. Let us now describe more formally how the CFG is used to generate sentences.

**Definition 2.** *A left-most derivation is a sequence of strings $s_1 \rightarrow ... \rightarrow s_n$ where:*

- $s_1 = \texttt{S}$, i.e. *the first string is equal to the start symbol,*
- $s_n \in \Sigma^*$, i.e. *the last string only contains terminal symbols,*
- *Each $s_i$, $i \in \{2, ..., n\}$ is derived from $s_{i-1}$ by replacing the left-most non terminal symbol $X$ by $Y_1...Y_k$ where the rule $X \rightarrow Y_1...Y_k$ belongs to the set of rewriting rules $R$.*

The valid sentences of the language are then defined as the set of string $s \in \Sigma^*$ made of words of the vocabulary such that there exists a left-most derivation $s_1 \rightarrow ... \rightarrow s_n = s$ generating the sentence. Derivations are often represented as trees, where the root of the tree is the start symbol `S`, each internal node of the tree is a non-terminal symbol and the leaves of the tree are terminal symbols. The children of each node are the symbols that are obtained by applying the rewriting rule to the corresponding node. Given a sentence, finding a left-most derivation that generates that sentence is called syntactic analysis or syntactic parsing.

In context free grammars encoding natural languages, there exist multiple derivations that generate the same sentence, because of ambiguity. It is thus necessary to introduce probabilistic context free grammar (PCFG) to resolve these ambiguities. In a PCFG, each rewriting rule has a certain probability of being applied given the most-left non terminal symbol. The probability of a derivation is thus the product of the probabilities of the rules that are applied during the derivation. Given an ambiguous sentence that has multiple possible derivations, it is thus possible to choose the most probable one, in order to resolve the ambiguity.

### 1.2.2 Dependency grammars

Dependency grammars are based on the concept of dependency relations between words, and are heavily inspired by the work of the French linguist Tesnière (1959). In the case of syntax,[1]

[1]Dependency grammars are not only used for syntax. For instance, they can also be used to encode semantic relations.

those dependency relations are binary relations that capture the grammatical relations that exists between those words. For example, in the sentence

*The blonde girl is riding a red bike.*

a syntactic dependency grammar will encode the fact that the noun *girl* is the subject of the verb *riding*, or the fact that the adjective *blonde* is modifying the word *girl*. Thus, in dependency grammars, the structure of a sentence is not captured by the constituents that appear in the sentence, but is captured by the binary syntactic relations that exists between the words. Those binary relations are not symmetric (think about the subject relation), and for each relation, there is a head word and a dependent word. In the case of the relation between *riding* and *girl*, the head word is the verb *riding* while the dependent word is the noun *girl*. We note relations using the following notation

```
nominal_subject(riding, girl)
adjective_modifier(girl, blonde)
determiner(girl, the)
```

where the first word is the head word and the second word is the dependent word. Words of a sentence and the corresponding syntactic relations form a directed acyclic graph (DAG), where words are the vertices and syntactic relations are the edges. Many representations often enforces that this DAG must be a tree, by removing some syntactic relations. We refer the reader to De Marneffe and Manning (2008) for an exposition of different representations of syntactic dependencies.

## 1.3 Semantics

Semantics is the subfield of linguistics focussing on the study of meaning. It is a very broad area of research in both linguistics and computational linguistics, and providing an extensive presentation of that field is out of the scope of this thesis. We thus restrict our presentation to notions that will be usefull in the following. We start by introducing lexical semantics, that is, the study of the meaning of words, before briefly discussing semantic compositionality.

### 1.3.1 Lexical semantics

Natural languages are highly ambiguous: a word can belong to different grammatical classes, a sentence can be parsed in different ways and of course, a given word can have different meanings. For example, consider the word *bridge* in the two following sentences:

1. *He likes to play* bridge *during his holidays.*
2. *The Pont Neuf is the oldest* bridge *in Paris.*

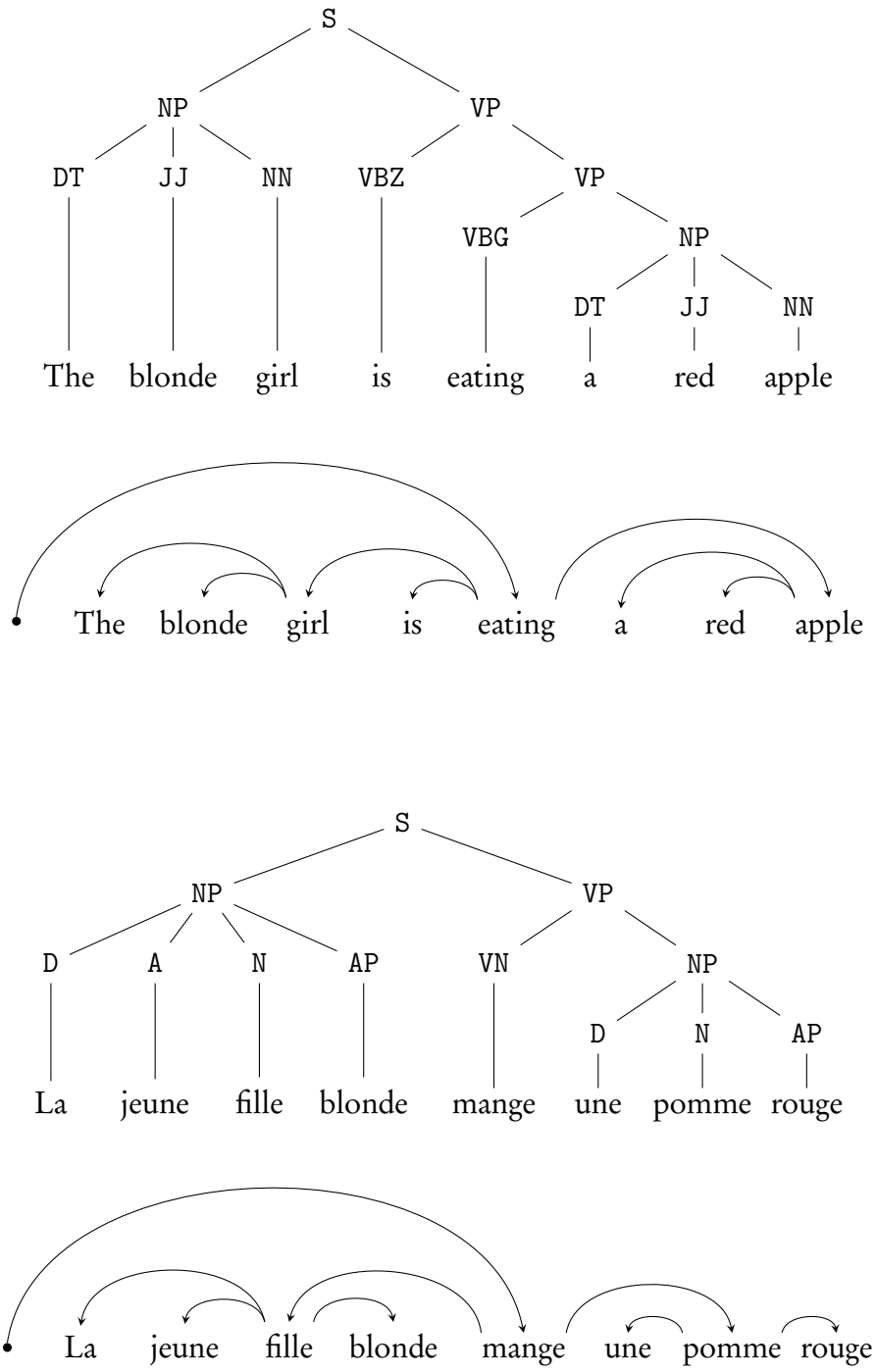



Figure 1.1: Example of a PCFG parse tree (top) and a dependency tree (bottom) for the English sentence: *The blonde girl is eating a red apple* and the French sentence: *La jeune fille blonde mange une pomme rouge*.

Dependending on the context, a *bridge* can design a card game or a structure for carrying traffic over water. To each different sense corresponds a different lexeme, and we note them by using superscript: *bridge*[1] and *bridge*[2]. Thus, a lexeme is a pair of a meaning and a set of words which are inflected variants of the same lemma. The relation between those two lexemes, *bridge*[1] and *bridge*[2], is called homonymy: they share the same orthographic form, *bridge*, but refer to unrelated things. Now, consider the word *university* in the two following sentences:

*a. Everyday, he goes to* university *by car.*

*b. He is a* university *professor in computer science.*

In the first sentence, the word *university* refers to the place, while in the second, it refers to the institution. Thus, the word *university* has two different, but related meaning, corresponding to two lexemes that we note *university*[a] and *university*[b]. The relation between those two lexemes is called polysemy. Given a surface form and a context, finding the corresponding lexeme is called word sense disambiguation. It is a very hard task and is still considered an open problem.

*Car*, *vehicle*, *engine* and *automobile* are four related words. More precisely, those words belong to lexemes that have a related meaning. The senses associated to those words are not all related in the same way: for example the relation between *car* and *vehicle* is not the same as the relation between *car* and *engine*. We now introduce the different semantic relations that exist between word senses and lexemes. First, *car* and *automobile* share the same sense: in that case, we say that the corresponding lexemes are synonyms, and the relation is called synonymy. Second, a *car* is a kind of *vehicle*. In that case, *car* is a hyponym of *vehicle*, while *vehicle* is a hypernym of *car*. Two senses sharing the same hypernym are called co-hyponyms, for example the senses associated to *truck* and *plane*. Third, since an *engine* is a part of a *car*, we say that *engine* is a meronym of *car* and that *car* is a holonym of *engine*. Finally, two senses that have opposite meanings, such as *hot* and *cold* are called antonyms.

### 1.3.2 Semantic compositionality

In the previous section, we discussed the meaning of individual words (possibly in context). A crucial characteristic of natural languages is their ability to express a potentially infinite number of different ideas while only using a finite number of words and their associated meanings. This is possible thanks to the principle of semantic compositionality. This principle, often attributed to German logician Gottlob Frege, states that the meaning of a complex expression, such as a sentence, is a function of the meaning of its lexical parts and the way they are arranged together. In two seminal articles (Montague, 1970, 1973), Richard Montague applied the principle of compositionality to natural languages by introducing a theory of semantics in which meaning is represented by logical forms based on predicate logic and lambda calculus. He dicussed the relation between this kind of semantic representation and the syntax, and demonstrated how this could be applied to English. A large body of work has built upon the work of Montague, trying to parse natural language sentences into representations based on formal logic.

## 1.4 Distributional semantics

Distributional semantics methods aim at capturing words meanings (or the meaning of other linguistic units, such as morphemes, phrases or documents) based on the contexts in which they appear and their usage patterns. It is indeed believed that words appearing in similar contexts share similar meanings (Harris, 1954), an idea known as the distributional hypothesis which can be summarized as follow (Firth, 1957):

> "You shall know a word by the company it keeps".

Thus distributional semantics methods extract distributional information about words from vast quantities of unlabeled data, and builds a representation based on this information. We now present the main approaches proposed in distributional semantics.

### 1.4.1 Vector space models

In vector space models, words are represented as vectors in a high-dimensional space, with the underlying assumption that the similarity between words can be approximated by the similarity between the corresponding vectors. Combined with the distributional hypothesis, it means that these vectors can be obtained from large quantity of unlabeled text, by extracting distributional information about word usage and context. A corpus is represented as a word-by-context matrix, where each line of the matrix represents a unique word and each column represents a context in which words appear. Three choices have to be made when designing a vector space representation for words: first, what is the context of a word, second, given distributional information, how to build the word-by-context matrix from that information, third, what similarity measure to use between vectors.

One of the first applications of vector space models was in information retrieval, and in particular to document retrieval (Salton et al., 1975). In order to obtain document representations, Deerwester et al. (1990) introduced latent semantic analysis (LSA), where contexts are defined as documents in which the word appears: the corpus is thus represented as a word-by-document co-occurence matrix and the coefficient $(i, j)$ of that matrix is the number of times the word $i$ appears in document $j$. The main contribution of LSA was to propose to apply a dimension reduction technique, namely principal component analysis, on that matrix before computing document similarity. Landauer and Dumais (1997) later proposed to use the same kind of representation to compute word similarity (and not document similarity), and apply it to multiple-choice synonym test. In that case, contexts were not whole documents but paragraphs.

In order to compute word similarity, Schutze (1992) and Lund and Burgess (1996) proposed to consider word-word co-occurences, instead of word-document co-occurence. In that case, the contexts that are considered are the words that appear in the neighbourhood of the target word, for example in the same sentence or in a fixed-size window arount the target word. Thus, the corpus is represented as a word-by-word co-occurence matrix and the coefficient $(i, j)$ of

| | | |
|---|---|---|
| considerable body of songs and | piano | music , as well as symphonic suites |
| hed a reputation in Vienna as a | piano | virtuoso , but he apparently with |
| nd guitar , Darcy on guitar and | piano | , and Ollie Murphy on drums . |
| nist , and Birdy learned to play | piano | at the age of seven , and began writ |
| I e Distacco II " for voice and | piano | on a text by Ranieri Gnoli , " Verrà |
| musicians : Niels Lan Doky ( | piano | ) , Niels-Henning Ørsted |
| sson practice , giving lessons in | piano | playing , arranging , general |
| ning early , beginning with the | piano | at the age of four and continuing |
| n on drums , Chris Gardner on | piano | , Danny Brittain on lead vocals |
| eek , Jose met with the nun for | piano | lessons and having no piano at |
| his short pieces , sonatina , for | piano | , enjoyed great popularity both |
| st compositions were songs and | piano | pieces inspiBrickRed by these European |
| piece that originated from Lisa 's | piano | and vocal compositions , but was |
| to be released as a single , and a | piano | break was edited out , and the edited |
| I 'd learned a few chords on the | piano | , maybe two , so I 'd already tried |
| ist , and won numerous amateur | piano | competitions while working as |
| of drumming , with additional | piano | , percussions , synthesizers |
| was born to them in 1833 ( the | piano | pedagogue Charles-Wilfrid de |
| instruments including trumpet , | piano | , flute , harp , bassoon , percussion |
| bass ; Spiewak and Capps played | piano | and organ ; and Sara Jean Kelley |
| to play instruments such as the | piano | and bass at an early age . |
| ek gig , that included Holler on | piano | , Jimmy Clanton on lead guitar |
| accompany singers , while the | piano | was all the rage throughout Europe |
| several major international | piano | competitions and regularly teaches |
| eighty arrangements for organ , | piano | and chamber ensembles on works |
| works ; five concertos ( 2 for | piano | ; 1 for violin ; 1 for cello ; 1 for |
| sonatas for violin and cello , a | piano | quintet and a string quartet for |
| sisted of : Walt Gates on grand | piano | , Artie Singer on upright bass |
| his first music achievement , a | piano | duet , at age of nine . |
| in addition to Holopainen on | piano | , Marco Hietala performs vocals |

Table 1.1: Examples of usage pattern of the word *piano*.

that matrix is the number of times that the word $i$ appears in the same context as the word $j$. Once again, dimension reduction is performed on that matrix before computing word similarity. This method was successfully applied to word sense disambiguation (Schutze, 1992), word categorization and lexical priming (Lund and Burgess, 1996).

One of the main limitation of the previous models is the fact that they do not take syntax into account. Several models have been proposed to address this shortcoming by using the syntactic dependencies to define the context of the target word (Lin, 1998; Curran and Moens, 2002; Turney, 2006; Padó and Lapata, 2007; Van de Cruys, 2010). For building those representations, syntactic dependencies are extracted from the corpus, and two words are considered in the same context if there exists a dependency between them, or a path in the dependency graph. Some models, such as the ones considered by Padó and Lapata (2007) simply discard the type of the dependencies, while other models represent context not by using words but pairs of word and syntactic relation (Lin, 1998; Curran and Moens, 2002). This allows, for example, to distinguish between subjects and objects for verbs and thus to capture finer information. Baroni and Lenci (2010) proposed to store the distributional information in a third-order tensor and then to apply different kind of matricizations to recover various vector space models.

Another active area of research in semantic space models is the problem of how to combine the representations of words to form good representations of larger linguistic units such as phrases or sentences. For example, given the two vectors representing the noun *agency* and the adjective *federal*, how to combine them to get a good representation of the noun phrase *federal agency*. Indeed, natural languages are extremely compositional, meaning that in many cases it is possible to get the meaning of a linguistic unit based on the meaning of its parts.[2] Mitchell and Lapata (2008) proposed simple ways to compose vector representations, such as addition, componentwise multiplication, tensor product or dilation, and evaluated them on a dataset of adjective-noun, noun-noun and verb-noun pairs, introduced by Mitchell and Lapata (2010).

Another approach to semantic composition is to learn the function that combines vector representations using a supervised method. First, a semantic space is built where both the phrases and their constituents appear. For example, *federal*, *agency* and *federal_agency* all have a corresponding vector. Then, the goal is to learn a function that maps representations of the constituents onto the representation of the phrase. Guevara (2010, 2011) proposed to use partial least square regression, while Baroni and Zamparelli (2010) proposed to learn a matrix $\mathbf{A}$ for each adjective such that the vector $\mathbf{p}$ corresponding to the adjective-noun pair can be obtained from the vector $\mathbf{b}$ corresponding to the noun by

$$\mathbf{p} = \mathbf{A}\mathbf{b}.$$

This model was later generalized by Socher et al. (2012): a matrix in $\mathbb{R}^{n \times n}$ and a vector in $\mathbb{R}^n$ are associated to each word of the vocabulary. Then given two matrix-vector pairs ($\mathbf{A}$, $\mathbf{a}$) and

[2] This is not always true. Consider the following multi-word expressions: White House, by the way, to look something up...

($\mathbf{B}$, $\mathbf{b}$) representing words, a new matrix-vector ($\mathbf{P}$, $\mathbf{p}$) pair representing the composition of the two words is obtained by

$$\mathbf{P} = \mathbf{W}_M \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$$

and

$$\mathbf{p} = g\left(\mathbf{W} \begin{bmatrix} \mathbf{Ab} \\ \mathbf{Ba} \end{bmatrix}\right),$$

where $\mathbf{W}_M, \mathbf{W} \in \mathbb{R}^{n \times 2n}$ and $g$ is an elementwise non linear function. The idea is that the word matrices $\mathbf{A}$ and $\mathbf{B}$ will capture the compositional effects of words, while the vectors $\mathbf{a}$ and $\mathbf{b}$ will capture their meaning. The matrices $\mathbf{W}_M$ and $\mathbf{W}$ capture general composition functions that apply to every words. Moreover, thanks to them, $\mathbf{P}$ and $\mathbf{p}$ live in the same space as $\mathbf{A}$ and $\mathbf{a}$, allowing to recursively apply this composition. The authors propose to do so for every node in a parse tree.

### 1.4.2 Latent dirichlet allocation and topic models

Probabilistic latent semantic analysis (pLSA) is a probabilistic model of documents proposed by Hofmann (1999) and inspired by latent semantic analysis. The fact that pLSA does not define a proper generative model for new documents once fitted on a training set, led Blei et al. (2003) to propose latent Dirichlet allocation (LDA), a generative model of documents. The idea behind latent Dirichlet allocation (and other topic models such as pLSA), is that each document can be viewed as a mixture of $k$ topics, where each topic is a distribution over the words of the vocabulary. Topics are shared among all the documents of a corpus, and can be seen as the underlying structure of the documents. We now present how a document $\mathbf{w}$ is generated, according to the LDA model:

1. Draw the length $N$ of the document from a Poisson distribution:

$$N \mid \xi \sim \text{Poisson}(\xi),$$

2. Draw the mixture of topics $\theta$ from a Dirichlet distribution:

$$\theta \mid \alpha \sim \text{Dirichlet}(\alpha),$$

3. For each word $W_i$, $i \in \{1, ..., N\}$ of the document:

   (a) Draw the topic indicator $Z_i$ from a multinomial distribution:

$$Z_i \mid \theta \sim \text{Multinomial}(\theta),$$

   (b) Draw the word $W_i$ from the topic $\gamma_{Z_i}$:

$$W_i \mid Z_i, \gamma_{1..k} \sim \text{Multinomial}(\gamma_{Z_i}).$$

The random variable $\theta$ represents the proportion of each topic in the document, while the variable $Z_i$ indicates from which topic the word $W_i$ is coming. Both $\theta$ and $Z$ are latent variables, meaning that, given a document, they are not observed and have to be infered. The topic parameters $\gamma_{1..k}$ are shared among the different documents of the corpus and have to be learnt. Depending on the value of the parameters $\alpha$ of the Dirichlet prior over topic distributions, the mixture of topics $\theta$ will be more or less concentrated over a few topics, meaning that for a given document, words come from a more or less larger number of topics. Recently, Hoffman et al. (2013) proposed a stochastic variational inference algorithm, making it possible to train latent Dirichlet allocation models on millions of documents.

Latent Dirichlet allocation can be seen as a dimension reduction technique: each document of a corpus can be represented by its mixture of topic $\theta$ instead of the word frequencies. Since the number of topics is usually much smaller than the number of words, this greatly reduces the dimension of the representation. Moreover, probabilistic latent semantic analysis, latent Dirichlet allocation and non-negative matrix factorization are closely related (Buntine, 2002; Gaussier and Goutte, 2005). Indeed, when the latent variables $Z_i$ have been marginalized out, the generation of a document resumes to:

$$\begin{aligned}\theta &\sim \text{Dirichlet}(\alpha),\\ W &\sim \text{Multinomial}(\Gamma\theta, N),\end{aligned}$$

where $\Gamma = [\gamma_1, ..., \gamma_k]$ is the matrix representing topics and $N$ is the number of words of the document. Thus, latent Dirichlet allocation can be viewed as the analogue of principal component analysis for discrete and positive variables, and thus as a probabilistic model for non-negative matrix factorization. Blei et al. (2003) used a LDA model with 50 topics for document classification, showing that when the number of labeled documents is small, using the topic proportions $\theta$ performs better than using the word frequencies.

Many variants and extensions of latent Dirichlet allocation have been proposed in the last decade, in order to address its limitations. For example, in LDA, topic proportions are independent of one another which is clearly not the case in practice: in news articles, the *politics* topic is much more likely to co-occur with the *economics* topic than with the *science* topic. Thus, Blei and Lafferty (2006) proposed the correlated topic model, where the Dirichlet distribution for the mixture of topics is replaced by the logistic normal distribution that capture correlations between the mixture components. A second possible extension, proposed by Blei et al. (2004), is to consider that the topics forms a hierarchy. The topics are thus arranged in a tree, and for each document a path from the root to a leaf of the tree is chosen, and only topics from that path can appear in the given document. Finally, in most topic models, words inside a document are considered exchangeable, leading to the bag of words representation. Griffiths et al. (2005) proposed to address this limitation by using a hidden Markov model to capture transitions between words, and in particular between function words and content words, the latter being generated by a topic model.

Topic models have been applied to a lot of different tasks in natural language processing. For example, an extension of LDA was proposed by Toutanova and Johnson (2007) for semi-supervised POS tagging: in that case, LDA is used to model the contexts of words. Another extension of LDA was proposed for word sense disambiguation by Boyd-Graber et al. (2007). In that case, topics are not used directly to generate words, but instead defines random walks in WordNet, that in turn, generate words. Misra et al. (2009) proposed to use LDA to perform the task of text segmentation, which consists in dividing unlabeled textual data into meaningful segments. Eisenstein et al. (2010) proposed a latent topic model that aims at capturing geographic lexical variations, while Séaghdha (2010) proposed to use LDA to capture selectional preferences. More recently, Titov and Klementiev (2012) proposed a topic model for semantic role induction, while O'Connor et al. (2013) designed a probabilistic topic model to detect and extract events from political contexts.

### 1.4.3 Brown clustering and other clusterings

Our tour of distributional semantic methods ends with clustering methods, and more particularly, the one introduced by Brown et al. (1992) and known as Brown clustering. This method aims at finding a function $\mathscr{C}$ that maps words to clusters, and which maximizes the likelihood of the data, assuming the following model for a sentence $\mathbf{w}$:

$$p(\mathbf{w}) = \prod_k p(w_k \mid \mathscr{C}(w_k))\, p(\mathscr{C}(w_k) \mid \mathscr{C}(w_{k-1})).$$

This corresponds to a generative model of sentences, where words are generated sequentially, according to the following process. First, the clusters are generated according to a Markov chain. Then for each $k$, knowing the cluster $\mathscr{C}(w_k)$, the corresponding word $w_k$ is generated independently from the other words.

Maximizing this likelihood is equivalent to maximize the mutual information between adjacent clusters, and only depends on the counts $c(w_k, w_{k+1})$ of bigrams. Brown et al. (1992) proposed a bottom-up greedy agglomerative algorithm to find the clustering $\mathscr{C}$. At the beginning of the algorithm, each word form a different cluster. Then, at each iteration of the algorithm, two clusters are merged, such that it least reduces the likelihood. Choosing these clusters can done in $O(n^2)$, where $n$ is the current number of clusters. Since this operation has to be done $V - C$ times, where $V$ is the size of the vocabulary and $C$ is the final number of clusters wanted, the overall complexity is $O(V^3)$, which is prohibitive for large vocabularies. Thus, Brown et al. (1992) proposed to only consider the first $C$ clusters for each merging step, where the clusters are sorted by frequency. The complexity then becomes $O(VC^2)$, which is tractable in practice.

Another optimization technique is the one proposed by Kneser and Ney (1993) and called the exchange clustering algorithm. At each iteration, the current clustering is improved by trying to switch each word to a new cluster such that it most increases the likelihood of the data. In order to speed up this algorithm, Uszkoreit and Brants (2008) considered a slightly

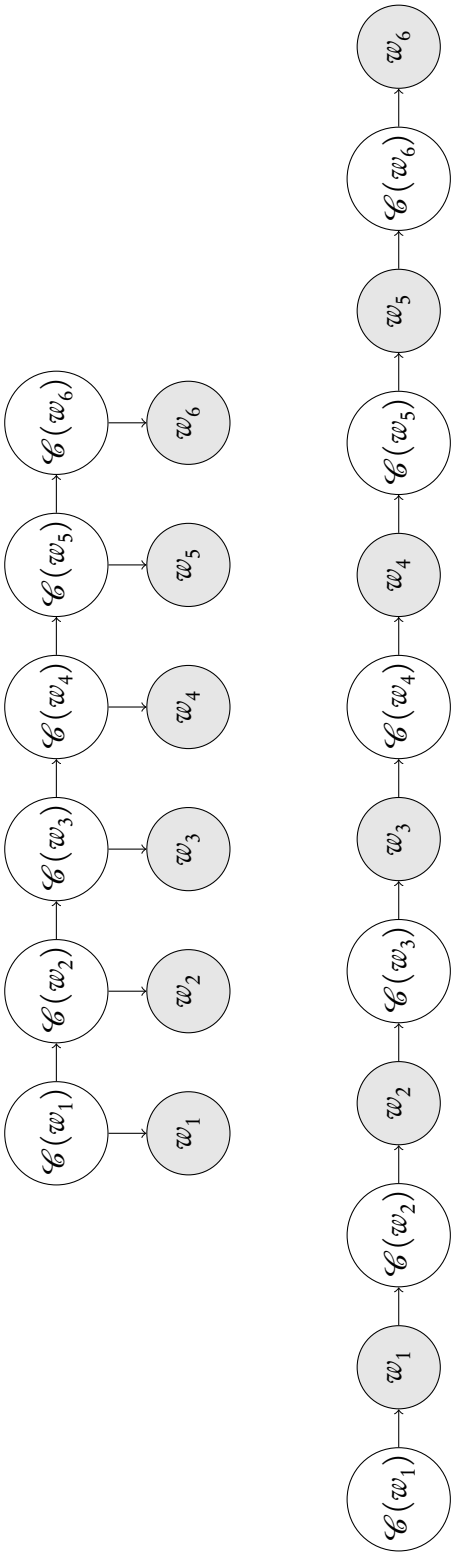



Figure 1.2: Graphical models corresponding to the clustering model proposed by Brown et al. (1992) (top) and clustering model proposed by Goodman (2001) (bottom).

different model, which was previously proposed by Goodman (2001), and where the class-to-class transitions are replaced by word-to-class transitions, giving the following probability of a word sequence:

$$\prod_k p(w_k \mid \mathscr{C}(w_k))\; p(\mathscr{C}(w_k) \mid w_{k-1}).$$

Thanks to that modification, Uszkoreit and Brants (2008) designed an efficient variant of the exchange algorithm, allowing them to train models on very large datasets. This model was later extended to the multilingual setting by Täckström et al. (2012).

# CHAPTER 2



# HIDDEN MARKOV TREE MODELS FOR SEMANTIC CLASS INDUCTION

IN THIS CHAPTER, we describe a new unsupervised method for semantic class induction. This is achieved by introducing a generative model of sentences with latent variables, based on dependency trees and which takes into account homonymy. This model can be seen as a generalization of Brown clustering taking into account the syntax and homonymy. Then, we describe an efficient algorithm to perform inference and learning in this model, in order for our method to be scalable to large datasets containing tens of millions of sentences. This algorithm, based on approximate message passing and online EM, allowed us to train models with hundreds of latent states on a dataset with hundreds of millions of tokens in less than two days on a single core. Finally, we apply the proposed method on two large datasets ($10^8$ tokens, $10^5$ words types), and qualitatively discuss the semantic classes we obtain. Quantitative evaluations are performed in chapter 3 and 4.

The material of this chapter is based on the following work:

E. Grave, G. Obozinski and F. Bach. Hidden Markov tree models for semantic class induction. In *Seventeenth Conference on Computational Natural Language Learning (CoNLL)*. 2013.

## 2.1 Model

In this section, we introduce our probabilistic generative model of sentences. We start by setting up some notations. A sentence is represented by a $K$-tuple $\mathbf{w} = (w_1, ..., w_K)$ where each $w_k \in \{1, ..., V\}$ is an integer representing a word and $V$ is the size of the vocabulary. Our goal will be to infer a $K$-tuple $\mathbf{c} = (c_1, ..., c_K)$ of semantic classes, where each $c_k \in \{1, ..., C\}$ is an integer representing a semantic class, corresponding to the word $w_k$.

The generation of a sentence can be decomposed in two steps: first, we generate the semantic classes according to a Markov process, and then, given each class $c_k$, we generate the corresponding word $w_k$ independently of other words. The Markov process used to generate the semantic classes will take into account selectional preference. Since we want to model homonymy, each word can be generated by multiple classes.

We now describe the Markov process we propose to generate the semantic classes. We assume that we are given a directed tree defined by the function $\pi : \{1,...,K\} \mapsto \{0,...,K\}$, where $\pi(k)$ represents the unique parent of the node $k$ and $0$ is the root of the tree. Each node, except the root, corresponds to a word of the sentence. First, we generate the semantic class corresponding to the root of the tree and then generate recursively the class for the other nodes. The classes are conditionally independent given the classes of their parents. Using the language of probabilistic graphical models, this means that the distribution of the semantic classes factorizes in the tree defined by $\pi$ (See Fig. 2.1 for an example). We obtain the following distribution on pairs $(\mathbf{w}, \mathbf{c})$ of words and semantic classes:

$$p(\mathbf{w}, \mathbf{c}) = \prod_{k=1}^{K} p(c_k \mid c_{\pi(k)})\, p(w_k \mid c_k),$$

with $c_0$ being equal to a special symbol denoting the root of the tree.

In order to fully define our model, we now need to specify the observation probability distribution $p(w_k \mid c_k)$ of a word given the corresponding class and the transition probability distribution $p(c_k \mid c_{\pi(k)})$ of a class given the class of the parent. Both these distributions will be categorical (and thus multinomial with one trial). The corresponding parameters will be represented by the stochastic matrices $\mathbf{O}$ and $\mathbf{T}$ (i.e. matrices with non-negative elements and unit-sum columns):

$$p(W_k = i \mid C_k = j) = O_{ij},$$
$$p(C_k = i \mid C_{\pi(k)} = j) = T_{ij}.$$

Finally, we introduce the trees that we consider to define the distribution on semantic classes. (We recall that the trees are assumed given, and not a part of the model.)

### 2.1.1 Markov chain model

The simplest structure we consider on the semantic classes is a Markov chain. In this special case, our model reduces to a hidden Markov model. Each semantic class only depends on the class of the previous word in the sentence, thus failing to capture selectional preference of semantic class. But because of its simplicity, it may be more robust, and does not rely on external tools. It can be seen as a generalization of the Brown clustering algorithm (Brown et al., 1992) taking into account homonymy.

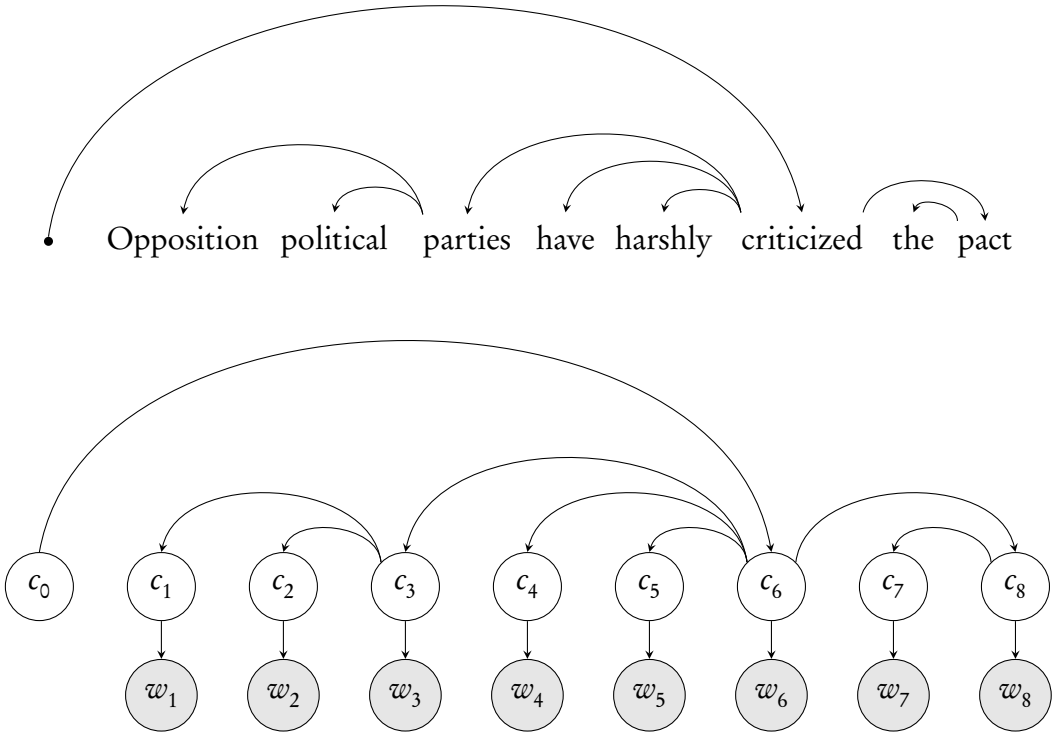



Figure 2.1: Example of a dependency tree and its corresponding graphical model.

### 2.1.2 Dependency tree model

The second kind of structure we consider to model interactions between semantic classes is a syntactic dependency tree corresponding to the sentence. A dependency tree is a labeled tree in which nodes correspond to the words of a sentence, and edges represent the grammatical relations between those words, such as *nominal subject*, *direct object* or *determiner*. We use the Stanford typed dependencies basic representations, which always form a tree (De Marneffe and Manning, 2008).

We believe that a dependency tree is a better structure than a Markov chain to learn semantic classes, with no additional cost for inference and learning compared to a chain. First, syntactic dependencies can capture long distance interactions between words. See Fig. 2.1 and the dependency between `parties` and `criticized` for an example. Second, the syntax is important to model selectional preference. Third, we believe that syntactic trees could help much for languages which do not have a strict word order, such as Czech, Finnish, or Russian. One drawback of this model is that all the children of a particular node share the same transition probability distribution. While this is not a big issue for nouns, it is a bigger concern for verbs: subject and object should not share the same transition probability distribution.

A potential solution would be to introduce a different transition probability distribution for each type of dependency. This possibility will be explored in future work.

### 2.1.3 Brown clustering on dependency trees

As for Brown clustering, we can assume that words are generated by a single class. In that case, our model reduces to finding a deterministic clustering function $\mathscr{C}$ which maximizes the following likelihood:

$$\prod_k p(w_k \mid \mathscr{C}(w_k)) p(\mathscr{C}(w_k) \mid \mathscr{C}(w_{\pi(k)})).$$

In that case, we can use the algorithm proposed by Brown et al. (1992) to greedily maximize the likelihood of the data. This model can be seen as a generalization of Brown clustering taking into account the syntactic relations between words.

## 2.2 Inference and learning

In this section, we present the approach used to perform learning and inference in our model. Our goal here is to have efficient algorithms, in order to apply our model to large datasets ($10^8$ tokens, $10^5$ words types). The parameters $\mathbf{T}$ and $\mathbf{O}$ of the model will be estimated with the maximum likelihood estimator:

$$\hat{\mathbf{T}}, \hat{\mathbf{O}} = \underset{\mathbf{T},\mathbf{O}}{\operatorname{argmax}} \prod_{n=1}^{N} p(\mathbf{w}^{(n)} \mid \mathbf{T}, \mathbf{O}),$$

where $(\mathbf{w}^{(n)})_{n\in\{1,\dots,N\}}$ represents our training set of $N$ sentences.

First, we present an online variant of the well-known expectation-maximization (EM) algorithm, proposed by Cappé and Moulines (2009), allowing our method to be scalable in term of numbers of examples. Then, we present an approximate message passing algorithm which has a linear complexity in the number of classes, instead of the quadratic complexity of the exact inference algorithm. Finally, we describe a state-splitting strategy to speed up the learning.

### 2.2.1 Online EM

In the batch EM algorithm, the E-step consists in computing the expected sufficient statistics $\tau$ and $\omega$ of the model, sometimes referred as pseudocounts, corresponding respectively to $\mathbf{T}$ and $\mathbf{O}$:

$$\tau_{ij} = \sum_{n=1}^{N} \sum_{k=1}^{K_n} \mathbb{E}\left[\mathbf{1}\{C_k^{(n)} = i,\ C_{\pi(k)}^{(n)} = j\} \mid W^{(n)} = \mathbf{w}^{(n)}\right],$$

$$\omega_{ij} = \sum_{n=1}^{N} \sum_{k=1}^{K_n} \mathbb{E}\left[\mathbf{1}\{W_k^{(n)} = i,\ C_k^{(n)} = j\} \mid W^{(n)} = \mathbf{w}^{(n)}\right].$$

On large datasets, $N$ which is the number of sentences can be very large, and so, EM is inefficient because it requires that inference is performed on the entire dataset at each iteration. We therefore consider the online variant proposed by Cappé and Moulines (2009): instead of recomputing the pseudocounts on the whole dataset at each iteration $t$, those pseudocounts are updated using only a small subset $\mathcal{B}_t$ of the data, to get

$$\tau_{ij}^{(t)} = (1-\alpha_t)\tau_{ij}^{(t-1)} + \alpha_t \sum_{n\in\mathcal{B}_t}\sum_{k=1}^{K_n} \mathbb{E}\left[\mathbf{1}\{C_k^{(n)}=i,\ C_{\pi(k)}^{(n)}=j\} \mid W^{(n)}=\mathbf{w}^{(n)}\right],$$

and

$$\omega_{ij}^{(t)} = (1-\alpha_t)\omega_{ij}^{(t-1)} + \alpha_t \sum_{n\in\mathcal{B}_t}\sum_{k=1}^{K_n} \mathbb{E}\left[\mathbf{1}\{W_k^{(n)}=i,\ C_k^{(n)}=j\} \mid W^{(n)}=\mathbf{w}^{(n)}\right],$$

where the scalars $\alpha_t$ are defined by $\alpha_t = 1/(a+t)^\gamma$ with $0.5 < \gamma \leq 1$. In the experiments, we used $a = 4$. We chose $\gamma$ in the set $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

### 2.2.2 Approximate inference

Inference is performed on trees using the sum-product message passing algorithm, a.k.a. belief propagation, which extends the classical $\alpha-\beta$ recursions used for chains, see e.g. Wainwright and Jordan (2008). We denote by $\mathcal{N}(k)$ the set containing the children and the father of node $k$. In the exact message-passing algorithm, the message $\mu_{k\to\pi(k)}$ from node $k$ to node $\pi(k)$ takes the form:

$$\mu_{k\to\pi(k)} = \mathbf{T}^\top \mathbf{u},$$

where $\mathbf{u}$ is the vector obtained by taking the elementwise product of all the messages received by node $k$ except the one from node $\pi(k)$, i.e.,

$$u_i = \prod_{k'\in\mathcal{N}(k)\setminus\{\pi(k)\}} \mu_{k'\to k}(i).$$

Similarly, the pseudocounts can be written as

$$\mathbb{E}\left[\mathbf{1}\{C_k^{(n)}=i, C_{\pi(k)}^{(n)}=j\} \mid W^{(n)}=\mathbf{w}^{(n)}\right] \propto u_i T_{ij} v_j,$$

where $\mathbf{v}$ is the vector obtained by taking the elementwise product of all the messages received by node $\pi(k)$, except the one from node $k$, i.e.,

$$v_j = \prod_{k'\in\mathcal{N}(\pi(k))\setminus\{k\}} \mu_{k'\to\pi(k)}(j).$$

Both these operations thus have quadratic complexity in the number of semantic classes. In order to reduce the complexity of those operations, we propose to start by projecting the vectors $\mathbf{u}$ and $\mathbf{v}$ on a set of sparse vectors, and then, perform the operations with the sparse approximate vectors. We consider two kinds of projections:

- *k-best projection*, where the approximate vector is obtained by keeping the $k$ largest coefficients,
- *$\varepsilon$-best projection*, where the approximate vector is obtained by keeping the smallest set of larger coefficients such that their sum is greater than $(1-\varepsilon)$ times the $\ell_1$-norm of the original vector.

This method is similar to the one proposed by Pal et al. (2006). Another approach, proposed to learn large scale conditional random fields, is to regularize the coefficients of the model using the $\ell_1$-norm and to take advantage of the induced sparsity (Lavergne et al., 2010). The advantage of the $k$-best projection is that we control the complexity of the operations, but not the error, while the advantage of the $\varepsilon$-best projection is that we control the error but not the complexity. As shown in Fig. 2.2, good choices for $\varepsilon$ and $k$ are respectively 0.01 and 16. We use these values in the experiments.

We also note on Figure. 2.3, that during the first iterations of EM, the sparse vectors obtained with the $\varepsilon$-best projection have a large number of non-zero elements. Thus, this projection is not adequate to directly learn large latent class models. This issue is addressed in the next section, where we present a state splitting strategy in order to learn models with a large number of latent classes.

### 2.2.3 State splitting

A common strategy to speed up the learning of large latent state space models, such as ours, is to start with a small number of latent states, and split them during learning (Petrov, 2009). As far as we know, there are still no good heuristics to choose which states to split, or how to initialize the parameters corresponding to the new states. We thus apply the simple, yet effective method, consisting in splitting all states into two and in breaking the symmetry by adding a bit of randomness to the emission probabilities of the new states. As noted by Petrov (2009), state splitting could also improve the quality of learnt models.

### 2.2.4 Initialization

Because the negative log-likelihood function is not convex, initialization can greatly change the quality of the final model. Initialization for online EM is done by setting the initial pseudocounts, and then performing an M-step. We have considered the following strategies to initialize our model:

- *random initialization:* the initial pseudocounts $\tau_{ij}$ and $\omega_{ij}$ are sampled from a uniform distribution on $[0,1]$,
- *Brown initialization:* the model is initialized using the (normalized) pseudocounts obtained by the Brown clustering algorithm. Because a parameter equal to zero remains equal to zero when using the EM algorithm, we replace null pseudocounts by a small smoothing value, e.g., for observation $i$, we use $10^{-5} \times \max_j \omega_{ij}$,

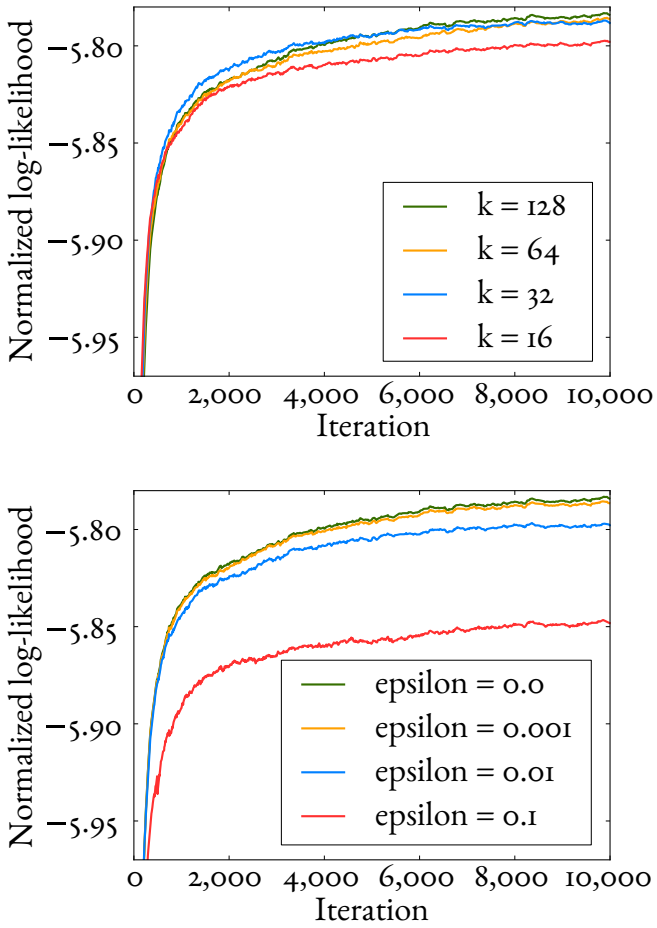



Figure 2.2: Comparison of the two projection methods for approximating vectors, for a model with 128 latent classes. The two plots are the log-likelihood on a held-out set as a function of the iterates of online EM. Green curves ($k = 128$ and $\varepsilon = 0$) correspond to learning without approximation.

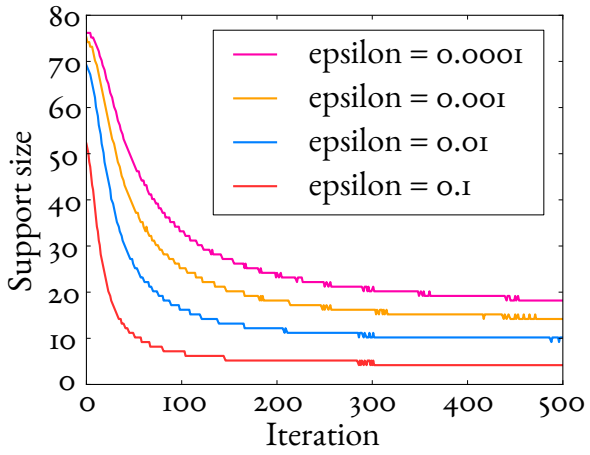



Figure 2.3: Size of the support of the approximate vector for the $\varepsilon$-best projection.

## 2.3 Experiments

In this section, we present the datasets used for the experiments and we qualitatively evaluate the proposed model.

### 2.3.1 Datasets

We considered five datasets: the first one, which we refer to as the *music dataset*, corresponds to all the Wikipedia articles refering to a musical artist. They were extracted using the Freebase database.[1] This dataset comprises 2.22 millions sentences and 56 millions tokens. We choose this dataset because it corresponds to a restricted domain. The second dataset are the articles of the NYT corpus (Sandhaus, 2008) corresponding to the period 1987-1997 and labeled as *news*. This dataset comprises 14.7 millions sentences and 310 millions tokens. The third dataset are biomedical abstracts from PubMed, obtained by performing a search with the keyword *cancer*. This dataset comprises 8.1 millions sentences and 190 millions tokens. The fourth dataset is all the articles from the French Wikipedia (2013 dump). It comprises 19.6 millions sentences and 425 millions tokens. The fifth dataset is all the articles from the Italian Wikipedia, made available by Baroni et al. (2009). It comprises 7.7 millions sentences and 170 millions tokens.

We parsed English datasets using the Stanford parser, and converted parse trees to dependency trees (De Marneffe et al., 2006). We decided to discard sentences longer than 50 tokens, for parsing time reasons, and then lemmatized tokens using Wordnet. We parsed the French dataset using the Malt parser a described by Candito et al. (2010). Each word of our vocabulary is then a pair of lemma and its associated part-of-speech. This means that the noun *attack* and the verb *attack* are two different words. Finally, we introduced a special token, -*-, for infrequent (lemma, part-of-speech) pairs, in order to perform smoothing. Forexample, for the music dataset, we kept the 25 000 most frequent words, while for the NYT corpus, we kept the 100 000 most frequent words.

### 2.3.2 Semantic classes

Before moving on to the quantitative evaluation of our model, we discuss qualitatively the induced semantic classes. Examples of semantic classes are presented in Tables 2.1, 2.2 and 2.3. Tree models with random initialization were used to obtain those semantic classes. First we observe that most classes can be easily given natural semantic interpretation. For example, class 3 of Table 2.1 contains musical genres, while class 4 contains musical instruments.

Table 2.2 presents groups of classes that contain a given homonymous word; it seems that the different classes capture rather well the different senses of each word. For example, the word *head* belongs to the class 1, which contains words referring to leaders and to the class 2, which contains body parts.

[1] www.freebase.com

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| bach | tour | rock | guitar | school | win | reach |
| mozart | show | pop | bass | university | receive | peak |
| liszt | concert | jazz | vocal | college | sell | hit |
| beethoven | performance | classical | drum | hall | gain | chart |
| wagner | appearance | folk | keyboard | conservatory | earn | go |
| chopin | gig | punk | piano | academy | award | debut |
| brahms | date | metal | saxophone | center | achieve | make |

Table 2.1: Selected semantic classes corresponding to the music dataset.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| president | **head** | metal | **oil** | stock | score | kill |
| member | hand | gas | salt | price | hit | **shoot** |
| director | face | **oil** | sauce | index | lead | arrest |
| chairman | hands | paint | butter | market | give | die |
| executive | foot | steel | mixture | future | take | injure |
| officer | knee | wood | potato | **oil** | make | found |
| **head** | shoulder | paper | heat | exchange | go | wound |
| friend | eyes | plastic | juice | gold | run | beat |
| leader | hair | material | sugar | commodity | get | fire |
| minister | back | fuel | tomato | trading | **shoot** | release |

Table 2.2: Semantic classes containing homonymous words. Different classes capture different senses of each word.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1,000 | work | orchestra | saudi | add | country | percent |
| 2,000 | job | music | eastern | place | city | or |
| 300 | school | symphony | southern | serve | area | those |
| 10,000 | training | chamber | northern | cook | town | 81/2 |
| 3,000 | program | piano | Puerto | stir | state | 8 |
| 20,000 | class | mozart | central | remove | nation | 87/8 |
| 5,000 | education | quartet | northern | cut | community | 225 |

Table 2.3: Randomly selected semantic classes corresponding to the news dataset.

### 2.3.3 Transitions between semantic classes

We have plotted the transition matrices of a tree model and a chain model, both with 128 latent classes in Figure 2.4. We notice that the transition matrices are very sparse. An interesting difference to notice between the two matrices is the fact that for the tree model, the weight of the diagonal is more important than for the chain model. This means that for tree models, semantic classes have a higher probability of self transition than for chain models.

We now give some examples of semantic classes, along with the classes that are the most probably generated after them, in Table 2.4. We observe that our model is able to capture selectional preferences, for example the fact that *team, champion* or *player* can *win, play* or *lose* things such as *game, league* or *series*. We also observe that using unlabeled dependency trees leads to errors, such as putting *team* in the class containing the words *game, league, series* and *championship*.

| win play lose lead beat defeat be make |
|---|
| in for to on at into of from |
| team champion player winner record |
| to with by for he on after but |
| game league series team championship |
| · win play lose lead beat defeat be make |

| announce call agree refuse ask begin vote |
|---|
| make have discuss win seek negotiate |
| plan agreement law bill program proposal |
| today yesterday also recently tonight |
| have already never do eventually |
| court government commission jury judge |

| sell buy help build use provide make |
|---|
| to from them him how into with it |
| system program business product |
| · sell buy help build use provide make |
| food money equipment supply good |
| student people worker employee job |

| rise fell be close offer drop gain |
|---|
| to from at by with for while as |
| up down compare yield sharply |
| share bond stock dollar note issue |
| to 1/2 at 1/4 3/4 1 12 1 1 18 |
| stock price index market future oil |

| judge attorney justice general secretary |
|---|
| former prime republican democratic |
| federal state united supreme district |
| a an such with to for his like |
| · judge attorney justice general secretary |
| the both his along our each itself |

| night season morning day afternoon |
|---|
| this that every three another the |
| last next few recent earlier later past |
| friday monday tuesday wednesday |
| each every ago any two all one some |
| the on of 's between itself 37 1967 |

Table 2.4: Examples of semantic classes and the five most probable classes that are generated after them.

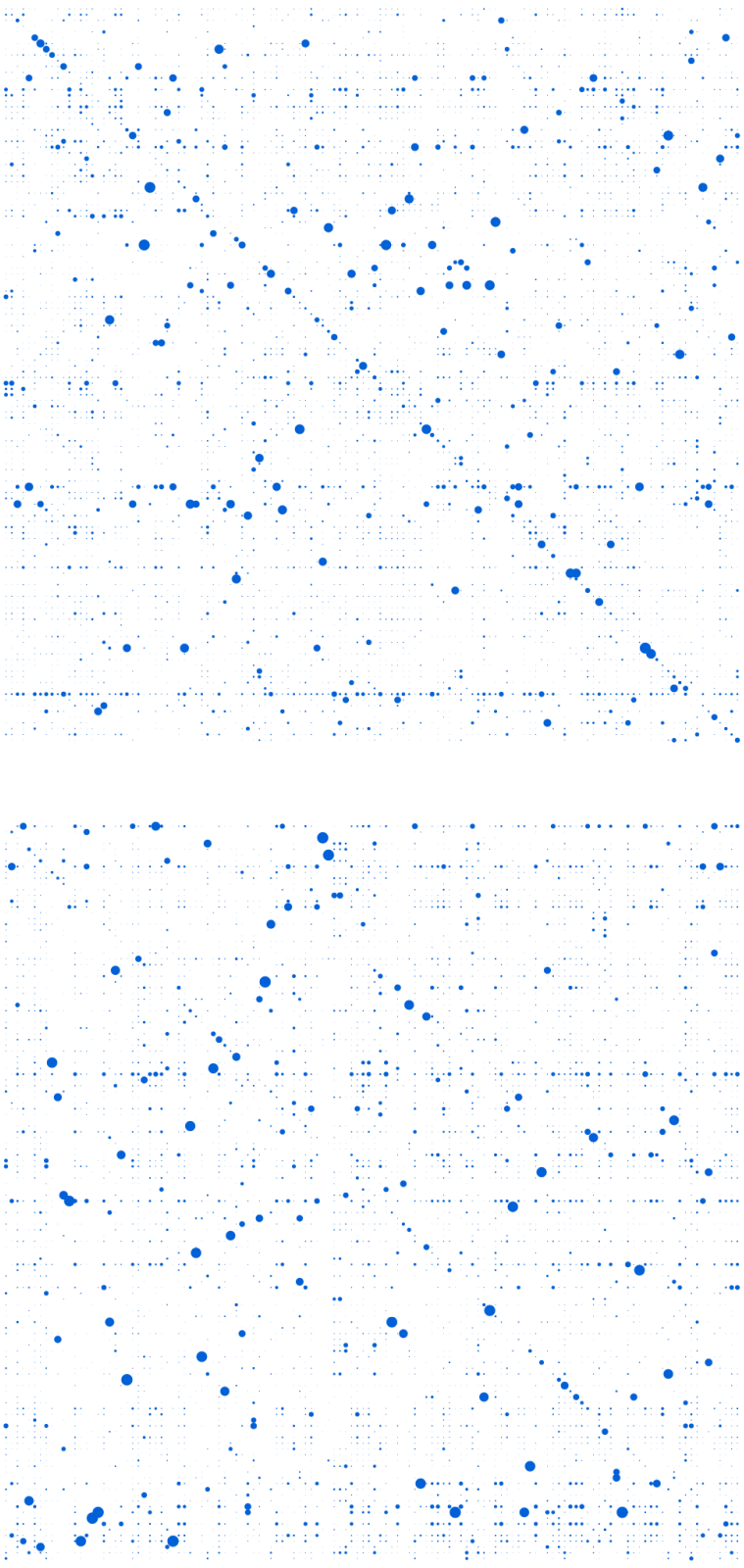


Figure 2.4: Transition matrices of a tree model (top) and a chain model (bottom), with 128 latent classes.

### 2.3.4 Vectorial representation of words

The models we introduced can also be used to represent words as vectors. Indeed, we can associate to each word the corresponding posterior distribution of latent classes, averaged over all the appearances of that word in the training corpus. More formally, given a training set of $N$ sentences $(\mathbf{w}^{(n)})_{n\in\{1,\dots,N\}}$ and an integer $a$ representing a word type, we define the vector $\tilde{\mathbf{u}}$ by

$$\tilde{u}_i = \sum_{n=1}^{N}\sum_{k=1}^{K_n} \mathbb{E}\left[\mathbf{1}\{\, C_k^{(n)} = i,\ W_k^{(n)} = a\} \mid W^{(n)} = \mathbf{w}^{(n)}\right].$$

Then, the word type $a$ is represented by the normalized vector

$$\mathbf{u} = \frac{\tilde{\mathbf{u}}}{||\tilde{\mathbf{u}}||_1}.$$

We note that the vector $\tilde{\mathbf{u}}$ corresponds to the $a$th line of the sufficient statistics matrix $\omega$. Thus, we use the sufficient statistics computed during learning to extract vectorial representation of words (even if the model was trained using online EM). We now present some applications of this vectorial representation of words, such as data visualization or distributional thesaurus building.

#### Distributional thesaurus

In a distributional thesaurus, each word of the vocabulary is associated to a list of related words. Using the vectorial representation of words we just introduced, it is possible to compute the similarity between two words by computing the similarity between their associated vectors $\mathbf{u}$ and $\mathbf{v}$, for example by using the Bhattacharyya coefficient defined by

$$BC(\mathbf{u},\mathbf{v}) = \sum_i \sqrt{u_i v_i}.$$

We will compare different distances between probability distributions in Chapter 3. In order to build a distributional thesaurus, we then associate to each word of the vocabulary its $k$ most similar terms, according to the similarity measure we have just introduced. We give such examples in Table 2.5 and Table 2.6. We observe that the similarity induced by our models does not differentiate between synonyms and antonyms: for example, both *sell* and *buy* appear in the thesaurus entry for the verb *purchase*. We also notice that for nouns, most of the related terms are co-hyponyms of the target word. We will investigate what kind of relations are favored by our models in more details in Chapter 3.

| | |
|---|---|
| **sunday-n**: | saturday(0.96), thursday(0.92), friday(0.92), tuesday(0.92) |
| **linguistics-n**: | biology(0.86), psychology(0.79), mathematics(0.79), physics(0.73) |
| **salt-n**: | pepper(0.82), flour(0.78), vinegar(0.77), parsley(0.76), garlic(0.76) |
| **red-j**: | yellow(0.92), brown(0.90), green(0.89), orange(0.87), blue(0.85) |
| **cute-j**: | pretty(0.87), beautiful(0.80), funny(0.78), nice(0.71), neat(0.69) |
| **stunning-j**: | remarkable(0.79), dramatic(0.74), impressive(0.71), spectacular(0.71) |
| **purchase-v**: | sell(0.87), buy(0.85), market(0.79), acquire(0.78), handle(0.74) |
| **cook-v**: | bake(0.93), saute(0.86), brown(0.82), simmer(0.80), sprinkle(0.80) |
| **nominate-v**: | elect(0.72), appoint(0.71), name(0.68), resign(0.65), select(0.62) |
| **slowly-a**: | smoothly(0.90), quickly(0.86), easily(0.82), freely(0.78), rapidly(0.77) |
| **financially-a**: | economically(0.91), politically(0.84), potentially(0.80), seemingly(0.79) |
| **definitely-a**: | always(0.93), obviously(0.89), really(0.84), basically(0.83) |

Table 2.5: Examples of distributional thesaurus entries of nouns (-n), adjectives (-j), verbs (-v) and adverbs (-a). The number in parentheses is the similarity score between the term and its target, induced by our model. A tree model with 512 latent classes trained on the NYT corpus was used to compute those similarity scores.

| | |
|---|---|
| **dimanche-n**: | samedi(0.92), mercredi(0.86), jeudi(0.84), vendredi(0.84) |
| **avion-n**: | hélicoptère(0.82), camion(0.81), voiture(0.79), char(0.78) |
| **ville-n**: | village(0.84), commune(0.82), localité(0.78), principauté(0.77) |
| **rouge-j**: | orange(0.90), brun(0.87), bleu(0.85), violet(0.85), jaune(0.85) |
| **chaud-j**: | doux(0.91), sec(0.87), frais(0.85), sombre(0.84), froid(0.81) |
| **épais-j**: | mince(0.91), plat(0.89), coloré(0.89), ovale(0.89) |
| **acheter-v**: | racheter(0.86), louer(0.83), récupérer(0.76), acquérir(0.75) |
| **construire-v**: | aménager(0.94), édifier(0.94), bâtir(0.94), ériger(0.92) |
| **gagner-v**: | perdre(0.80), remporter(0.79), obtenir(0.79), décrocher(0.79) |
| **lentement-a**: | rapidement(0.86), fermement(0.86), volontairement(0.83) |
| **parfois-a**: | quelquefois(0.93), souvent(0.82), généralement(0.80), simplement(0.80) |
| **pourtant-a**: | cependant(0.86), toutefois(0.84), certes(0.84), paradoxalement(0.81) |

Table 2.6: Examples of distributional thesaurus entries of nouns (-n), adjectives (-j), verbs (-v) and adverbs (-a). The number in parentheses is the similarity score between the term and its target, induced by our model. A tree model with 512 latent classes trained on the French Wikipedia was used to compute those similarity scores.

**Word visualization**

A second application of vectorial representation of words is to visualize words in the two-dimensional plan. To that end, vectors corresponding to selected words are gathered, and a dimensionality reduction technique, such as multi-dimensional scaling is applied to these vectors. The 2-dimensional vectors are then plotted, and it is then possible to evaluate which words are similar, according to the model. We have plotted some vectors corresponding to words in Figure 2.5. We observe that similar words tend to form clusters: for example, words designing vehicles (*plane, car, bus, etc.*) are close together. Ambiguous words, such as *duck* which can design the food or the animal are between the two corresponding clusters.

Of course, it is also possible to obtain a vector representing a word in the context of a sentence, by computing the posterior distribution of latent classes for that token. Given a sentence $\mathbf{w}$, the vector $\mathbf{u}$ representing the $k^{th}$ token of the sentence is defined by

$$u_i = \mathbb{E}\left[\mathbf{1}\{C_k = i\} \mid W = \mathbf{w}\right].$$

We computed the posterior distributions of latent classes corresponding to the word *head* in the two following sentences, and compared them with posterior distributions representing the word *head* out of context and words designing body parts (*eye, hand, should, etc.*) or leaders (*chairman, director, manager, etc.*):

> *1: A well-known Wall Street figure may join the Cabinet as* head *of the Treasury Department.*
>
> *2: The nurse stuck her* head *in the room to announce that Dr. Reitz was on the phone.*

First, we observe that the posterior distribution representing the word *head* is between two clusters, the first one formed by words designing leaders and the second one formed by words referring to body parts. Second, we observe that the posterior distribution of latent classes representing words in context are shifted toward the clusters corresponding to the sense of the ambiguous word *head*.

### 2.3.5 On optimization parameters

We briefly discuss the different choices that can influence the learning efficiency in the proposed models. In practice, we have not observed noticeable differences between *$\varepsilon$-best projection* and *$k$-best projection* for the approximate inference, and we thus advise to use the latter as its complexity is controled. By contrast, initialization can greatly change the performance in semi-supervised learning, in particular for tree models. We thus advise to initialize with Brown clusters. We will discuss this in greater details in Chapter 4. Finally, as noted by Liang and Klein (2009), the step size of online EM also has a significant impact on performance.

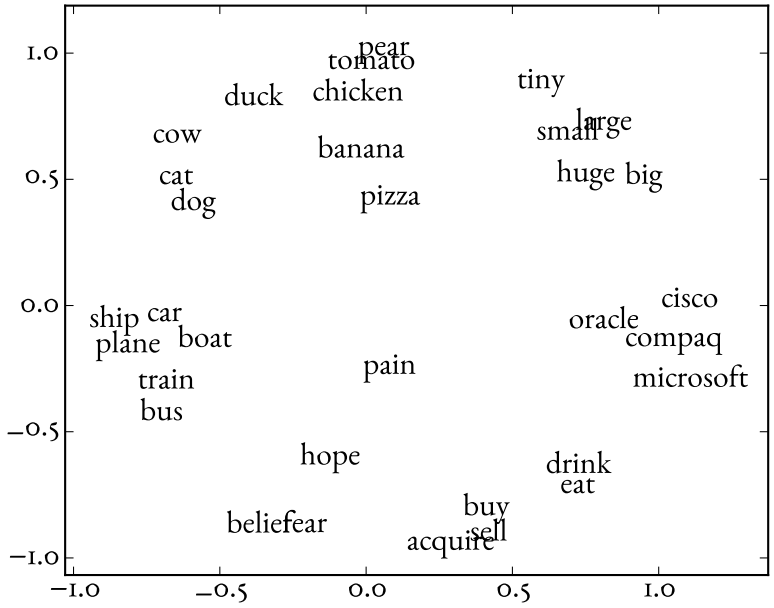



Figure 2.5: Out of context words visualization.

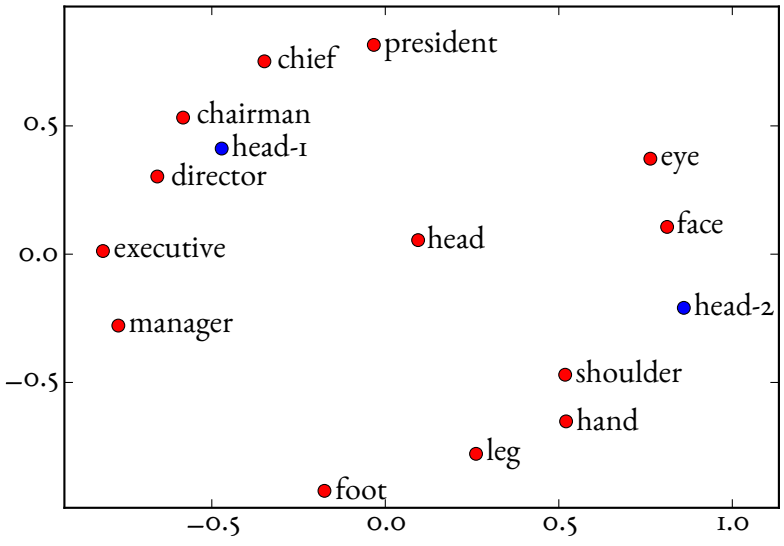



Figure 2.6: Word representation in context: red dot represent words out of context, while blue dots represents words in context. See text for details.

## 2.4 Relation to previous work

Brown clustering (Brown et al., 1992) is the most commonly used method for word cluster induction, and is often used for semi-supervised learning (see Chapter 4). The goal of this algorithm is to discover a clustering function $\mathscr{C}$ from words to clusters which maximizes the likelihood of the data, assuming the following sequential model of sentences:

$$\prod_k p(w_k \mid \mathscr{C}(w_k)) p(\mathscr{C}(w_k) \mid \mathscr{C}(w_{k-1})).$$

One of the limitations of this model is the fact that it does not take into account polysemy and homonymy. Each word of the vocabulary is assigned to a single cluster. This also means that there is no way to obtain representations of words in context.

In recent work, Huang et al. (2011, 2013) proposed to address this limitation by using a hidden Markov model to learn word representations for semi-supervised learning. The chain model we consider is equivalent to their hidden Markov model representation. They also proposed to replace the latent Markov chain by a lattice of $K \times N$ binary variables, $K$ being the length of the sentence. The motivation is that words have different "features", such as gender, number, tense, aspect, semantic role, *etc* and those features should be captured by the various binary latent variables. Syntax, that we aimed to take into account, is ignored by both these models.

Closest to our model is the variant of LDA proposed by Boyd-Graber and Blei (2009), in which syntactic trees are used to model dependencies between the different topics. Given that we aim for our classes to capture as much of the word semantics reflected by the syntax, such as the semantic roles of words, we believe that it is not necessarily useful or even desirable that the latent variables should be determined, even in part, by topic parameters that are sharing information at the document level. Moreover, our model being significantly simpler, we were able to design fast and efficient algorithms, making it possible to use our model on much larger datasets, and with many more latent classes.

## 2.5 Conclusion

In this chapter, we considered an arguably natural generative model of sentences for semantic class induction. It can be seen as a generalization of Brown clustering in two directions. First, replacing the Markov chain linking the semantic classes by a dependency tree allows our model to capture the syntax. Second, allowing each word to belong to multiple semantic classes permits to capture homonymy. We developed an efficient algorithm to perform inference and learning, which makes it possible to train this model on large datasets, such as the New York Times corpus. We showed that this model induces relevant semantic classes and relations between them.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Paris | art | président | an | blanc | publier | dire |
| Londres | histoire | directeur | mois | noir | écrire | penser |
| Rome | économie | membre | jours | rouge | sortir | savoir |
| Berlin | science | professeur | heure | petit | jouer | noter |
| Lyon | recherche | ministre | minute | vert | enregistrer | estimer |
| Marseille | musique | secrétaire | semaine | bleu | apparaître | considérer |
| Montréal | cinéma | député | année | jaune | composer | affirmer |

Table 2.7: Example of semantic classes obtained using French Wikipedia as a corpus.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Roma | filosofia | presidente | anno | conquistare | bianco | scrivere |
| Milano | arte | direttore | giorno | lasciare | nero | interpretare |
| Venezia | storia | membro | mese | occupare | colore | suonare |
| Torino | scienze | capo | tempo | sconfiggere | rosso | pubblicare |
| Firenze | medicina | professore | ora | attaccare | blu | cantare |
| Parigi | teologia | segretario | minuto | distruggere | scuro | realizzare |
| Napoli | arti | generale | episodio | abbandonare | verde | fare |

Table 2.8: Example of semantic classes obtained using Italian Wikipedia as a corpus.

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| membrane | cell | association | treatment | analyze | increase |
| nucleus | lymphocyte | relationship | exposure | evaluate | change |
| surface | fibroblast | correlation | stimulation | assess | reduction |
| cytoplasm | macrophage | difference | injection | examine | decrease |
| tissue | progenitor | interaction | administration | determine | difference |
| structure | xenograft | relation | transfection | investigate | improvement |
| matrix | hepatocyte | link | incubation | measure | variation |

Table 2.9: Example of semantic classes obtained using biomedical abstracts as a corpus.

# CHAPTER 3



# INTRINSIC EVALUATIONS

EVALUATING distributional model of semantics is not easy. Traditionally, people have performed two kinds of evaluation: intrinsic evaluations, in which the quality of the similarity measure induced by the model is evaluated against a gold standard, such as human similarity judgements or lexical databases such as WordNet, and extrinsic evaluations, in which people investigate how helpful is their model for solving other tasks, such as part-of-speech tagging or named entity recognition. In this chapter, we perform intrinsic evaluations of our models, while extrinsic evalutions are carried out in the next chapter.

## 3.1 Predicting similarity judgements

In this section, we evaluate our models on a similarity prediction task. In this task, pairs of words are presented to human subjects, who are asked to rate the *relatedness* of the two words. We then compare the human judgements with the distributional similarity induced by our models by computing the correlation between the two score distributions, for example by using the Pearson correlation coefficient. We use the WordSim353 dataset, collected by Finkelstein et al. (2001). This dataset comprises 353 word pairs, which were rated by 13 to 16 human subjects on a $0-10$ scale, $0$ meaning that the two words are completely unrelated, while $10$ means that the two words are very much related or identical. Agirre et al. (2009) proposed to evaluate distributional semantics models on two subsets of the WordSim353 dataset, the first one grouping words that are *similar* and the second one grouping word that are *related*. Similar words are defined as synonyms, antonyms, and hyperonym-hyponym, while related words are defined as meronym-holonym and topically related words. See Table 3.1 for examples of word pairs of the WordSim353 dataset.

### Comparison of similarity measures

Each word is represented by its corresponding posterior distribution of latent semantic classes, averaged on the whole training corpus. Since words are represented by a probability distribu-

| word 1 | word 2 | score | relation | split |
|---|---|---|---|---|
| tiger | tiger | 10.00 | identical | S |
| dollar | buck | 9.22 | synonymy | S |
| dollar | profit | 7.38 | topic | R |
| smart | stupid | 5.81 | antonymy | S |
| smart | student | 4.62 | topic | R |
| psychology | discipline | 5.58 | hyponymy | S |
| psychology | cognition | 7.48 | topic | R |
| planet | moon | 8.08 | co-hyponymy | S |
| planet | galaxy | 8.11 | meronymy | R |

Table 3.1: Examples of word pairs from the WordSim353 dataset. The split column indicates to which subset the word pair belongs: S stands for similar, while R stands for related.

tion, we have considered the following measures to compute the similarity between two words: the symmetrised Kullback-Leibler divergence

$$D_{KL}(p,q) = \frac{1}{2}\sum_{i=1}^{n} p_i \ln\left(\frac{p_i}{q_i}\right) + q_i \ln\left(\frac{q_i}{p_i}\right),$$

the $\chi^2$-distance

$$D_{\chi^2}(p,q) = \sum_{i=1}^{n} \frac{(p_i - q_i)^2}{p_i + q_i},$$

the Jensen-Shannon divergence

$$D_{JS}(p,q) = \frac{1}{2}\sum_{i=1}^{n} p_i \ln\left(\frac{2p_i}{p_i + q_i}\right) + q_i \ln\left(\frac{2q_i}{p_i + q_i}\right)$$

and the Hellinger distance:

$$D_H(p,q) = \sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2.$$

We also included the cosine similarity measure as a baseline, as it is widely used in the field of distributional semantics. We report results on the whole WordSim353 dataset in Table 3.2, for various model sizes. Unsurprisingly, we observe that the dissimilarity measures giving the best results are the one tailored for probability distribution, namely the Jensen-Shannon divergence and the Hellinger distance. The Kullback-Leibler divergence is too sensitive to fluctuations of small probabilities and thus does not perform as well as other similarity measures between probability distributions. In the following, we will use the Hellinger distance, unless otherwise stated.

| | tree | | | chain | | |
|---|---|---|---|---|---|---|
| | 128 | 256 | 512 | 128 | 256 | 512 |
| Cosine | 0.34 | 0.35 | 0.33 | 0.35 | 0.33 | 0.36 |
| KL-divergence | 0.34 | 0.38 | 0.39 | 0.31 | 0.32 | 0.35 |
| Chi-squared | 0.37 | 0.38 | 0.38 | 0.39 | 0.39 | 0.39 |
| Jensen-Shannon | 0.37 | 0.38 | 0.39 | 0.40 | 0.39 | 0.40 |
| Hellinger | 0.37 | 0.39 | 0.40 | 0.40 | 0.39 | 0.40 |

Table 3.2: Absolute value of the Pearson correlation coefficient between human relatedness judgements and distances induced by our models.

**Relatedness v.s. similarity**

As we mentioned before, words might be rated as related for different reasons and there exist different relations between related words, such as synonymy, antonymy or meronymy. Words can even be rated as similar only because they tend to appear in the same contexts, such as *guitar* and *music*, even if there is not relation between them. Such words are refered to as topically related. In order to determine what kind of relations are captured by our models, we evaluated them on the two subsets of the WordSim353 dataset proposed by Agirre et al. (2009), namely related words and similar words. We report results in Table 3.3. We observe that both models capture similarity much better than relatedness. This is not surprising at all since for both models, word order or syntactic roles are very important. Thus, these models tend to rate as similar words that are truly *exchangeable*, and not topically related words.

| | tree | | | chain | | |
|---|---|---|---|---|---|---|
| | 128 | 256 | 512 | 128 | 256 | 512 |
| Relatedness | 0.22 | 0.23 | 0.23 | 0.24 | 0.22 | 0.21 |
| Similarity | 0.57 | 0.60 | 0.60 | 0.60 | 0.63 | 0.63 |

Table 3.3: Absolute value of the Pearson correlation coefficient between human relatedness judgements and distances induced by our models.

## 3.2 BLESS

As we saw in the previous section, different relations exists between words and a distributional semantic model does not necessarily capture all those relations equally well. Determining what kind of relations are favored by a model is thus as important as evaluating to what extent the model captures relatedness in general. This lead Baroni and Lenci (2011) to design and publish the BLESS dataset. The dataset comprises 200 concrete concepts. For each concept, a list of

related words referred to as relatum, is given, with the type of the relation. Five relations are considered: co-hyponymy, hypernymy, meronymy, attribute and event. The attribute relation means that the relatum is an adjective expressing an attribute of the concept, while the event relation means that the relatum is verb designing an activity or event in which the concept is involved. Finally, random nouns, adjectives and verbs are added, to estimate the preference of a model towards related terms over random ones. Examples of concept-relation-relatum are given in Table 3.4.

| **concept** | **relation** | **relatum** |
|---|---|---|
| library | co-hyponymy | restaurant |
| library | meronymy | door |
| library | hypernymy | institution |
| library | attribute | public |
| library | event | build |
| library | rand-n | crime |
| library | rand-j | important |
| library | rand-v | surround |

Table 3.4: Examples of concept-relation-relatum triples from the BLESS dataset introduced by Baroni and Lenci (2011).

### Comparison of relations captured by our models

We follow the evaluation proposed by the authors: for each concept and each relation, we keep the score of the closest relatum. Thus, for each concept, we have eight scores, one for each relation. We normalize these eight scores (mean: 0, std: 1), in order to reduce concept-specific effects, such as denser neighborhood. We then report these score distributions for each relation as box plots in Figure 3.1. It is thus possible to analyse which relations are favored by the model by comparing the score distributions.

We observe that tree models and chain models tend to have the same distributions of similarity scores for the different relations, and thus to capture the same kind of lexical information. Both models favor the co-hyponymy relation by a large margin. It is followed by the hypernymy and meronymy relations. The fourth relation is the random-n relation, which is prefered over the attribute and the event relations. This happens because words with similar part-of-speech tend to share the same semantic classes, while words with different part-of-speech appear in disjoint semantic classes. It is thus impossible to compare words with different parts-of-speech and to capture relation such as the event or the attribute relation as defined by the BLESS dataset. We now present a way to address this limitation.

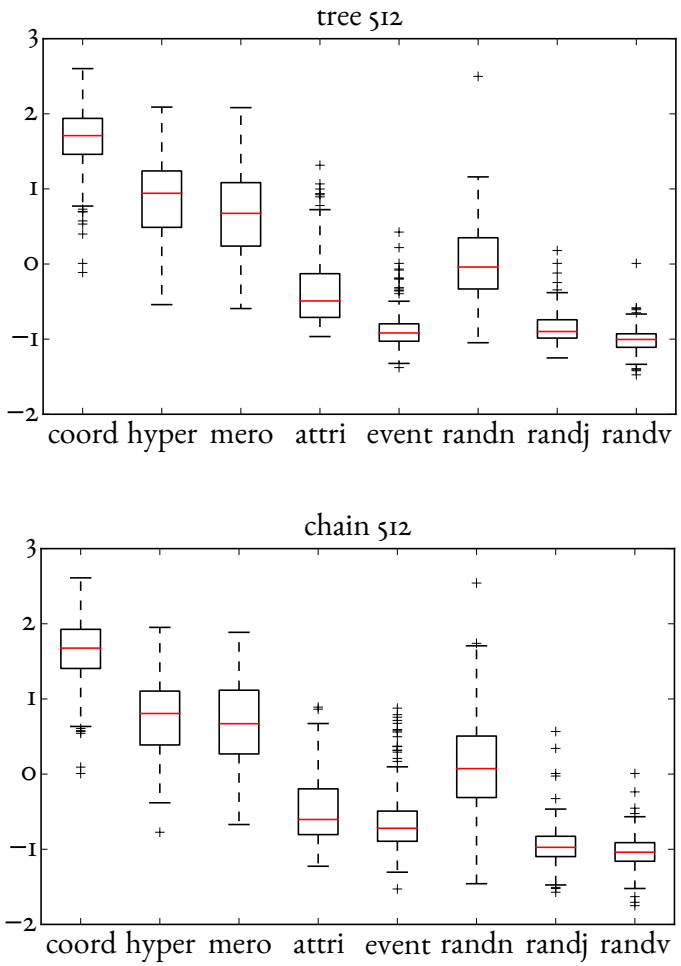



Figure 3.1: Distributions of similarity scores for different relations, for a tree model (top) and a chain model (bottom).

**Transforming adjectives into nouns and nouns into verbs**

In syntactic relations between nouns and adjectives, the noun is the head word and the adjective is the dependent. Similarly, in syntactic relations between nouns and verbs, most often, the verb is the head and the noun is the dependent. Given a vector $\mathbf{v}_a$ representing an adjective and a vector $\mathbf{v}_n$ representing a noun, it is thus natural to left multiply them by the transition matrix of the model to obtain a vector $\mathbf{u}_a$ comparable to nouns and a vector $\mathbf{u}_n$ comparable to verbs:

$$\mathbf{u}_a = \mathbf{T}^\top \mathbf{v}_a \quad \text{and} \quad \mathbf{u}_n = \mathbf{T}^\top \mathbf{v}_n.$$

For chain model, transitions are reversed: there is a transition from adjectives to nouns and a transition from subject to verbs.[1] Thus, we right multiply the vectors $\mathbf{v}_a$ and $\mathbf{v}_n$ by the transition matrix to obtain $\mathbf{u}_a$ and $\mathbf{u}_n$:

$$\mathbf{u}_a = \mathbf{T}\mathbf{v}_a \quad \text{and} \quad \mathbf{u}_n = \mathbf{T}\mathbf{v}_n.$$

We report the new score distributions obtained when adjective and noun representations are transformed when compared to nouns and verbs in Table 3.2. We observe that, when using these transformations, the attribute and event relations are preferred over the random relations, which was the goal of applying them.

**Related words retrieval**

A second way to evaluate distributional semantic models using the BLESS dataset is related words retrieval. In the BLESS dataset, each concept is associated to approximately the same number of related words and random words. For each concept, the associated words can be ranked accordingly to their similarity with the concept and the positions of related words in that ranking indicate how good is the similarity measure: related words should be ranked higher than random words. We report the precision-recall curves for various models in Figure 3.3: tree models perform better than chain models and the performance improves with the number of latent states.

## 3.3 Word categorization

In this section, we evaluate our models on a word categorization task: given a set of words, the goal is to cluster them into semantic classes. We considered three datasets, that were used during the 2008 workshop *Bridging the gap between semantic theory and computational simulations*.[2] The first one is composed of concrete nouns, such as *dog*, *knife* or *rocket*, the second one is composed of both concrete and abstract nouns, examples of abstract nouns being *hope*, *concept* or *hypothesis*. Finally, the third dataset is composed of verbs, such as *talk*, *fly* or *eat*.

[1] We decided to ignore the transitions from verb to object.

[2] http://wordspace.collocations.de/doku.php/workshop:esslli:start

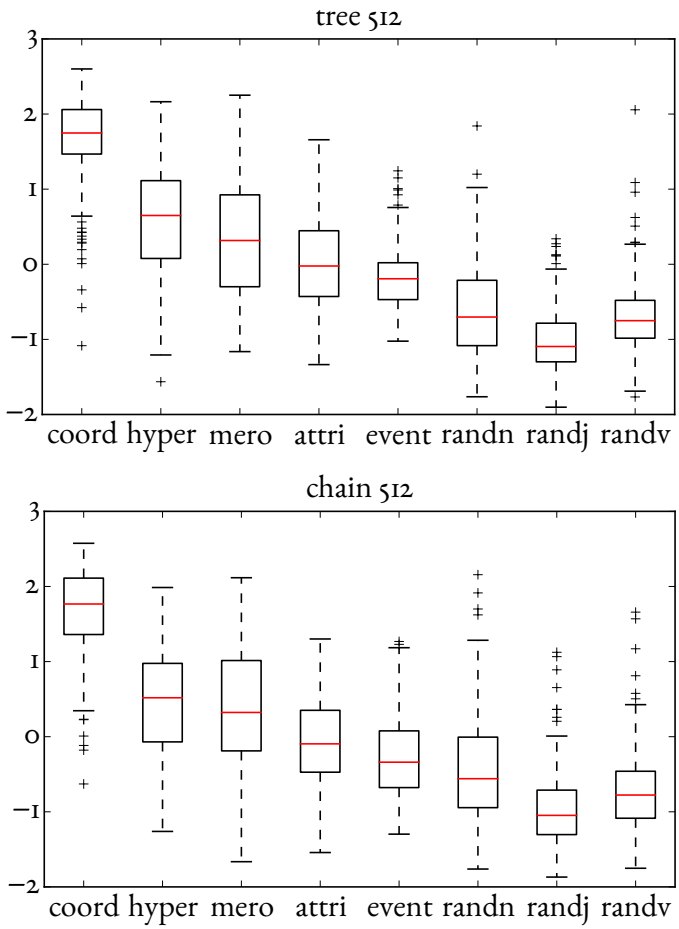



Figure 3.2: Distributions of similarity for different relations, for a tree model (top) and a chain model (bottom).

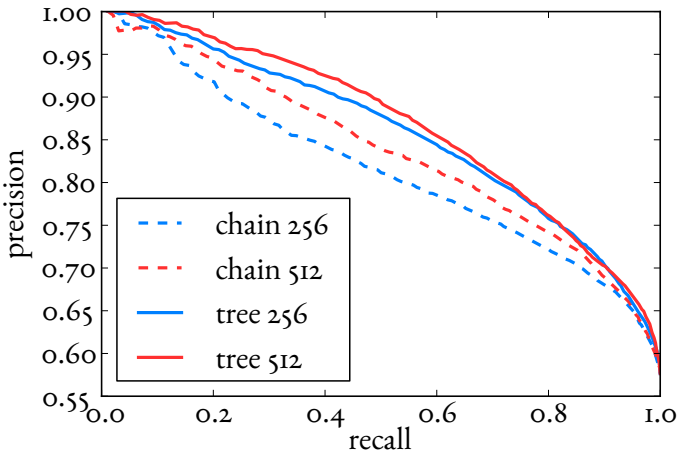



Figure 3.3: Precision-recall of retrieved terms.

**Methodology**

As before, we represent a word by the corresponding posterior distribution of semantic classes over the training corpus and we use the Hellinger distance as a similarity measure between words. We then use an agglomerative clustering algorithm, with complete linkage. We also tried single linkage and average linkage, but complete linkage outperforms the two others for both chain and tree models. Following the guidelines of the shared task from which the dataset comes from, we evaluate the quality of a clustering using two metrics: entropy and purity. Given a cluster $S$, the entropy is equal to

$$\mathrm{En}(S) = -\frac{1}{\log(C)} \sum_{c=1}^{C} f_c \log\left(f_c\right),$$

where $f_c$ is the proportion of elements from the gold class $c$ appearing in the cluster $S$ and $C$ is the number of gold classes. It measures how much the cluster $S$ mixes elements from different classes. The purity of the cluster $S$ is equal to

$$\mathrm{Pu}(S) = \max_c f_c.$$

It measures the proportion of the largest class appearing in $S$. Both these measures are in the range $[0, 1]$. The lower the entropy the better, and the higher the purity the better. Finally, each measure is averaged accross clusters, weighted by the size of each cluster.

### 3.3.1 Concrete nouns categorization

The concrete nouns dataset comprises forty-four nouns, coming from six fine grained semantic categories: bird (*eagle*), ground animal (*cow*), fruit tree (*pear*), green good (*lettuce*), tool (*spoon*) and vehicle (*helicopter*). These categories can be merged into animal, vegetable and artifact, giving rise to a coarser clustering problem. We will report results on both problems, referred to as 3-classes and 6-classes. Out of the forty-four nouns, six were removed from the evaluation set, since they do not appear in our training corpus (*peacock*, *snail*, *pineapple*, *scissors*, *kettle* and *chisel*).

| | | **tree** | | | **chain** | | |
|---|---|---|---|---|---|---|---|
| | | 128 | 256 | 512 | 128 | 256 | 512 |
| 6-classes | Purity | 0.66 | 0.71 | **0.76** | 0.58 | 0.58 | 0.71 |
| | Entropy | 0.41 | 0.28 | **0.27** | 0.51 | 0.47 | 0.42 |
| 3-classes | Purity | 0.55 | 0.87 | **0.95** | 0.50 | 0.63 | 0.87 |
| | Entropy | 0.72 | 0.29 | **0.16** | 0.87 | 0.62 | 0.34 |

Table 3.5: Purity and entropy scores for the concrete nouns clustering task.

First of all, we observe that larger models perform better than smaller ones. Second, for both problems and both metrics, the tree models, which take syntax into account, outperform the chain models by a large margin. This phenomenon was also noticed by Van de Cruys (2010).

| Cluster | 2 | 3 | 1 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Bird | 2 | 3 | 1 | - | - | - |
| Ground animal | 1 | 6 | - | - | - | - |
| Fruit tree | - | - | 3 | - | - | - |
| Green good | - | - | 5 | - | - | - |
| Tool | - | - | - | 1 | 9 | - |
| Vehicle | 1 | - | - | - | - | 6 |

Table 3.6: Confusion matrix of the best clustering for the 6-classes problem.

In order to understand the decisions made by our model, let us examine the errors of the best clustering for the 6-classes problem. We report the corresponding confusion matrix in Table 3.6 and clusters in Table 3.7. First, the semantic classes corresponding to food are merged into cluster 1. The bird appearing in cluster 1 is the word *chicken*, which is polysemous and can also refer to food. Similarly, the semantic classes corresponding to animals are merged into cluster 3. Cluster 2 might seem surprising, consisting of the words *penguin*, *eagle*, *lion* and *rocket*. In fact, all those words are names of American sport teams.[3] Since our corpus is composed of news articles, it is not surprising that this sense dominates the other. Finally, cluster 4 consists of the word *telephone* alone, which might be seen as an outlier in the semantic class comprising tools.

| | |
|---|---|
| **Cluster 1:** | *chicken, lettuce, pear, onion, potato, mushroom, corn, banana, cherry* |
| **Cluster 2:** | *eagle, lion, penguin, rocket* |
| **Cluster 3:** | *duck, dog, cat, cow, swan, owl, elephant, turtle, pig* |
| **Cluster 4:** | *bottle, spoon, pencil, knife, screwdriver, hammer, pen, cup, bowl* |
| **Cluster 5:** | *telephone* |
| **Cluster 6:** | *boat, car, truck, ship, helicopter, motorcycle* |

Table 3.7: Clusters of the best clustering for the 6-classes problem.

[3]Pittsburgh Penguins, Philadelphia Eagles, Detroit Lions and Houston Rockets.

### 3.3.2 Abstract v.s. concrete nouns categorization

The second dataset contains 15 highly concrete nouns, such as *truck*, *turtle* or *onion*, 15 highly abstract nouns, such as *pride*, *concept* or *hypothesis* and 10 nouns with medium concreteness, such as *pollution*, *empire* or *smell*. The first task is to cluster the 15 concrete nouns *v.s.* the 15 abstract nouns, in order to evaluate the ability of a model to make the distinction between concrete and abstract concepts. We refer to this task as conc/abst. In the second task, the goal is to cluster all the nouns, including those with medium concreteness, into two clusters and to examine for each noun with medium concreteness whether it is classified as concrete or abstract, and analyze why is so. Out of the forty nouns, two were removed from the evaluation set since they do not appear in our training corpus (*jealousy* and *ache*).

| | | **tree** | | | **chain** | | |
|---|---|---|---|---|---|---|---|
| | | 128 | 256 | 512 | 128 | 256 | 512 |
| conc/abst | Purity | **1.0** | **1.0** | **1.0** | 0.96 | **1.0** | **1.0** |
| | Entropy | **0.0** | **0.0** | **0.0** | 0.18 | **0.0** | **0.0** |

Table 3.8: Purity and entropy scores for the concrete *v.s.* abstract nouns clustering.

Results on the conc/abst task are reported in Table 3.8. We observe that, with the exception of the chain model with 128 latent states, all models perfectly discriminate concrete nouns *v.s.* abstract nouns. The only misclassified example by the 128 chain model is the abstract noun *hypothesis*. We do not have a compelling explanation for that error.

Let us now discuss the results of the second task. An example of obtained clustering is reported in Figure 3.4, for the dataset with and without the nouns with medium concreteness. First of all, we observe that adding the nouns with medium concreteness does not make a big difference in the obtained clustering. Indeed, clusters of nouns with medium concreteness are merged at the end of the algorithm, meaning that they are far from concrete and abstract nouns. The first cluster mixing concrete nouns with mildly concrete nouns is

{ *smell*, *shape*, *bottle* }

at a distance of 0.58.[4] The nouns *ceremony*, *invitation*, *fight*, *pollution* and *weather* are classified as abstracts while *smell*, *shape*, *empire* and *foundation* are classified as concrete. The word *smell* and *shape* are clustered with the artefacts they usually describe. The word *empire* and *foundation* are clustered with *lion* and *eagle* because one of the dominant sense associated to those animals are the American sport teams. These are organizations, and it is thus not surprising that *foundation* or *empire* appear in the same cluster.

[4]We recall that we are using complete linkage agglomerative clustering. Thus, the distance between two clusters is the maximum distance between two words from each clusters.

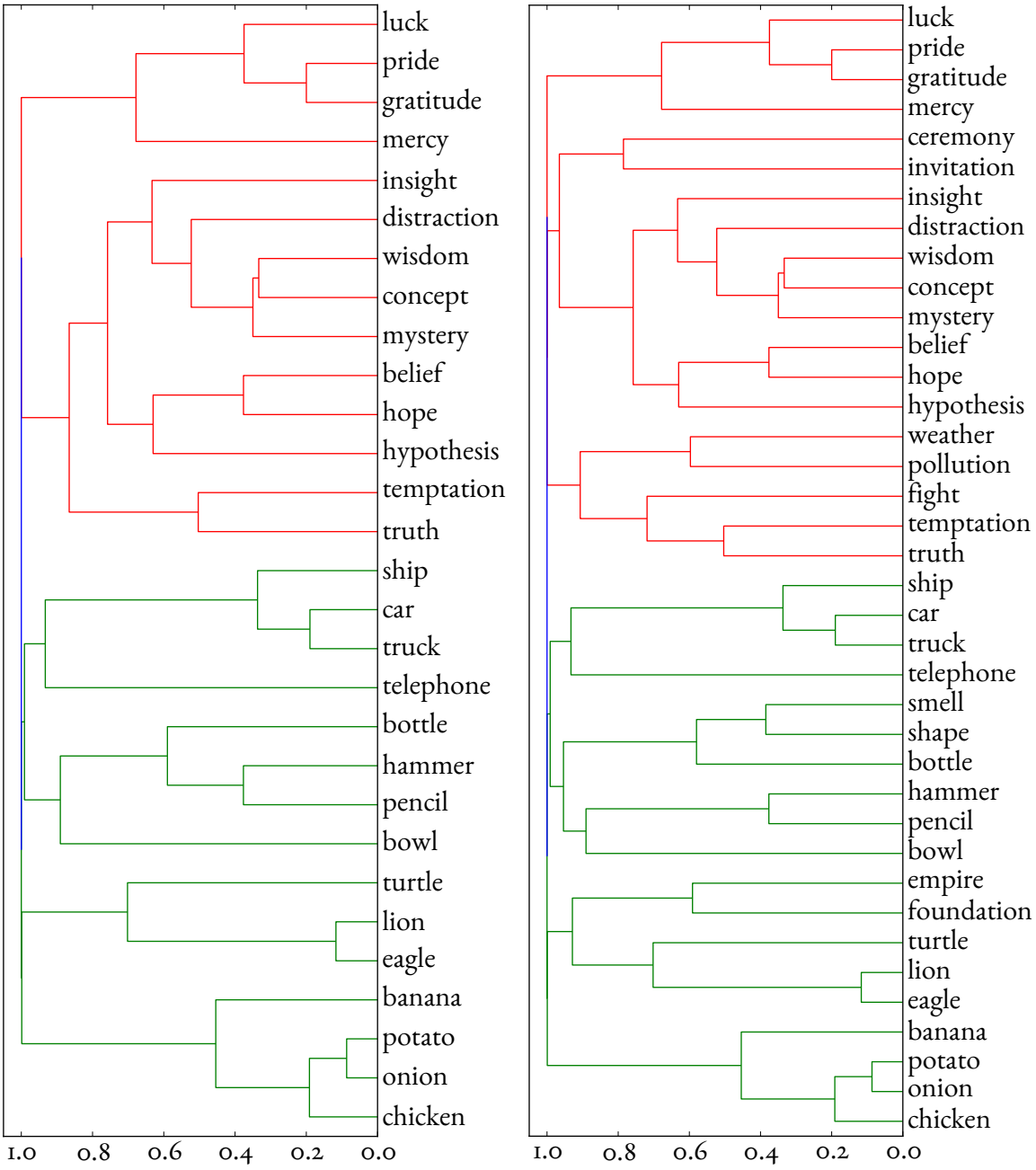



Figure 3.4: Dendrograms of the clustering obtained by using a tree model with 512 latent states, on the concrete *v.s.* abstract nouns dataset without mildly concrete nouns (left) and with mildly concrete nouns (right).

| | | tree | | | chain | | |
|---|---|---|---|---|---|---|---|
| | | 128 | 256 | 512 | 128 | 256 | 512 |
| verb-9 | Purity | 0.51 | **0.55** | 0.51 | 0.37 | 0.51 | 0.51 |
| | Entropy | 0.45 | **0.43** | 0.44 | 0.66 | 0.45 | 0.49 |
| verb-5 | Purity | 0.57 | 0.53 | **0.71** | 0.42 | 0.48 | 0.53 |
| | Entropy | 0.58 | 0.62 | **0.42** | 0.82 | 0.65 | 0.65 |

Table 3.9: Purity and entropy scores of verb clustering.

Overall, the proposed models are able to make the distinction between abstract and concrete concepts. Moreover, the choices made for mildly concrete nouns seem reasonable, and clusters which are not related to others are among the last ones to be merged.

### 3.3.3 Verbs categorization

The third dataset is composed of forty-five verbs that can be grouped into nine fine grained semantic classes: communication (*talk*), mental state (*remember*), motion manner (*drive*), motion direction (*leave*), change location (*carry*), body sense (*smell*), body action (*drink*), exchange (*sell*) and change state (*break*). These classes can be merged into the five coarser following classes: cognition, motion, body, exchange and change state. We refer to this two tasks as verb-9 and verb-5 respectively.

We report results in Table 3.9. As for the concrete nouns categorization, we observe that tree models perform better than chain models. Let us now discuss in details the clustering obtained by using a tree model with 512 latent states, on the verb-9 task. We provide the confusion matrix of this clustering in Table 3.10. The semantic classes with the highest number of misclassifactions are communication, motion direction, body sense and change state. Our system makes true mistakes, such as mixing verbs from the classes mental states, exchange and change state into the cluster 8:

{ *evaluate, acquire, buy, sell, destroy, repair* }.

A possible explanation for this cluster is the fact that all those verbs are transitive verbs whose object could be property or all sorts of manufactured goods.

But it also makes reasonable mistakes, such as confusing verbs from the different motion classes, classifying *smell* as a body action and not as body sense or mixing up communication, body sense and body action into the cluster 2:

{ *talk, speak, listen, look, smile, cry* }.

Finally, clustering *kill* and *die* away from the other change state verbs can be explained by the fact that they apply to animate thing, while the others apply to inanimate things. Similarly,

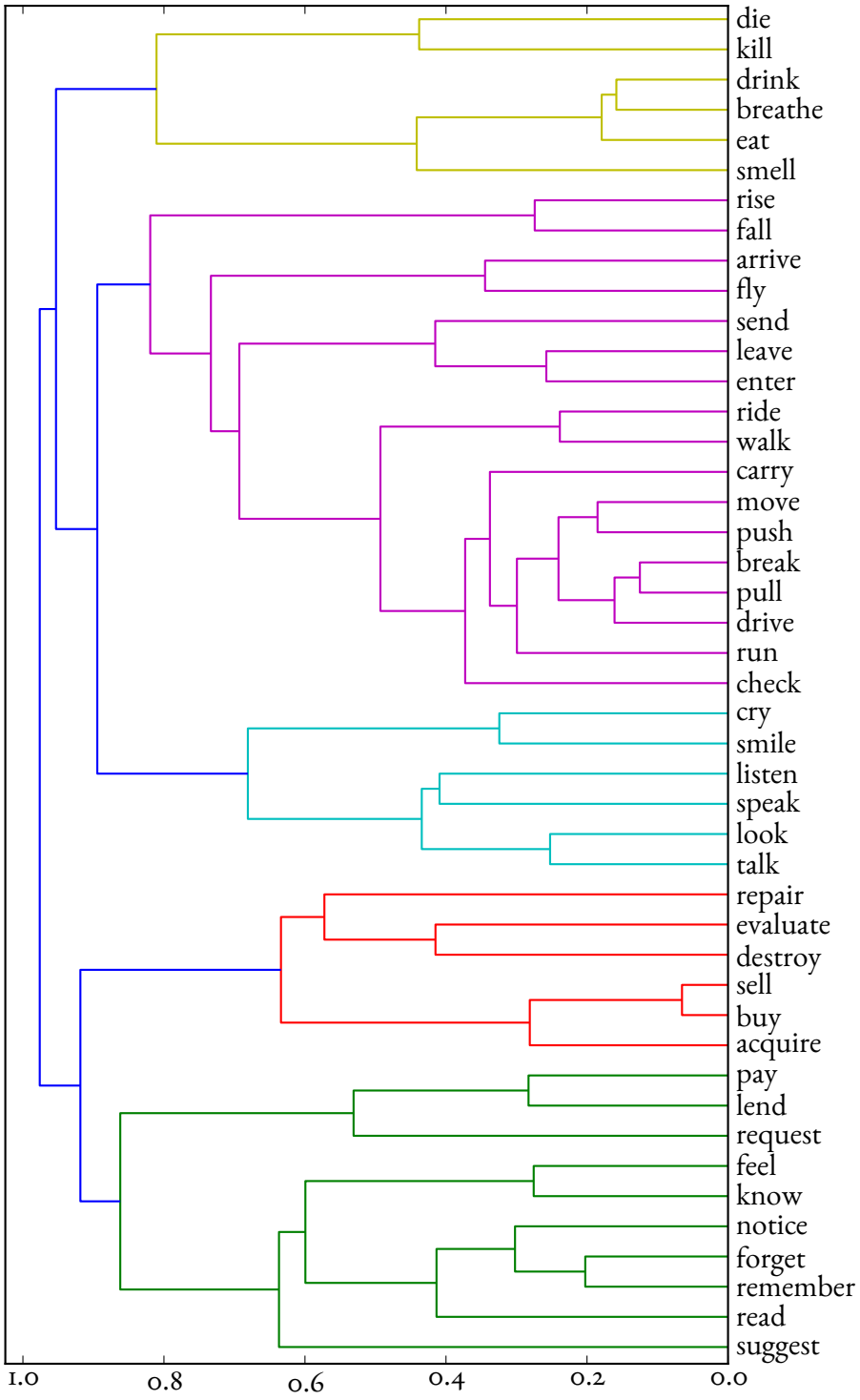



Figure 3.5: Dendrogram of the clustering obtained by using a tree model with 512 latent states, on the verb-5 task.

| Cluster | 3 | 1 | 5 | 6 | 4 | 2 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Communication | 1 | 2 | - | - | - | 2 | - | - | - |
| Mental state | - | 3 | - | - | 1 | - | - | 1 | - |
| Motion manner | - | - | 1 | - | 4 | - | - | - | - |
| Motion direction | - | - | 1 | 2 | 2 | - | - | - | - |
| Change location | - | - | - | - | 5 | - | - | - | - |
| Body sense | - | 2 | - | - | - | 2 | 1 | - | - |
| Body action | - | - | - | - | - | 2 | 3 | - | - |
| Exchange | 2 | - | - | - | - | - | - | 3 | - |
| Change state | - | - | - | - | 1 | - | - | 2 | 2 |

Table 3.10: Confusion matrix of the clustering obtained by using a tree model with 512 latent states, on the verb-9 task.

our system decides to cluster *fall* and *rise* away from other motion direction verbs, since in our dataset, these two verbs are mainly used to describe stock prices.

Clustering verbs is a notoriously hard task, and our proposed models fail on this problem like other methods do. Some of the mistakes they make can be explained by the highly polysemous nature of verbs or by the dataset bias. Moreover, there might potentially exist a larger number of meaningful ways to cluster verbs than nouns. However, we believe there is room for improvement, since there are still some true system mistakes.

## 3.4 Compositional semantics

In this section, we evaluate our models on semantic composition tasks. The principle of semantic compositionality, often attributed to the German logician Gottlob Frege, is the essential property of natural languages that the meaning of a complex unit such as a sentence, can be deduced from the meaning of its parts and the syntactic relations between them. This allows to express a potentially infinite number of ideas with a finite vocabulary. So far, we have only evaluated how well our models were able to capture the meaning of words taken as individual units. We now investigate how well our models capture the meaning of short phrases, such as adjective-noun, verb-object, compound noun or subject-verb-object phrases.

### Composition functions

Several ways to combine vectors representing individual words have been previously considered. Mitchell and Lapata (2008) considered vector functions such as addition, Hadamard product, convolution, Kronecker product or dilation. Baroni and Zamparelli (2010) represents nouns by vectors and adjectives by matrices and considered matrix-vector multiplication to combine the representations. We now introduce the composition functions we will consider in the following.

Let $\mathbf{u}$ be the vector representing the head word, and $\mathbf{v}$ be the vector representing the dependent word. The goal is to find a vector $\mathbf{p}$ that capture the meaning of the phrase formed by the two words. Following Mitchell and Lapata (2008), the two first functions we consider to combine $\mathbf{u}$ and $\mathbf{v}$ are the addition

$$\mathbf{p}_{add} = \mathbf{u} + \mathbf{v}$$

and the Hadamard product (a.k.a. elementwise product)

$$\mathbf{p}_{mult} = \mathbf{u} \otimes \mathbf{v}.$$

We also consider to first multiply the vector $\mathbf{v}$ representing the dependent word by the transition matrix of our Markov model, before using the addition

$$\mathbf{p}_{T-add} = \mathbf{u} + \mathbf{T}^\top \mathbf{v}$$

and the Hadamard product

$$\mathbf{p}_{T-mult} = \mathbf{u} \otimes \left(\mathbf{T}^\top \mathbf{v}\right).$$

We denote these methods by **T**-add and **T**-mult. As a baseline, we also consider to just compare the vectors representing the head words, without taking the dependent words into account. In that case, we have

$$\mathbf{p}_{baseline} = \mathbf{u}.$$

### 3.4.1 Mitchell and Lapata dataset

The first dataset we consider was introduced by Mitchell and Lapata (2010), and is composed of pairs of adjective-noun, compound-noun and verb-object phrases, whose similarities were evaluated by human subjects on a $1-7$ scale. Example of such phrases are given in Table 3.11. The goal is to predict the similarity judgements.

| type | phrase 1 | phrase 2 | score |
|---|---|---|---|
| adjective-noun | national government | cold air | 1 |
| adjective-noun | small house | little room | 4 |
| adjective-noun | special circumstance | particular case | 6 |
| compound noun | oil industry | railway station | 2 |
| compound noun | assistant secretary | company director | 5 |
| compound noun | state control | government intervention | 7 |
| verb-object | shut door | provide datum | 1 |
| verb-object | develop technique | use method | 5 |
| verb-object | stress importance | emphasise need | 7 |

Table 3.11: Examples of phrase pairs from the Mitchell and Lapata (2010) dataset.

| | tree 512 | | | chain 512 | | |
|---|---|---|---|---|---|---|
| | AN | NN | VN | AN | NN | VN |
| Baseline | 0.41 | 0.13 | 0.32 | 0.37 | 0.07 | 0.23 |
| Add | 0.46 | 0.35 | 0.38 | 0.43 | 0.30 | 0.31 |
| Mult | 0.04 | 0.31 | 0.01 | 0.01 | 0.18 | 0.05 |
| **T**-add | 0.43 | 0.30 | 0.40 | 0.42 | 0.33 | 0.35 |
| **T**-mult | 0.47 | 0.30 | 0.38 | 0.39 | 0.19 | 0.32 |
| Mitchell and Lapata | 0.46 | 0.49 | 0.38 | 0.46 | 0.49 | 0.38 |
| Humans | 0.52 | 0.49 | 0.55 | 0.52 | 0.49 | 0.55 |

Table 3.12: Pearson correlation coefficients between human similarity judgements and similarity computed by our models on the Mitchell and Lapata (2010) dataset. AN stands for adjective-noun, NN stands for compoundnoun and VN stands for verb-object.

We report the results for tree and chain models with 512 latent states in Table 3.12. The Hadamard product (Mult), which is one of the best composition function for traditional semantic vector space (Mitchell and Lapata, 2010), does not work at all for our representation for adjective-noun and verb-noun pairs. The reason is that latent states associated to words with different part-of-speech are often disjoint. Thus, when taking the Hadamard product of the two vectors, all coefficients are set to value near zero. The second observation is that there is no clear winner between between the remaining three methods. In particular, the Add method is surprisingly competitive. We believe this is the case because for most examples of the dataset, it is sufficient to independently compare both words of the phrases to evaluate their similarity, and not to compare them as phrases.

### 3.4.2 Grefenstette and Sadrzadeh dataset

The second dataset we consider was introduce by Grefenstette and Sadrzadeh (2011). Each example of this dataset consists in a triple of subject-verb-object, forming a small sentence, and a landmark verb. Human subjects were then asked to evaluate the similarity between the verb and its landmark in the context of the small sentences formed with the subject and the object. Since both verbs share the same context, the baseline, which only compares the non-contextualized verbs, is equivalent to comparing the words independently. Examples of triples and their associated landmarks are given in Table 3.13.

Following Van de Cruys et al. (2013), we compare the contextualized verb with the non-contextualized landmark, because it is believed to better capture the compositional ability of a model and it works better in practice. We report results for tree and chain models with 512 latent states in Table 3.14. First, we observe that the results for the tree and the chain models are comparable. Second, we observe that the **T**-mult composition function outperforms the three other functions by a large margin.

| subject | verb | object | landmark | score |
|---|---|---|---|---|
| scholar | write | book | publish | 7 |
| writer | write | book | spell | 3 |
| user | write | word | spell | 5 |
| user | write | word | publish | 2 |
| people | run | company | operate | 7 |
| people | run | company | move | 1 |
| people | run | round | operate | 1 |
| people | run | round | move | 7 |

Table 3.13: Examples of subject-verb-object triples and associated landmarks from the Grefenstette and Sadrzadeh (2011) dataset.

| | tree 512 | chain 512 |
|---|---|---|
| Baseline | 0.22 | 0.23 |
| Add | 0.22 | 0.24 |
| Mult | 0.03 | 0.01 |
| **T**-add | 0.24 | 0.24 |
| **T**-mult | 0.37 | 0.36 |
| Van de Cruys et al. | 0.37 | |
| Humans | 0.62 | |

Table 3.14: Pearson correlation coefficients between human similarity judgements and similarity computed by our models on the Grefenstette and Sadrzadeh (2011) dataset.

### 3.4.3 Vecchi et al. dataset

The last dataset we consider to evaluate the compositional ability of our models was introduced by Vecchi et al. (2011) in order to determine if distributional semantic models are able to detect semantic deviance, that is expressions which are nonsensical. It consists of adjective-noun pairs that are unattested in the ukWaC corpus,[5] a 2009 dump of Wikipedia and the British National corpus. The authors classified these pairs as either acceptable or deviant. Examples of adjective-noun pairs from the dataset are given in Table 3.15

| **adjective** | **noun** | **class** |
|---|---|---|
| parliamentary | celebration | acceptable |
| parliamentary | mountain | deviant |
| printed | anecdote | acceptable |
| printed | avenue | deviant |
| innovative | biker | acceptable |
| innovative | centimetre | deviant |

Table 3.15: Examples of acceptable and deviant adjective-noun pairs from the Vecchi et al. (2011) dataset.

Let us now present how we evaluate our models on that dataset. For each pair of adjective and noun, we compute a score indicating how much the pair is nonsensical according to our model. We then compare the score distributions for pairs rated as deviant and pairs rated as acceptable, and in particular estimate how much those are different. Let $\mathbf{u}$ be the vector representing the noun and $\mathbf{v}$ the vector representing the adjective. The measure of acceptability $s$ we consider is:

$$s = \mathbf{v}^\top \mathbf{T}\, \mathbf{u}.$$

We then perform a Welch's $t$-test to determine if the two distributions have the same mean. For a tree model with 512 latent states, the $t$-statistics is equal to $-4.76$, corresponding to a $p$-value of $2.5e^{-6}$, while for a chain model with 512 latent states, the $t$-statistics is equal to $-4.88$, correponding to a $p$-value of $1.34e^{-6}$. We can thus conclude that both models are able to capture semantic deviance.

[5] http://wacky.sslmit.unibo.it/doku.php?id=corpora

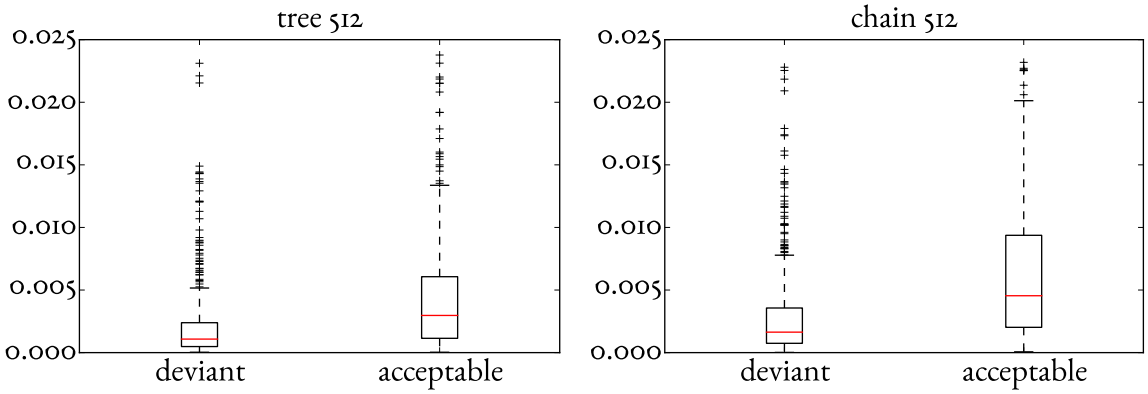



Figure 3.6: Distributions of acceptability score for deviant pairs and acceptable pairs, for a tree model (left) and a chain model (right).

# CHAPTER 4



# SEMI-SUPERVISED LEARNING

NOWADAYS, most state-of-the-art methods in natural language processing are based on supervised machine learning. Despite recent advances in statistical learning, those methods still suffer from limitations such as data sparsity and domain shift. It is thus common to learn a word representation on unlabeled data, in order to use it as features for the supervised task. In this chapter, we give some motivations for the use of semi-supervised learning methods in natural language processing and then evaluate our models when used to obtain word representations for two semi-supervised tasks: named entity recognition and supersense tagging.

## 4.1 Challenges of statistical methods for NLP

Since the mid nineties, statistical learning methods have encountered great success in computational linguistics but they still face significant challenges. First, labeling data for natural language processing is usually very expensive, because it requires expert knowledge in linguistics. Thus, most languages do not have as much labeled data as English. Second, a characteristic of natural languages is the fact that the distribution of words roughly follows a Zipf law, meaning that most words appear very infrequently. It is therefore very likely to encounter words at test time that were not seen in the training data. For example, more than ten percent of the tokens of the test section of the Penn treebank (Marcus et al., 1993) are not observed in the train section. This phenomenom is referred to as data sparsity. Finally, statistical methods are quite sensitive to domain shift: if a model is trained on labeled data distributed according to a given probability distribution, and tested on data distributed according to a different distribution, the performance usually degrades. This happens for example when a syntactic parser or a part-of-speech tagger is trained on labeled data from the Wall Street Journal and then tested on data such biomedical articles or electronic mails. For example, a syntactic dependency parser trained on Wall Street Journal data achieves 89.4% accuracy when tested on news articles, but only 78.8% when tested on a web corpus (Petrov and McDonald, 2012).

### 4.1.1 A solution: semi-supervised learning

On the one hand, labeling large amount of data for each new task, new domain or new language is unconceivable because it is too expensive. On the other, acquiring unlabeled data is almost free. It is thus natural to try to address the shortcomings of supervised methods we previously described by using a vast amount of available unlabeled data, through semi-supervised methods. One of the easiest way to do so is by learning a word representation using an unsupervised algorithm on the unlabeled data and to use this representation for the supervised task. This scheme has proven to be effective for various tasks such as named entity recognition (Freitag, 2004; Miller et al., 2004; Liang, 2005; Faruqui et al., 2010), part-of-speech tagging (Li and McCallum, 2005; Huang et al., 2011, 2013), syntactic chunking (Turian et al., 2010), Chinese word segmentation (Li and McCallum, 2005) or syntactic dependency parsing (Koo et al., 2008; Haffari et al., 2011; Tratz and Hovy, 2011). It was also successfully applied for transfer learning of multilingual structure by Täckström et al. (2012).

The most commonly used clustering method for semi-supervised learning is the one proposed by Brown et al. (1992), and known as Brown clustering. According to Turian et al. (2010), it is still a very competitive word representation for semi-supervised learning, and will thus be used as a baseline in the following. In recent work, Huang et al. (2011, 2013) also proposed to use a hidden Markov model to learn word representations for semi-supervised learning. The chain model we described in Chapter 2 is equivalent to their HMM model.

## 4.2 Experimental setting

Before moving to the extrinsic evaluation of our models on named entity recognition and supersense tagging, let us briefly describe the experimental settings we have considered and how to represent words given a hidden Markov model.

**Experimental settings.** We have decided to test our models in the two following settings:

- **Unlexicalized setting**: in this setting, words are not used as features,
- **Lexicalized setting**: in this setting words are used as features (in addition to a vectorial semantic class representation).

The unlexicalized setting allows us to evaluate how much information is captured by our models and emphazises the differences between the models. On the other hand, the lexicalized setting evaluates how useful the considered representations are for real problems, and in particular how redundant they are compared to the words themselves.

**Word representations.** Given a hidden Markov model (on chain or tree), there are several ways to use it in order to represent words. We decided to compare three representations that we now describe.

- `Viterbi`: this is the most commonly used solution in the literature. It consists in computing the most probable sequence of latent states by using Viterbi decoding, and to use this sequence as features. In that case, each word is represented by a integer. This representation is context dependent.
- `Posterior-Token`: the second possibility is to compute the posterior distribution of latent states associated to each token of the sentence and use this distribution as features. In that case, each word is represented by a distribution over the latent states. This representation is also context dependent.
- `Posterior-Type`: finally, the last possibility is, given a word type, to use the posterior distribution of latent states averaged over all the appearances of that word type in the training corpus. Each word is thus represented by a distribution over latent states, which is context independent.

The advantage of the `Posterior-Type` representation is that it does not need decoding or inference at test time. We also believe that this representation should be more robust, for example to errors in the dependency trees. The advantage of the `Viterbi` representation is the fact that it is sparse, and thus can lead to faster learning algorithms for the supervised step.

## 4.3 Named entity recognition

The first semi-supervised task on which we evaluate the different models we described in Chapter 2 is named entity recognition. We now briefly present this task before evaluating our models.

### 4.3.1 Presentation

Named entity recognition, sometimes abbreviated as NER, is an information extraction task whose goal is to detect and classify all the mentions of named entities in unstructured text. A named entity is anything that can be referred to with a proper name, such as people or places. We now give an example of a sentence where the named entities were annotated. This example is taken from the MUC7 dataset:

> *The seven-month re-examination of why [U.S.]$_{\texttt{LOC}}$ forces were caught off-guard by the Japanese attack was done at the request of Sen. [Strom Thurmond]$_{\texttt{PER}}$, R-[S.C.]$_{\texttt{LOC}}$, chairman of the [Senate Armed Services Committee]$_{\texttt{ORG}}$, and members of the [Kimmel]$_{\texttt{PER}}$ family.*

The types of named entities depends on the domain of application. For news articles, the goal is traditionally to detect mentions of people (`PER`), locations (`LOC`) and organizations (`ORG`). For abstracts of biomedical articles, the goal is to detect mentions of genes, proteins or cancers. Named entities can also be products, such as *iPhone 4* or *Xbox One*, movies, such as *The King's Speech* or *Taxi Driver* or novels, such as *Les Misérables* or *The Great Gatsby*.

Named entity recognition is a difficult task because of ambiguity. For example, the name *Charles de Gaulle* can refers to the former French president but also to one of the Paris' airport or to the French navy aircraft carrier, or to a subway stop in Paris. Thus, the same name can be classified, depending on the context, as a person, as a location or as a ship. Another source of ambiguity is the frequent use of metonymy to refer to an organization by using the location where it is hosted. For example, the name *White House* can either refer to the building, such as in

*The President of the United States lives in the White House.*

or can refer to the President and his administration, such as in

*The White House is confident on winning Hill support on Syria.*

Named entity recognition is often cast as a sequence labeling problem. Each word is labeled with a tag that captures both the type and the boundaries of the named entities, by using the `IOB` notation. For each type of entity, there are two tags, one starting with `B-` and one starting with `I-`. For example, for locations there is two tags: `B-LOC` and `I-LOC`. Tags starting with `B-` are used to label words that begin a named entity, tags starting with `I-` are used to label words that are inside a named entity while the tag `O` are used to label words that are not part of a named entity. For example, let us consider the sentence

*U.S. Air Force AWACS surveillance plane circled high over the Straits of Florida.*

Using the `IOB` notation, this sentence is labeled as:

| | |
|---|---|
| *U.S.* | `B-ORG` |
| *Air* | `I-ORG` |
| *Force* | `I-ORG` |
| *AWACS* | `O` |
| *surveillance* | `O` |
| *plane* | `O` |
| *circled* | `O` |
| *high* | `O` |
| *over* | `O` |
| *the* | `O` |
| *Straits* | `B-LOC` |
| *of* | `I-LOC` |
| *Florida* | `I-LOC` |
| . | `O` |

State of the art named entity recognition systems are based on conditional random fields, using various word-level features such as: the surface form, the corresponding lemma, the part-of-speech, the words that appear in a fixed size window around the word, the fact that the word appear in a list of known named entities (a.k.a. gazetteer) or features based on the shape of the word. Those shape features encode the fact that the word is capitalized, the fact that the word contains hyphen or numbers, etc.

### 4.3.2 Experiments

Following state of the art NER systems, we cast this problem as a sequence tagging problem, and thus use a linear conditional random field (CRF) (Sutton and McCallum, 2012; Lafferty et al., 2001) as our supervised classifier. Beside the representation derived from our models, the other features we use for both the lexicalized and unlexicalized setting are binary features indicating if the word is capitalized or not and the part-of-speech of the word. We perform experiments on the MUC7 dataset, evaluating our systems on the dryrun. The baseline for this task is assigning named entity classes to word sequences that occur in the training data.

#### Comparison of word representations

We start by comparing Brown clusters with the different word representations derived from hidden Markov models and described in section 4.2. We performed experiments using a tree model with 256 latent states and report the results for both the lexicalized and the unlexicalized setting in Table 4.1.

| | | Tree 256 | | |
|---|---|---|---|---|
| | | **PR** | **RE** | **F1** |
| Unlexicalized | Brown Clusters | 70.3 | 70.0 | 70.1 |
| | Viterbi | 73.2 | 69.6 | 71.3 |
| | Posterior-Type | 77.9 | 77.4 | 77.6 |
| | Posterior-Token | 81.5 | 76.8 | 79.1 |
| | Posterior-Both | **84.1** | **80.9** | **82.5** |
| Lexicalized | Brown Clusters | 85.8 | 79.0 | 82.2 |
| | Viterbi | 87.9 | 81.9 | 84.8 |
| | Posterior-Type | **89.3** | 80.9 | 84.9 |
| | Posterior-Token | **88.9** | **84.6** | **86.7** |
| | Posterior-Both | **89.0** | **84.4** | **86.7** |

Table 4.1: Comparison of different word representations for named entity recognition. PR stands for precision and RE stands for recall.

For the unlexicalized setting, the Brown clusters and the features obtained by using Viterbi decoding are outperformed by the other features. This is the case because it leads to a poor representation of words, compared to posterior methods: each word is represented only by an integer. The second interesting thing to observe is that representations which use the context, `Viterbi` and `Posterior-Token`, performs slightly better than their counterparts which do not use the context. This is especially true for the precision measure, meaning that some disambiguation is performed by context-dependent methods. Finally, combining the two posterior representations gives the best results overall, on both precision (84.1%) and recall (80.9%).

For the lexicalized setting, the story is a bit different. In that case, the features obtained

using Viterbi decoding perform essentially as well as the `Posterior-Type` representation. On the other hand, the `Viterbi` representation performs significantly better than Brown clusters. Similarly, the `Posterior-Token` representation performs significantly better than the `Posterior-Type` method. This means that in the lexicalized case, context is more important than the richness of the representation. Finally, combining both posterior representations gives similar results than `Posterior-Token`, attaining a precision of 89.0% and a recall of 84.4%.

Since combining both posterior representations produces the best results overall, we will use this representation in the following experiments.

### Influence of model size and initialization

In this section, we conduct experiments to study the influence of the number of latent states on performance, and if initializing hidden Markov models by using Brown clustering actually yields better results. We performed experiments using tree models with 128, 256, 512 and 1024 latent states. We report the F1 scores obtained in Figure 4.1.

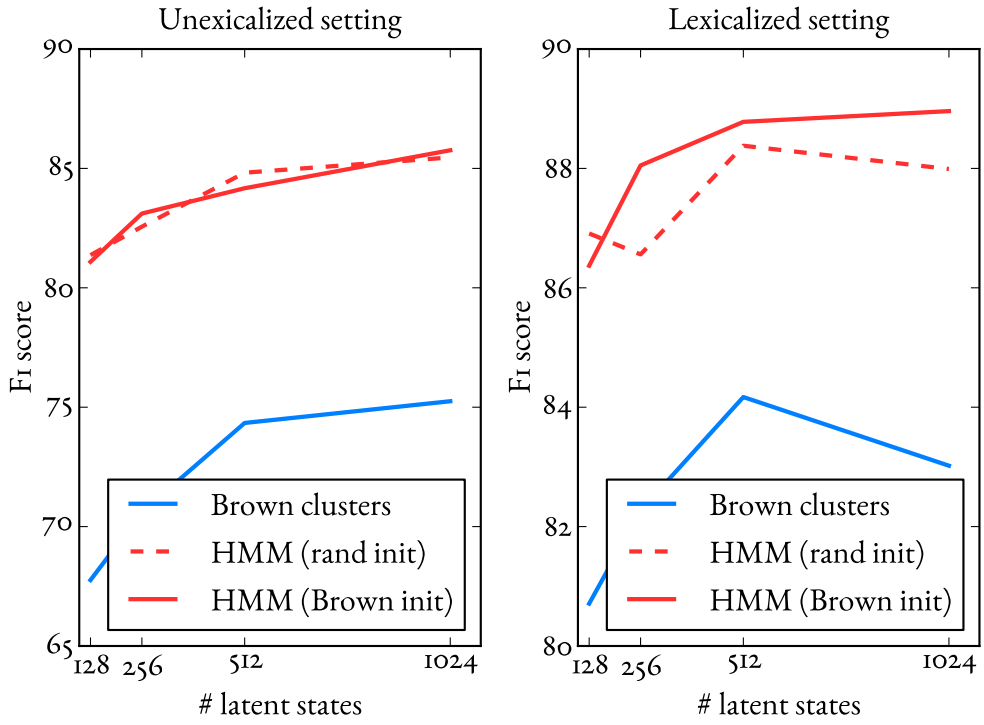



Figure 4.1: F1 scores obtained by our NER system for various number of latent classes. Blue curves are for tree Brown clusters, while red curves are for tree hidden Markov models. Dashed curves are for random initialization, while solid curves are for Brown initialization.

First, we observe that for Brown clustering and hidden Markov models, increasing the number of latent states improves the performance, for both the unlexicalized and the lexicalized setting. Second, we observe that for the unlexicalized setting, the initialization does not

play an important role, while it seems that it improves the performance for the lexicalized setting. In the following, we will thus use models that were initialized with Brown clusters.

**Influence of syntax**

We finally perform experiments to investigate the influence of syntax, and in particular to determine if using dependency tree models leads to better semantic classes. We perform experiments on models with 512 latent classes and report the results in Table 4.2.

| | | 512 | | |
|---|---|---|---|---|
| | | **PR** | **RE** | **F1** |
| Unlexicalized | Chain Brown | 73.5 | 73.2 | 73.3 |
| | Tree Brown | 74.9 | 73.7 | 74.3 |
| | Chain HMM | **85.8** | **82.7** | **84.0** |
| | Tree HMM | **86.2** | 82.0 | **84.1** |
| Lexicalized | Chain Brown | 86.6 | 78.2 | 82.2 |
| | Tree Brown | 88.4 | 81.0 | 84.6 |
| | Chain HMM | 90.4 | **85.9** | 88.1 |
| | Tree HMM | **91.6** | **86.0** | **88.7** |

Table 4.2: Precision, recall and F1 scores obtained by our NER system for different models: chain and tree Brown clustering, chain and tree hidden Markov models.

For Brown clustering, using dependency trees instead of linear chains improve the results for unlexicalized and lexicalized setting, by 1.0 and 2.4 points respectively. For hidden Markov models, the influence of syntax is less important. Indeed, for the unlexicalized setting, the results are comparable for tree and chain models while for the lexicalized setting, using dependency trees slightly improves (0.6 points) the results over a linear hidden Markov model.

## 4.4 Supersense tagging

The second semi-supervised task on which we evaluate the different models we described in Chapter 2 is supersense tagging. We now briefly present this task before evaluating our models.

### 4.4.1 Presentation

Word sense disambiguation (WSD) is the task of determining, for ambiguous words, which sense is used given the context. WordNet, which was introduced by Miller (1995), is the most used resource for word sense disambiguation. It is a lexical database of English that groups words into sets of synonyms (known as synsets) that express the same concept. There is more than 117,000 synsets in the current version of WordNet, and it is thus very hard to design a supervised disambiguation method that covers all the senses defined by WordNet. Moreover, it might be argued that for some applications, the synsets as defined by WordNet are too fine-grained. Thus, researchers on WSD have mainly concentrated on two subtasks:

- lexical sample WSD: in this task, the goal of a WSD system is to disambiguate a small number of chosen ambiguous words. Usually, there is only one word per sentence to disambiguate.
- coarse-grained all-words WSD: in this task, the goal of a WSD system is to disambiguate all the content words of a sentence. But the set of possible sense is greatly reduced compared to the one of WordNet, for example by clustering the synsets.

We refer the reader to Navigli (2009) for a survey on word sense disambiguation.

The task proposed by Ciaramita and Altun (2006) and known as supersense tagging is an extremely coarse-grained word sense disambiguation task. In WordNet, synsets are grouped into forty-five lexicographer classes, based on part-of-speech and logical grouping. Ciaramita and Altun (2006) thus proposed to use these classes, which they call supersenses, as coarse word senses (see Table 4.3 for the list of supersenses). They argue that due to the limited number of supersenses, it is possible to develop broad coverage disambiguation systems based on sequence labeling tools and thus to model dependencies between the supersenses of a given sentence, as opposed to disambiguating words independently. Moreover, since the supersense tagset comprises the classes `group`, `location` and `person`, supersense tagging can also be considered as a generalization of named entity recognition.

### 4.4.2 Experiments

We decided to evaluate our models on this task to determine the effect of homonymy. We cast supersense tagging as a classification problem and use word representation induced by our models as features for a support vector machine with the Hellinger kernel, defined by

$$K(\mathbf{p},\mathbf{q}) = \sum_{c=1}^{C} \sqrt{p_c q_c},$$

| Supersenses for nouns | Supersenses for verbs |
|---|---|
| Tops, act, animal, artifact, attribute, body, cognition, communication, event, feeling, food, group, location, motive, object, person, phenomenon, plant, possession, process, quantity, relation, shape, state, substance, time. | body, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social, stative, weather. |

Table 4.3: Supersense tagset.

where **p** and **q** are normalized word representations. We train and test the SVM classifier on the section A, B and C of the Brown corpus, tagged with Wordnet supersenses (SemCor).

### Comparison of word representations

We start by comparing the different word representations introduced in section 4.2. We report results for the unlexicalized and the lexicalized settings obtained by using a tree model with 128 latent classes in Table 4.4.

| | | tree 128 | | |
|---|---|---|---|---|
| | | **noun** | **verb** | **both** |
| Unlexicalized | Brown clusters | 52.6 | 43.4 | 49.8 |
| | Viterbi | 53.0 | 36.8 | 48.0 |
| | Posterior-Token | 59.7 | 43.9 | 54.9 |
| | Posterior-Type | 65.3 | **55.8** | 62.4 |
| | Posterior-Both | **68.0** | **55.8** | **64.3** |
| Lexicalized | Brown clusters | 70.7 | 56.9 | 66.5 |
| | Viterbi | 71.5 | 56.0 | 66.7 |
| | Posterior-Token | 74.5 | 58.5 | 69.6 |
| | Posterior-Type | 73.1 | **60.1** | 69.2 |
| | Posterior-Both | **76.1** | **60.2** | **71.3** |

Table 4.4: Classification accuracies of supersense tagging for various word representations. Results obtained using a tree model with 128 latent classes.

For the unlexicalized setting, we observe that the `Posterior-Type` representation outperforms all other representations by a large margin, even the `Posterior-Token` representation, which is quite surprising. Moreover, combining the two posterior representations does not improve the classification accuracy for verbs over the `Posterior-Type` representation. This means that contextualized representations does not help for verb disambiguation, which is quite disappointing.

For the lexicalized setting, the two posterior representations yield similar results and outperform the two other representations, `Brown Clusters` and `Viterbi`. For this setting, the `Posterior-Token` representation obtain better results than `Posterior-Type` on nouns, meaning that disambiguation helps. On the other hand, it gets worst results on verbs, and combining the two posterior representations still does not improve verb classification.

### Influence of model size and initialization

We now investigate the influence of the model size on the accuracy of supersense tagging. We consider tree models with 128, 256, 512 and 1024 latent classes. We report the classification accuracy as function of the model size in Figure 4.2.

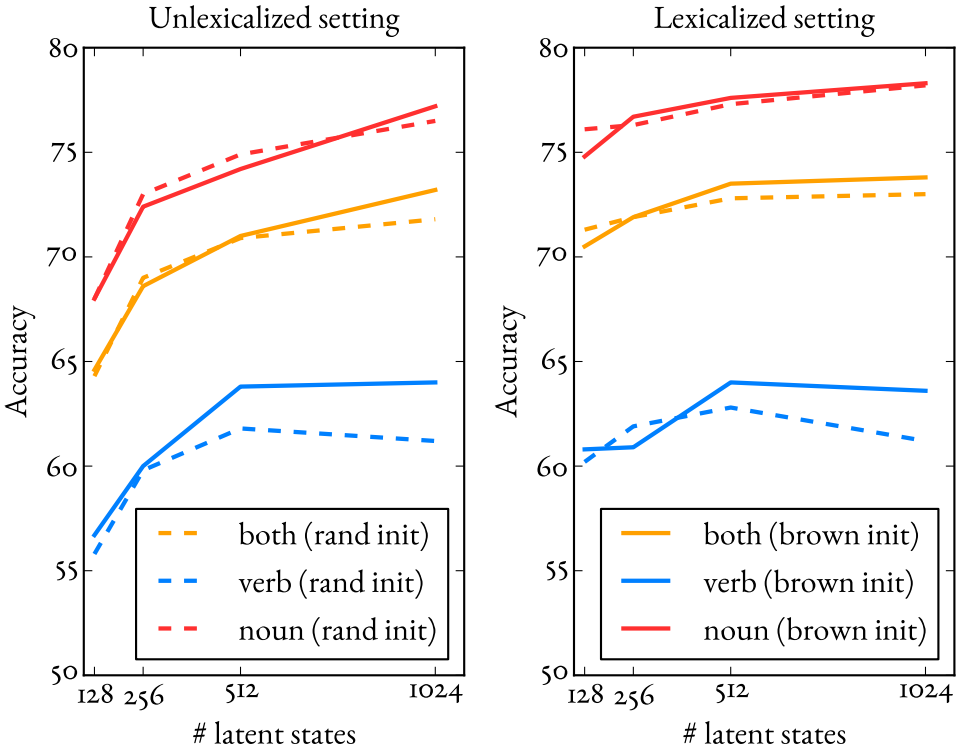



Figure 4.2: Classification accuracy for supersense tagging as a function of the number of latent classes for tree models. Accuracy for verb is in blue, for noun is in red and for both is in orange. Dashed lines are for random initialization, while plain lines are for Brown initialization.

We observe that for nouns, increasing the number of latent classes leads to a better classification accuracy. However, for verbs, the best classification accuracy is obtained for the model with 512 latent classes, for both the unlexicalized and the lexicalized setting and decreases for the model with 1024 classes. Moreover, we observe that random initialization is outperformed by Brown initialization for large models, and especially for verb classification accuracy.

### Influence of syntax

We finally perform experiments to determine the influence of syntax on semantic class induction for supersense tagging. We compare Brown clustering and hidden Markov models on trees and chains, with 512 latent classes and report the results in Table 4.5.

| | | noun | verb | both |
|---|---|---|---|---|
| Unlexicalized | Chain Brown | 59.8 | 51.0 | 57.1 |
| | Tree Brown | 63.4 | 51.1 | 59.6 |
| | Chain HMM | **74.1** | 62.8 | 70.6 |
| | Tree HMM | **74.2** | **63.8** | **71.0** |
| Lexicalized | Chain Brown | 71.5 | 58.4 | 67.5 |
| | Tree Brown | 73.6 | 58.4 | 69.0 |
| | Chain HMM | 76.8 | 63.0 | 72.5 |
| | Tree HMM | **77.6** | **64.4** | **73.6** |

Table 4.5: Classification accuracies for supersense tagging for different models: chain and tree Brown clustering, chain and tree hidden Markov models.

First, we observe that using the syntax improves the quality of Brown clusters for supersense tagging for both the unlexicalized (+2.5 point) and the lexicalized (+1.5 points) settings. For hidden Markov models, taking into account the syntax slightly helps in the unlexicalized setting (+0.4 point). Most of the improvement is due to better verbs classification. For the lexicalized setting, the improvement due to the suntax is more important, going from an accuracy of 72.5% to 73.6%.

It should be noted that these results might seem a bit contradictory with the one reported in Grave et al. (2013b). Indeed, in that article, it was reported that chain HMM representation was yielding better results for verbs than tree HMM, which is not the case here. This is because of the word representation that was used by Grave et al. (2013b): the `Posterior-Token` representation, while here we use the `Posterior-Both` one. We report in Table 4.6 the results for the unlexicalized setting with the `Posterior-Token` representation, in order to have results comparable with those of Grave et al. (2013b).

| | | noun | verb | both |
|---|---|---|---|---|
| Unlexicalized | Chain HMM | 64.7 | **55.4** | 61.9 |
| | Tree HMM | **66.8** | 53.1 | **62.6** |

Table 4.6: Classification accuracies for supersense tagging for chain and tree hidden Markov models, using the `Posterior-Token` representation.

We observe that the results reported in Table 4.6 are coherent with the ones reported in Grave et al. (2013b). We believe that the `Posterior-Token` representation does not work well for trees because verbs have a lot of children in dependency trees. Thus, the estimated posterior distribution of latent classes for verbs is more noisy than for chains. Moreover, since it depends on the context, it is more sensitive to errors in parse trees. Using the `Posterior-Type` representation, in addition to the `Posterior-Token` one, makes the classifier more robust.

## 4.5 Conclusion

In this section, we compared Brown clustering on both chains and trees and hidden Markov models, on both chains and trees. We also compared different word representations obtained using hidden Markov models, based on Viterbi decoding and posterior distributions over latent classes. First, we noted that continuous representations, based on posterior distributions, outperform discrete representations, such as Brown clustering or Viterbi decoding. Second, we observed that for most cases, using contextualized and non-contextualized representations together yields better results than using either one of them alone. Finally, using a dependency tree leads to better results than using a chain between the latent classes.

# Chapter 5

# Conclusion

In this first part, we introduced a simple and natural probabilistic model of sentences, based on dependency trees and hidden Markov models. We presented an efficient method to train such models on large quantities of unlabeled data. This allowed us to train models on datasets with tens of millions of sentences and hundreds millions of tokens in a day on a single core. We applied our model on various domains, such as news articles, biomedical abstracts or Wikipedia articles about musicians and diverse languages, such as English, French or Italian. We evaluated our model on tasks such as predicting human similarity judgements or word categorization. We also evaluated our model on two extrinsic tasks, named entity recognition ans supersense tagging.

We showed that for most of the considered tasks, using the syntactic dependency trees was helpful, compared to a regular hidden Markov chain model. However, one limitation of our model, which must be addressed in future work, is the fact that we do not use the labels of the dependency trees. As a consequence, it happens that agents and subjects of a given verb (or class of verbs) get mixed up in this same class. For example, we observed that the words *guitar* and *guitarist* sometimes appear in the same semantic class. It would also be interesting to train and test our model on languages which have more flexible word order than English, such as the Slavic languages. Indeed, using syntactic dependency trees should lead to larger improvements over a regular hidden Markov chain.

As far as we know, we are the first to propose to use hidden Markov models for distributional semantics. It is now possible to train such models on the large quantities of data used by researchers in distributional semantics thanks to the online expectation-maximization algorithm (Cappé and Moulines, 2009) and the approximate message passing algorithm (Pal et al., 2006; Grave et al., 2013b). In particular, the first one makes it possible to scale linearly in term of the number of sentences, while the second makes it possible to consider models with a large number of latent classes. We demonstrated that this approach seems to be competitive with state-of-the-art vector space models of semantics, even if our models were trained on a medium

sized specific corpus (300 millions tokens, only from news articles), leading to some dataset bias. It would thus be interesting to train our models on a more general and larger corpus, such as the WaCky corpus (3 billions tokens from the Web, Wikipedia and Brown corpus).

Most of the recent work in distributional semantics uses vector space models to represent words. One of the advantages of using a probabilistic model compared to previous approaches, is the fact that it makes it easier to interpret the obtained representations, and thus gives principled ways to use them for various tasks. For example, in section 3.4, the fact that we have a probabilistic model provides a very natural way to combine word representations in order to compare more complex linguistic units such as adjective-noun phrases or subject-verb-object triples. It is also very natural to obtain contextualized word representations.

Finally, since non-negative matrix factorization is closely related to probabilistic latent semantic analysis and latent Dirichlet allocation (Buntine, 2002; Gaussier and Goutte, 2005), and has been proposed as a dimensionality reduction technique for building vector space models of semantics based on bigrams (Van de Cruys, 2010), it would be interesting for future work to give a probabilistic interpretation of those models. Moreover, it would be interesting to study the similarities and differences between the hidden Markov models proposed in this thesis and the tensor factorization method proposed by Jenatton et al. (2012) and the one proposed by Van de Cruys et al. (2013). Indeed, a hidden Markov model can be viewed as a way to factorize the tensor of trigrams, and can also be learnt using tensor decompositions (Anandkumar et al., 2012).

# Part II

# Structured sparsity

# Chapter 6



# A brief introduction to statistical learning and variable selection

The goal of supervised statistical machine learning is to automatically learn to make predictions from data. For example, given some emails labeled as spam or non spam, a machine learning algorithm can learn to classify new mails as spam or non spam. Or given some pictures of cats and some pictures of dogs, a machine learning algorithm can learn to differentiate pictures of cats from pictures of dogs.

It is very easy to design an algorithm that will make no error on the training examples, but will behave very badly on new data. Thus, a good learning algorithm must find some kind of regularity in the training data, in order to generalize well to unseen data. A way to achieve this kind of regularities (among others), is to perform variable selection. The data that is fed into the learning algorithm is described by variables, and some of these variables might be irrelevant. In the example of email classification, an email can be described by the frequency of the words that appear in the text of the email, and thus each word frequency corresponds to a variable. But some words, such as *the* or *is* might be irrelevant to classify emails as spam or non spam. Thus, a good decision rule should not use these variables to make its prediction.

Another important motivation for variable selection is model interpretability. In some cases, understanding how the algorithm makes its predictions is essential. Let us consider an example from biomedical research, the problem of identifying which genes influence or are related to the development of certain diseases such as cancers. One way to do so is to collect a large number of gene expression levels from patients that suffer from cancer and from healthy people. This data is then used to learn a decison rule that predicts if a patient suffer from cancer or not, using variable selection. The variables that are selected by the algorithm are then more likely to be correlated with the development of cancer and research can be focused on those.

In the context of statistical learning, the concept of parsimony, also known as sparsity, is closely related to variable selection. Indeed, many learning algorithms reduce to estimating

a parameter vector that describes the decision rule. Performing variable selection can then be achieved by setting some of the coefficients of this vector to zero. In that case, the corresponding variables have no influence on the decision rule. In many cases, the problem of variable selection is equivalent to estimating a sparse parameter vector, *i.e.* a vector with many coefficients that are equal to zero.

As we just saw, the problem of variable selection has two main motivations:

- For some problems, it is believed that some variables are irrelevant, thus, an algorithm that do not use those variables will make fewer mistakes on new data. This assumption is essential for high dimensional problems, that is, problems where the number of variables is much larger than the number of observations ($n \ll p$).
- For some problems, understanding the decision rule is important. A model with fewer selected variables is easier to interpret.

In the following, we formalize statistical machine learning and we present how the variable selection problem can be formulated in that setting, through the use of sparsity inducing regularizers.

## 6.1 Empirical risk minimization

Let us now introduce more formally what is statistical learning and how it was formalize by Vapnik (1998). The goal of a supervised statistical learning algorithm is to automatically learn a function $f : \mathcal{X} \mapsto \mathcal{Y}$, that predicts a response variable $y \in \mathcal{Y}$, given an input variable $x \in \mathcal{X}$. For example, the set $\mathcal{X}$ can be the set of electronic mails, and the set $\mathcal{Y}$ can be equal to $\{-1, 1\}$, $y$ being equal to 1 if the corresponding email is spam and $-1$ otherwise. We suppose that there exists a joint probability distribution $P$ on the pairs $(x, y)$ and that we are given a loss function $\ell : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_+$. The real $\ell(\hat{y}, y)$ represents the loss suffered by the algorithm if it predicts $\hat{y}$ instead of $y$. For the email classification example, this loss can be equal to 1 if the predicted label is not equal to the true label and 0 otherwise. The goal of a statistical learning algorithm is then to find the function $f$ that has the smallest loss in expectation, meaning that it minimizes the Bayes risk $R$, defined as:

$$R(f) = \mathbb{E}_{(X,Y)\sim P}\Big[\ell(f(X), Y)\Big].$$

Unfortunately, the algorithm does not have direct access to the probability distribution $P$ of the data, and it is thus impossible to compute the Bayes risk $R$. But the algorithm has access to a set of $n$ examples, $(x_i, y_i)_{i\in\{1,\dots,n\}}$, called the training set, where each pair $(x_i, y_i)$ is independently drawn according to the probability distribution $P$. The Bayes risk of a function $f$ can then be estimated by the empirical risk $\hat{R}(f)$, defined as

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i).$$

Trying to find the function that minimizes the empirical risk $\hat{R}$ is not a good idea in general. Indeed, it is very easy to construct a function $f$ that have a zero empirical risk (if the minimum of $\ell$ is $0$), but will behave arbitrarily bad on unseen pairs $(x, y)$. This phenomenon is called overfitting. In order to generalize well to new data, it is thus necessary to restrict the set of functions on which we minimize the empirical risk, such as smooth functions, or functions that use a small subset of the variables (variable selection). We denote by $\mathscr{F}$ this set of functions, and call $\mathscr{F}$ the model. Empirical risk minimization then reduces to the following optimization problem:

$$\hat{f} \in \operatorname*{argmin}_{f \in \mathscr{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i). \tag{6.1}$$

When trying to solve a new problem using statistical machine learning, a practitioner has two decisions to make: the first one is the choice of the loss function $\ell$ and the second one is the choice of the set of prediction functions $\mathscr{F}$. We will now briefly discuss those.

### 6.1.1 Loss functions

The choice of the loss function depends on the kind of problem one is trying to solve, and in particular, depends on the type of the response variable $y$. We thus discuss the two main kind of supervised learning problems, namely classification and regression.

**Classification.** Classification problems arise when the set of response variables $\mathscr{Y}$ is discrete. An example of such problem is trying to classify emails into spams *v.s.* non-spams. In that case, the set $\mathscr{Y}$ is equal to $\{-1, 1\}$, $y$ being equal to 1 if the mail is a spam and $-1$ otherwise. This is an example of binary classification, since there are only two classes. Another example is digit recognition: in that case, the set $\mathscr{X}$ is the set of scanned images of digits, while the set $\mathscr{Y}$ is equal to the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. This is an example of multiclass classification. A natural loss function for classification is the $0-1$-loss, defined by

$$\ell_{0-1}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y, \\ 1 & \text{if } \hat{y} \neq y. \end{cases}$$

Unfortunately, this loss function leads to very hard optimization problems when used for empirical risk minimization, Eq. 6.1, and is thus hard to use in practice. For binary classification problems (where $\mathscr{Y} = \{-1, 1\}$), practitioners usually replace it by one of its convex surrogate, the hinge loss:

$$\ell_{\text{hinge}}(f(x), y) = \max(0, 1 - yf(x)),$$

or the logistic loss

$$\ell_{\text{logistic}}(f(x), y) = \log_2(1 + \exp(-yf(x))).$$

Multiclass classification can be reduced to a set of binary classifications problems, by using the one-*v.s.*-all or one-*v.s.*-one strategies.

**Regression.** When the response variable lives in a continuous space, such as the set of real numbers $\mathbb{R}$, the supervised learning problem is called a regression problem. Many such problems arise in economics for example, where researchers try to predict continuous variables such as consumption spending, labor demand or gross domestic product, or in image and sound processing. The most natural loss function for regression problems is the squared loss, defined by

$$\ell_{\text{square}}(f(x), y) = \frac{1}{2}(f(x) - y)^2.$$

In the remainder of this thesis, we will mainly focus on regression problems, even if most of our discussion and contributions are not restricted to this setting.

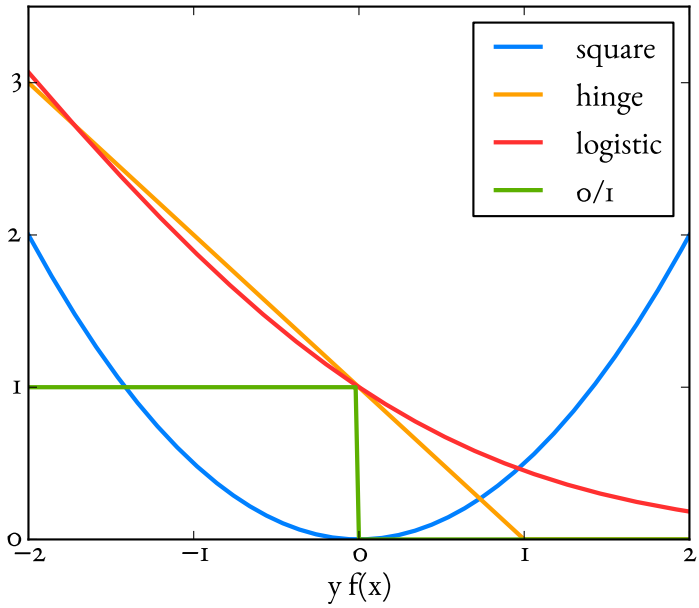



Figure 6.1: Various loss functions.

### 6.1.2 Linear models

One of the simplest, yet useful and powerful set $\mathcal{F}$ of prediction functions are the linear functions. If the input variable $x$ is a vector $\mathbf{x} \in \mathbb{R}^p$ living in a $p$-dimensional vector space, there exists an isomorphism between $\mathcal{F}$ and $\mathbb{R}^p$, and the application of $f$ to $\mathbf{x}$ can be written as the dot product between a vector $\mathbf{w}$ and $\mathbf{x}$:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}.$$

Linear models can be used for binary classification. In that case, the predicted label $\hat{y}$ is equal to $\text{sign}(\mathbf{w}^\top \mathbf{x})$. In the remainder of this thesis, we will mainly be interested in linear models. Another classical set of function classes are reproducing kernel Hilbert space, that we do not discuss here. We refer the interested reader to Shawe-Taylor and Cristianini (2004) for an introduction to reproducing kernel Hilbert spaces.

## 6.2 Approximation-estimation tradeoff

When choosing which model $\mathscr{F}$ to use, a practitioner wants to know how this choice influences performance on new data. In particular, it is interesting to know what is the performance of the learnt prediction function $\hat{f}$ is compared to the best possible prediction function $f^*$ called the Bayes estimator, and defined by

$$f^* = \operatorname*{argmin}_{f \in \mathscr{Y}^{\mathscr{X}}} \mathbb{E}_{(X,Y)\sim P}\Big[\ell(f(X),Y)\Big].$$

The classical measure of performance is the excess risk of $\hat{f}$, which is how much loss will be suffered by the function $\hat{f}$ compared to $f^*$, in expectation. The excess risk is thus equal to $R(\hat{f}) - R(f^*)$, and can be decomposed as

$$R(\hat{f}) - R(f^*) = \Big(R(\hat{f}) - R(\tilde{f})\Big) + \Big(R(\tilde{f}) - R(f^*)\Big), \tag{6.2}$$

where the function $\tilde{f}$ is the best possible function in our model $\mathscr{F}$ and is thus defined by[1]

$$\tilde{f} = \min_{f \in \mathscr{F}} \mathbb{E}_{(X,Y)\sim P}\Big[\ell(f(X),Y)\Big].$$

Let us now discuss the two terms that appear in the decomposition of the excess risk, defined in Eq. (6.2). Both of these terms are non negative. The first one, equal to

$$R(\hat{f}) - R(\tilde{f}) = \mathbb{E}_{(X,Y)\sim P}\Big[\ell(\hat{f}(X),Y)\Big] - \min_{f \in \mathscr{F}} \mathbb{E}_{(X,Y)\sim P}\Big[\ell(f(X),Y)\Big],$$

is called the estimation error. The estimation error depends on the training set $(x_i, y_i)_{i\in\{1,\dots,n\}}$, and in particular, will decrease when the size $n$ of the training set increases, because the empirical risk becomes a better estimator of the true risk. On the other hand, when the size of the model $\mathscr{F}$ increases, the estimation error also increases, because the search space is larger.

The second term, which is equal to

$$R(\tilde{f}) - R(f^*) = \min_{f \in \mathscr{F}} \mathbb{E}_{(X,Y)\sim P}\Big[\ell(f(X),Y)\Big] - \min_{f \in \mathscr{Y}^{\mathscr{X}}} \mathbb{E}_{(X,Y)\sim P}\Big[\ell(f(X),Y)\Big],$$

is called approximation error. The approximation error does not depend on the training set. It is a measure of how well the model $\mathscr{F}$ is able to approximate the best possible prediction function $f^*$. In particular, when the size of the model $\mathscr{F}$ increases, the approximation error decreases.

[1] We make the simplifying assumption that this infimum is attained, and thus a minimum.

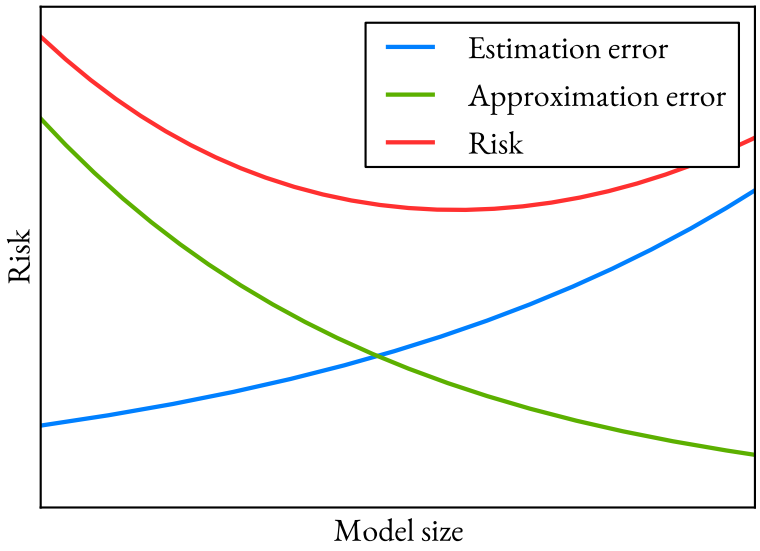



Figure 6.2: Illustration of the approximation-estimation tradeoff.

We thus see that there is a tradeoff in the choice of the model $\mathscr{F}$: the bigger the size of the model is, the bigger is the estimation error but the smaller is the approximation error. This is called the approximation-estimation tradeoff, since when the approximation error increases, the estimation error decreases. It thus suggests that we should consider models of various sizes. For example, we can impose a constraint on the smoothness of the functions we consider. For linear models, this corresponds to imposing a constraint on the norm of the parameter vector $\mathbf{w}$:

$$\mathscr{F}_B = \{\mathbf{w} \in \mathbb{R}^p \mid ||\mathbf{w}|| \leq B\}.$$

In the setting of variable selection, larger models correspond to decision functions with more selected variables. For linear models, this correspond to imposing a constraint on the number of non-zero coefficients. We will come back to that in section 6.4.3.

Another way to address the approximation-estimation tradeoff, instead of adding constraints to the empirical risk minimization problem, defined in Eq. (7.1), is to add a regularizer to the objective function. A regularizer will penalize prediction functions that are not smooth. In the case of linear models, regularizers are often norms of the parameter vector $\mathbf{w}$. We obtained the following optimization problem, called regularized empirical risk minimization:

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \lambda||\mathbf{w}||,$$

where $\lambda \in \mathbb{R}_+$ is called the regularization parameter. If the loss $\ell$ and the regularizer are both convex, the constrained and the regularized empirical risk minimization problems are

equivalent. It is often simpler to deal with the regularized version, from an optimization point of view. Constraining or regularizing the norm of the prediction function adds a free parameter $B$ or $\lambda$ that has to be chosen. This is called model selection, and we will briefly discuss how to address it in the following section.

## 6.3 Model selection

The goal of model selection is to pick the regularization parameter $\lambda$ or model $\mathcal{F}_B$, such that the corresponding learnt prediction function achieves the smallest possible risk. Usually, the regularization parameter $\lambda$ is chosen from a finite grid that discretized the set of non-negative real numbers $\mathbb{R}_+$. One of the easiest way to select the best $\lambda$ from this finite set is to train a model for each value of $\lambda$, and estimate its risk on a validation set containing the data points $(x_i, y_i)_{i\in\{n+1,\dots,m\}}$ that were not used for training. Since we assume that the data points are drawn independently from the probability distribution $P$, this means that the quantity

$$\frac{1}{m-n}\sum_{i=n+1}^{m} \ell(\hat{f}(x_i), y_i)$$

is a good estimator of the (true) risk of the function $\hat{f}$ learnt on the training set $(x_i, y_i)_{i\in\{1,\dots,n\}}$. This method is very simple to apply, but unfortunately, this means that the data points of the validation set are not used for training. Since more data points means smaller estimation error and smaller risk, all the available data should be used for learning the prediction function. We now introduce another model selection method that makes this possible.

### 6.3.1 $k$-fold cross validation

The idea of $k$-fold cross validation is to split the available data into a training set and a validation set, and repeat this operation multiple times. The estimated risk for each $\lambda$ is then averaged over the multiple runs, and the best $\lambda$ is then kept to learn a prediction function using all the available data. More formally, the training data is partitioned into $k$ subsets $(S_1, \dots, S_k)$. Then, for each subset $S_j$ of the partition, a prediction function is learnt on the $k-1$ remaining subsets, and its risk is estimated using $S_j$. For each $\lambda$, the estimated risk is averaged over the $k$ validation sets, and the best one is kept for learning a function on all the data. Usually, $k$ is equal to 5 or 10. If $k$ is equal to the size of the training set $n$, this means that each validation set correspond to one data point and in that case, it is called leave-one-out cross validation. The main drawback of $k$-fold cross validation is the fact that it is computationally expensive.

## 6.4 Some classical estimators for linear regression

In this section, we introduce some classical estimators for linear regression. We present different regularizers, based on different norms of the parameter vector, and we discuss how these

different regularizers affect the property of the estimated vector, and in particular its sparsity. We also discuss how the structure of the problem can be used to design better regularizers.

In the following, we will assume that the data is actually generated by a linear model. This means that for each pair $(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$, we have

$$y_i = \mathbf{x}_i^\top \mathbf{w}^* + \varepsilon_i,$$

where $\varepsilon_i$ is a zero-mean sub-Gaussian random variable, with variance $\sigma^2$. We further assume that the noise variables $\varepsilon_i$ are uncorrelated, meaning that for $i \neq j$, we have $\mathbb{E}[\varepsilon_i \varepsilon_j] = 0$.

### 6.4.1 Least squares regression

When combining linear models with the squared loss for regression, we obtain the well known ordinary least squares estimator. Let $(\mathbf{x}_i, y_i)_{i \in \{1,\dots,n\}}$ be the training set, where each $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. We note $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times p}$ and $\mathbf{y} = [y_1, \dots, y_n]^\top \in \mathbb{R}^n$. Then, the empirical risk minimization criterion is

$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{2n} \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}_i - y_i)^2.$$

It can be rewritten in the more concise and practical form:

$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2. \tag{6.3}$$

Finding the optimal vector $\mathbf{w}$ is done by taking the derivative of the empirical risk defined in Eq. (6.3) and setting it to zero. This yields the following linear system:

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}. \tag{6.4}$$

Thus, finding the optimal vector $\mathbf{w}$ for least squares regression problem is equivalent to solving a $p \times p$ linear system. In the particular case where the design matrix $\mathbf{X}$ is equal to the identity $\mathbf{I}$, the coefficients $\hat{w}_i$ of the solution of the least squares problem and the coefficients $y_i$ of the response vector verify the following very simple identity:

$$\hat{w}_i = y_i.$$

While this observation seems rather trivial for ordinary least squares, we will also make it for the following estimators since it will provide a good insight at the properties the corresponding regularizers.

Eq. (6.4) also means that the solution $\hat{\mathbf{w}}$ of the least squares problem is a linear function of the response variables $\mathbf{y}$. A famous result in statistics, the Gauss-Markov theorem, states that among the unbiased linear estimators, the least squares estimator is the one with the smallest variance. However, as discussed in section 6.2, it is often better to have a biased estimator, since slightly augmenting the bias can greatly decrease the variance, and thus the risk. We will consider such biased estimators by adding a regularizer to the ordinary least squares objective function.

### 6.4.2 Ridge regression

As we said before, when the training set is small and especially when the sample size $n$ is smaller than the dimension $p$ of the parameter space, minimizing the empirical risk leads to overfitting, even for linear models. It is thus beneficial to favor parameter vectors with small norms. The most natural norm to consider is the Euclidian $\ell_2$-norm, and regularizing by the squared $\ell_2$-norm is called Tikhonov regularization (Tikhonov, 1963). When used with the squared loss for linear regression, the corresponding estimator is called ridge regression (Hoerl and Kennard, 1970) and is defined by

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2.$$

As for ordinary least squares, introduced in the previous section, the solution of the optimization problem can be computed in closed form, by solving a linear system. Indeed, computing the derivative of the regularized empirical risk and setting it to zero yields the linear system:

$$\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)\mathbf{w} = \mathbf{X}^\top\mathbf{y}.$$

As for ordinary least squares, finding the optimal parameter vector for ridge regression is thus very efficient. We also observe that, as for ordinary least squares, the solution of the ridge regression problem is a linear function of the response variables $\mathbf{y}$, but the ridge regression is a biased estimator. Finally, in the case where $\mathbf{X} = \mathbf{I}$, the coefficients of the solution are given by:

$$\hat{w}_i = \frac{y_i}{1+\lambda}.$$

We thus observe that Tikhonov regularization shrinks the coefficients of the parameters vector toward zero, and that the larger coefficients are more shrunk that the small ones. This is not surprising, since the large coefficients are more penalized than the small ones. This also means Tikhonov regularization does not perform variable selection.

### 6.4.3 Lasso

We now discuss how to estimate sparse vectors in order to perform variable selection. In the framework of regularized empirical risk minimization, we should use a regularizer that favor vectors with many coefficient equal to zero, or in other words that penalizes more vectors with many non zero coefficients. The most natural regularizer is thus the number of non zero coefficients, sometimes known as the "$\ell_0$-norm", although it is not a norm, and which is equal to

$$||\mathbf{w}||_0 = \#\{w_i \neq 0\},$$

where # denotes the cardinal of the set. The corresponding estimator, known as best subset selection for the constrained version, is defined by

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_0. \tag{6.5}$$

Unfortunately, it is a very hard problem to solve: it was shown by Davis et al. (1997) that it is actually an NP-hard problem for some design matrices $\mathbf{X}$. This lead to the development of greedy approximate algorithms to solve this problem, such as matching pursuit (Mallat and Zhang, 1993b) or orthogonal matching pursuit (Pati et al., 1993; Davis et al., 1994). Another common solution is to replace the $\ell_0$-penalty by its convex surrogate, the $\ell_1$-norm. The corresponding estimator, known as the Lasso in the statistics community (Tibshirani, 1996) and as basis pursuit in the signal processing community (Chen and Donoho, 1994; Chen et al., 1998) is defined by

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1. \tag{6.6}$$

The solution of this optimization problem cannot be expressed in closed form, and contrary to the ridge regression, it is not a linear function of the response variables $\mathbf{y}$. We will present algorithms to compute the optimal solution of the Lasso in section 6.5. In the particular setting where $\mathbf{X} = \mathbf{I}$, the coefficients of the solution are equal to

$$\hat{w}_i = \operatorname{sign}(y_i)\Big[|y_i| - \lambda\Big]_+,$$

where $[a]_+$ is the positive part of the real number $a$. We observe that the $\ell_1$-regularization also shrinks the coefficients towards zero, but in a very different way than the Tikhonov regularization. In particular, the amount of shrinkage, $\lambda$, is the same for all coefficients, and if a coefficient is smaller than $\lambda$, it is then set to zero. This operator is thus called soft-thresholding operator (Donoho and Johnstone, 1994), and since some coefficients are set to zero, this means that the $\ell_1$-regularization, and in particular the Lasso, will perform variable selection.

When one is interested in variable selection, support recovery is another measure of performance besides the Bayes risk of the estimator (which measures how well the prediction function performs on new data). In that case, it is assumed that the true vector $\mathbf{w}^*$ is sparse, and the goal of a good algorithm is to recover the true sparsity pattern of the vector $\mathbf{w}^*$. Under certain conditions on the design matrix $\mathbf{X}$ and the support of the true vector $\mathbf{w}^*$, it can be shown that the Lasso will recover the true support with high probability. One of the tightest conditions is the irrepresentable condition (Zhao and Yu, 2006; Zou, 2006; Yuan and Lin, 2007; Wainwright, 2009), defined by

$$||\mathbf{X}_{S^C}^\top \mathbf{X}_S (\mathbf{X}_S^\top \mathbf{X}_S)^{-1}||_\infty \leq 1 - \delta,$$

where $S$ is the support of the true vector $\mathbf{w}^*$ and $||\ ||_\infty$ is the operator norm subordinated to the $\ell_\infty$-norm. This condition is also a necessary condition, meaning that if it is not met, there exists some vector $\mathbf{w}^*$ whose sparsity pattern cannot be recovered exactly.

Unfortunately, the design matrices $\mathbf{X}$ of many real problems do not verify those conditions, and often exhibit strong correlations between the different predictors (the columns of $\mathbf{X}$). In that particular case, the Lasso can be quite unstable: let us consider the extreme case where two predictors are equal. Then, the Lasso will select either of these two predictors indifferently, or

even both, since all those solutions are equivalent. Even if this instability is not a big issue for prediction, it is a big problem for model interpretation. We thus believe that a more stable estimator for variable selection should be considered for certain cases.

### 6.4.4 Elastic net

Unlike the Lasso, the ridge regression estimator tends to shrink coefficients of the parameter vector towards each other. In particular, in the extreme case where two predictors are equal, the corresponding coefficients of the solution of the ridge regression will also be equal. It is thus natural to consider a regularizer which is a convex combination of the $\ell_1$-norm for variable selection and the squared $\ell_2$-norm for the stability. This estimator was proposed by Zou and Hastie (2005) and is called the Elastic net:

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \lambda_1||\mathbf{w}||_1 + \frac{\lambda_2}{2}||\mathbf{w}||_2^2. \tag{6.7}$$

This optimization problem is equivalent to

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\tilde{\mathbf{y}}-\tilde{\mathbf{X}}\mathbf{w}||_2^2 + \lambda_1||\mathbf{w}||_1,$$

where the new design matrix $\tilde{\mathbf{X}}$ and the new response variables $\tilde{\mathbf{y}}$ are equal to

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ n\lambda_2\mathbf{I} \end{bmatrix} \qquad \text{and} \qquad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}.$$

We thus see that the effect of the elastic net estimator, compared to the Lasso, is to "decorrelate" the predictors of the design matrix $\mathbf{X}$. Moreover, this formulation, which is similar to a Lasso problem, allows to reuse the efficient optimization algorithms that were designed for the Lasso. In the setting where $\mathbf{X} = \mathbf{I}$, we have

$$\hat{w}_i = \frac{\operatorname{sign}(y_i)\Big[|y_i| - \lambda_1\Big]_+}{1+\lambda_2}.$$

We thus observe that elastic net combines both effects of the ridge regression and the Lasso. One of the drawback of this estimator is the added free parameter to chose.

### 6.4.5 Pairwise elastic net

One of the limitations of the elastic net is the fact that it ignores the correlation structure of the predictors. In particular, groups of strongly correlated predictors should be more regularized by the $\ell_2$-norm, while almost orthogonal predictors should be more regularized by the

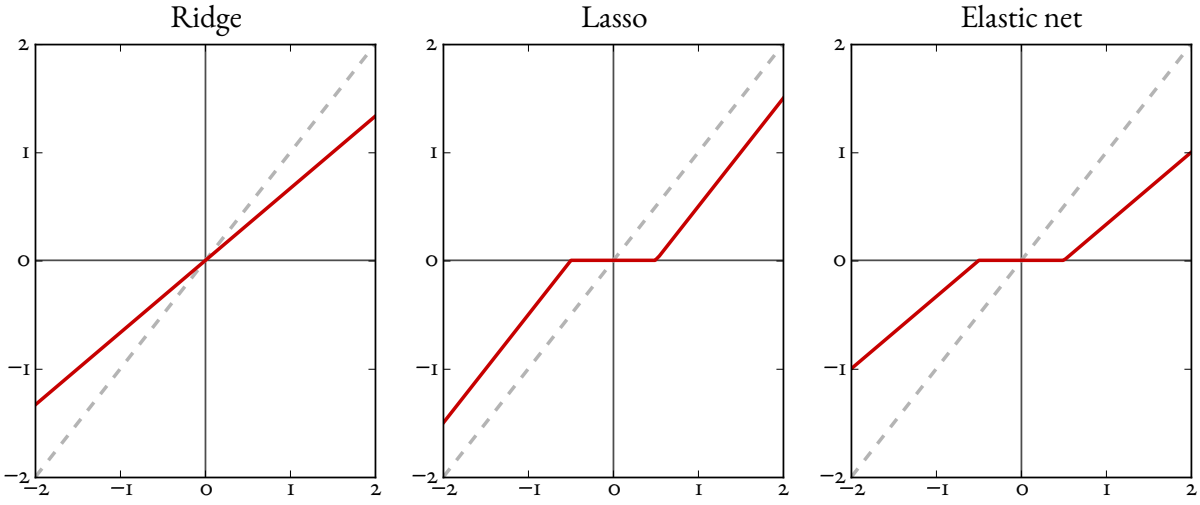



Figure 6.3: Thresholding operators corresponding to the ridge regression (left), the Lasso (center) and the elastic net (right).

$\ell_1$-norm. The pairwise elastic net (Lorbert et al., 2010) was proposed towards this goal: having a regularizer that tends to group highly correlated predictors, while performing variable selection for uncorrelated variables.

Before introducing the proposed regularizer, let us note that the squared $\ell_2$-norm and the squared $\ell_1$-norm of a vector $\mathbf{w} \in \mathbb{R}^p$, can be expressed using the vector $|\mathbf{w}|$, *i.e.*, the vector obtained by applying the elementwise absolute value function. Indeed, we have

$$||\mathbf{w}||_2^2 = |\mathbf{w}|^\top \mathbf{I}_p |\mathbf{w}|$$

and

$$||\mathbf{w}||_1^2 = |\mathbf{w}|^\top \mathbf{1}\mathbf{1}^\top |\mathbf{w}|.$$

The authors then proposed the pairwise elastic net estimator, which is defined by

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda\, |\mathbf{w}|^\top \mathbf{P} |\mathbf{w}|, \tag{6.8}$$

where $\mathbf{P} \in \mathbb{R}^{p\times p}$ is a symmetric, positive semi-definite matrix with nonnegative elements. These conditions on the matrix $\mathbf{P}$ ensure that the corresponding regularizer is convex. The authors proposed to use the following matrix:

$$\mathbf{P} = \mathbf{I}_p + \mathbf{1}\mathbf{1}^\top - \mathbf{X}^\top \mathbf{X}.$$

In that case, the pairwise elastic net regularizer is equal to $||\mathbf{w}||_2^2 + ||\mathbf{w}||_1^2 - |\mathbf{w}|^\top \mathbf{X}^\top \mathbf{X} |\mathbf{w}|$, which gives the Lasso estimator in the case of orthogonal predictors and the ridge regression in the case of equal predictors. The third term can thus be viewed as a tradeoff between the $\ell_1$ and

$\ell_2$-norms, based on the pairwise correlations. Unfortunately, this matrix is not always positive semi-definite, and the authors thus proposed to replace it by:

$$\mathbf{P} = \mathbf{I}_p + (1-\theta)\mathbf{1}\mathbf{1}^\top - (1-\theta)\mathbf{X}^\top\mathbf{X},$$

where $\theta$ is chosen to ensure positive semi-definiteness.

### 6.4.6 Group Lasso

Let us finish our tour of classical estimators for linear regression with the group Lasso regularizer. When one has more knowledge about the data, for example clusters of variables that should be selected together, the group Lasso (Yuan and Lin, 2006) is an efficient way to use this knowledge to improve quality of the estimated parameter vectors. Given a partition $(S_i)_{i\in\mathcal{G}}$ of the set of variables, the group Lasso penalty is the sum of the $\ell_2$-norms of the coefficient vectors $\mathbf{w}_{S_i}$ restricted to the groups $S_i$:

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \lambda \sum_{i\in\mathcal{G}} \sqrt{\#S_i}\, ||\mathbf{w}_{S_i}||_2. \tag{6.9}$$

The effect of this regularizer is to introduce sparsity at the group level: variables in a group are selected altogether and thus, the support of the solution is the union of a subset of the groups $(S_i)$. This regularizer was later extended to nested groups (Zhao et al., 2009) and to general overlapping groups (Jacob et al., 2009; Jenatton et al., 2011).

## 6.5 Optimization algorithms for the Lasso

In this section, we briefly review algorithms that were proposed to solve the Lasso optimization problem, defined by:

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^p_+} \frac{1}{2n}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1.$$

This problem is in fact a quadratic optimization problem. It can thus be solved using generic quadratic solver. However, because of the particular structure of the problem, faster algorithms can be derived, that we now present. We refer the interested reader to the survey on optimization for sparsity inducing norms by Bach et al. (2012).

### 6.5.1 Homotopy algorithm: LARS

The first algorithm we will present to compute the optimal solution of the Lasso is an algorithm called Least Angle Regression (LARS), and proposed by Efron et al. (2004). Under some assumptions on the design matrix $\mathbf{X}$, the solution of the Lasso is unique and we note $\hat{\mathbf{w}}(\lambda)$ its

solution associated to the regularization parameter $\lambda$. The function $\lambda \mapsto \hat{\mathbf{w}}(\lambda)$ is called the regularization path, and it was shown that for the Lasso, the regularization path is piecewise linear. The LARS algorithm exploits this property to compute the entire regularization path, by computing its kinks.

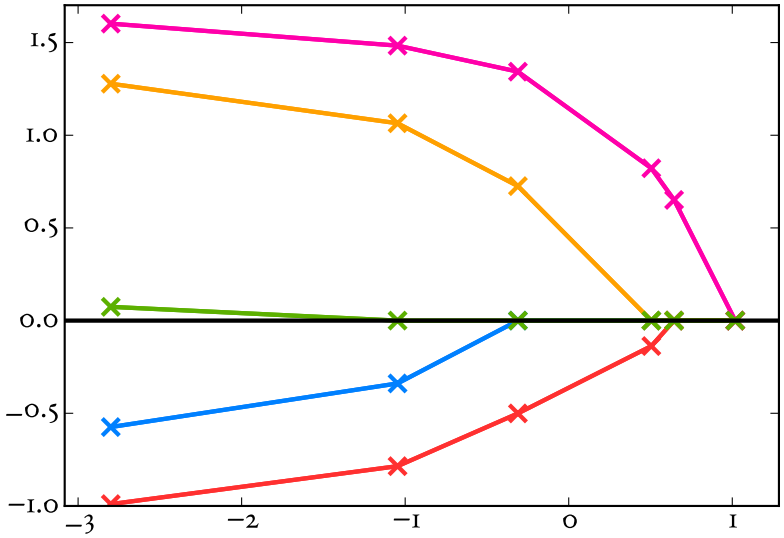



Figure 6.4: Example of regularization path of the Lasso. The coefficients of $\hat{\mathbf{w}}$ are plotted as a function of $\log(\lambda)$.

Let us start by stating the optimality conditions for the Lasso. A vector $\mathbf{w}$ is solution of the Lasso if and only if for all $i \in \{1, \ldots, p\}$:

$$\begin{aligned} \left|\mathbf{X}_i^\top (\mathbf{y} - \mathbf{X}\mathbf{w})\right| &\leq n\lambda && \text{if } w_i = 0, \\ \mathbf{X}_i^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) &= n\lambda \operatorname{sign}(w_i) && \text{if } w_i \neq 0. \end{aligned} \tag{6.10}$$

We denote by $S$ the support of $\mathbf{w}$, *i.e.* the set $\{i \ : \ w_i \neq 0\}$. We observe that as long as for the variables $i \in S^c$, the condition $\left|\mathbf{X}_i^\top (\mathbf{y} - \mathbf{X}\mathbf{w})\right| < n\lambda$ is verified and for the variables $i \in S$, $w_i$ does not change sign, we have a closed form solution for $\mathbf{w}$:

$$\mathbf{w}_S = (\mathbf{X}_S^\top \mathbf{X}_S)^{-1}(\mathbf{X}_S^\top \mathbf{y} - n\lambda \mathbf{s}_S), \tag{6.11}$$

$$\mathbf{w}_{S^c} = 0, \tag{6.12}$$

where $\mathbf{s}$ is defined by $s_i = \operatorname{sign}(w_i)$. The idea of the LARS algorithm is then to start from $\lambda = \frac{1}{n} \max_j |\mathbf{X}_j^\top \mathbf{y}|$, for which we have the trivial solution $\mathbf{w} = 0$, $S = \emptyset$ and $\mathbf{s} = 0$ and then to follow the regularization path $\hat{\mathbf{w}}(\lambda)$ using the closed form solution defined by Eq. (6.11), which is linear in $\lambda$, until the optimality conditions defined by Eq. (6.10) are violated. This happens either when

- for a variable $i \in S$, the corresponding coefficient $w_i$ hits zero. Then update $S$ by removing $i$ and set $s_i = 0$.
- for a variable $i \in S^c$, we have $\left|\mathbf{X}_i^\top (\mathbf{y} - \mathbf{X}\mathbf{w})\right| = n\lambda$. Then update $S$ by adding $i$ and set $s_i = \text{sign}(\mathbf{X}_i^\top (\mathbf{y} - \mathbf{X}\mathbf{w}))$.

In particular, the first variable to enter the set $S$ is $i = \text{argmax}_j |\mathbf{X}_j^\top \mathbf{y}|$. The $\lambda$ at which variables are added or removed from the support of $\mathbf{w}$, which are the kinks of the regularization path, can be easily computed in closed form from previous kink using Eq. (6.10) and Eq. (6.11).

### 6.5.2 Iteratively reweighted least squares

The term iteratively reweighted least squares (IRLS) was traditionally used to describe the Newton optimization method applied to the logistic regression, since each Newton step is then equivalent to solving a reweighted least squares problem. Now, IRLS designes a large family of algorithms (Grandvalet and Canu, 1999; Daubechies et al., 2010), where each step consists in solving a least squares problem, regularized by a reweighted $\ell_2$-norm. These kind of formulations arise when a non smooth regularizer is approximated by a smooth reweighted $\ell_2$-norm by using a variational formulation. In the case of the $\ell_1$-norm, we have the following variational formulation, sometimes known as the $\eta$-trick:

**Proposition 1.** *Let* $\mathbf{w} \in \mathbb{R}^p$. *The* $\ell_1$*-norm of* $\mathbf{w}$ *is equal to:*

$$||\mathbf{w}||_1 = \min_{\eta \in \mathbb{R}^p} \frac{1}{2} \sum_{i=1}^{p} \frac{|w_i|^2}{\eta_i} + \eta_i,$$

*and the infimum is attained for* $\eta_i = |w_i|$.

Using the variational formulation of the $\ell_1$-norm introduced in proposition 1, the Lasso problem can be reformulated as

$$\min_{\mathbf{w} \in \mathbb{R}^p} \min_{\eta \in \mathbb{R}_+^p} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2} \sum_{i=1}^{p} \frac{|w_i|^2}{\eta_i} + \eta_i.$$

This new optimization problem is jointly convex in $(\mathbf{w}, \eta)$, and the iteratively reweighted least squares algorithm consists in alternating the minimization over $\mathbf{w}$ and $\eta$. Some care must be taken, since the objective function is not continuous around $\eta$ with coefficients equal to zero, and thus, the alternating minimization algorithm is not convergent. In order to have a convergent algorithm, a smoothing term $\frac{\lambda\mu}{2} \sum_{i=1}^{p} \frac{1}{\eta_i}$ is added to the objective function[2]. In that case, minimization over $\eta$ still has a closed form solution, equal to

$$\eta_i = \sqrt{|w_i|^2 + \mu}$$

[2]The level sets of the objective function are then compact, ensuring the convergence of the algorithm.

while the minimization over $\mathbf{w}$ is equivalent to solving the following least squares problem regularized by a reweighted $\ell_2$-norm:

$$\min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2+\frac{\lambda}{2}\sum_{i=1}^{p}\frac{1}{\eta_i}|w_i|^2.$$

This method, which is quite simple to implement, has one main limitation: the obtained solution $\hat{\mathbf{w}}$ is not sparse, because of the added smoothing term.

### 6.5.3 Proximal methods

Proximal methods (Combettes and Pesquet, 2011) were designed to solve convex optimization problems where the objective function is the sum of a smooth function $f$ and a "simple" non-differentiable function $g$. What we call simple non differentiable functions are the functions for which we can easily compute their associated proximity operator, that we now define.

**Definition 3.** *Let $g$ be a convex function defined on $\mathbb{R}^p$. The proximity of operator of the function $g$, noted $prox_g$, is defined for all vectors $\mathbf{v}\in\mathbb{R}^p$ by*

$$prox_g(\mathbf{v})=\underset{\mathbf{u}\in\mathbb{R}^p}{\operatorname{argmin}}\frac{1}{2}||\mathbf{u}-\mathbf{v}||_2^2+g(\mathbf{u}).$$

The proximity operator of the $\ell_1$-regularizer is the soft-thresholding operator we have already introduced in section 6.4.3, and whose $i^{\text{th}}$ coefficient is equal to

$$\left(\text{prox}_{\lambda||\ ||_1}(\mathbf{v})\right)_i=\text{sign}(v_i)\Big[|v_i|-\lambda\Big]_+.$$

We will now present a simple proximal method to compute the optimal solution of the Lasso, introduced by Wright et al. (2009). For clarity, we will note $f(\mathbf{w})=\frac{1}{2n}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2$. Then at each iteration $t$, the smooth function $f$ is linearized around the current estimate $\mathbf{w}_t$:

$$\mathbf{w}_{t+1}=\underset{w\in\mathbb{R}^p}{\operatorname{argmin}} f(\mathbf{w}_t)+\nabla f(\mathbf{w}_t)^\top(\mathbf{w}-\mathbf{w}_t)+\lambda||\mathbf{w}||_1+\frac{L}{2}||\mathbf{w}_t-\mathbf{w}||_2^2.$$

The quadratic term is added to ensure that the next estimate $\mathbf{w}_{t+1}$ stays in the neighborhood of $\mathbf{w}_t$. This minimization problem is then equivalent to

$$\mathbf{w}_{t+1}=\underset{w\in\mathbb{R}^p}{\operatorname{argmin}}\frac{\lambda}{L}||\mathbf{w}||_1+\frac{1}{2}\left\|\left(\mathbf{w}_t-\frac{1}{L}\nabla f(\mathbf{w}_t)\right)-\mathbf{w}\right\|_2^2.$$

The solution of this optimization problem is exactly the proximity operator of the $\ell_1$-regularizer, rescaled by $\frac{1}{L}$ and applied to the vector $\mathbf{w}_t-\frac{1}{L}\nabla f(\mathbf{w}_t)$:

$$\mathbf{w}_{t+1}=\text{prox}_{\frac{\lambda}{L}||\ ||_1}\left(\mathbf{w}_t-\frac{1}{L}\nabla f(\mathbf{w}_t)\right).$$

We observe that if the non smooth term is equal to zero, its proximity operator is equal to the identity and the proximal method update is equal to the standard gradient descent update. For function $f$ with Lipschitz-continuous gradient (and thus for the Lasso), its convergence rate is $O\left(\frac{1}{t}\right)$. An accelerated version of this algorithm, called fast iterative soft thresholding algorithm (FISTA), was proposed by Beck and Teboulle (2009) and has a better convergence rate of $O\left(\frac{1}{t^2}\right)$.

# CHAPTER 7

# TRACE LASSO: A TRACE NORM REGULARIZATION FOR CORRELATED DESIGNS

USING THE $\ell_1$-norm to regularize the estimation of the parameter vector of a linear model leads to an unstable estimator when covariates are highly correlated. In this chapter, we introduce a new penalty function which takes into account the correlation of the design matrix to stabilize the estimation. This norm, called the trace Lasso, uses the trace norm of the selected covariates, which is a convex surrogate of their rank, as the criterion of model complexity. We analyze the properties of our norm, describe an optimization algorithm based on reweighted least-squares, and illustrate the behavior of this norm on synthetic data, showing that it is more adapted to strong correlations than competing methods such as the elastic net.

The material of this chapter is based on the following work:

E. Grave, G. Obozinski and F. Bach. Trace Lasso: a trace norm regularization for correlated designs. *Advances in Neural Information Processing Systems (NIPS)*, 2011.

## 7.1 Introduction

The concept of parsimony is central in many scientific domains. In the context of statistics, signal processing or machine learning, it takes the form of variable or feature selection problems, and is commonly used in two situations: first, to make the model or the prediction more interpretable or cheaper to use, i.e., even if the underlying problem does not admit sparse solutions, one looks for the best sparse approximation. Second, sparsity can also be used given prior knowledge that the model should be sparse. Many methods have been designed to learn sparse models, namely methods based on combinatorial optimization (Mallat and Zhang, 1993a; Zhang, 2008), Bayesian inference (Seeger, 2008) or convex optimization (Tibshirani, 1996; Chen et al., 1998).

In this chapter, we focus on the regularization by sparsity-inducing norms. The simplest example of such norms is the $\ell_1$-norm, leading to the Lasso, when used within a least-squares framework. In recent years, a large body of work has shown that the Lasso was performing optimally in high-dimensional low-correlation settings, both in terms of prediction (Bickel et al., 2009), estimation of parameters or estimation of supports (Zhao and Yu, 2006; Wainwright, 2009). However, most data exhibit strong correlations, with various correlation structures, such as clusters (i.e., close to block-diagonal covariance matrices) or sparse graphs, such as for example problems involving sequences (in which case, the covariance matrix is close to a Toeplitz matrix (Golub and Van Loan, 1996)). In these situations, the Lasso is known to have stability issues: although its predictive performance is not disastrous, the selected predictor may vary a lot. Typically, given two correlated variables, the Lasso will only select one of the two, at random, based on the fluctuation of the noise.

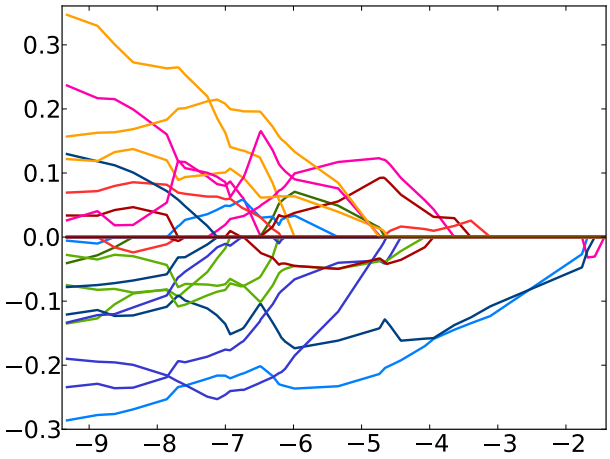



Figure 7.1: Example of regularization path of the Lasso, in the case of highly correlated predictors.

Several remedies have been proposed to this instability. First, the elastic net, proposed by Zou and Hastie (2005) adds a strongly convex penalty term (the squared $\ell_2$-norm) that will stabilize selection (typically, given two correlated variables, the elastic net will select the two variables). However, it is blind to the exact correlation structure, and while strong convexity is required for some variables, it is not for other variables. Another solution is to consider the group Lasso, proposed by Yuan and Lin (2006), which will divide the predictors into groups and penalize the sum of the $\ell_2$-norm of these groups. This is known to accomodate strong correlations within groups (Bach, 2008b); however it requires to know the groups in advance, which is not always possible. A third line of research has focused on sampling-based techniques (Bach, 2008a; Liu et al., 2010; Meinshausen and Bühlmann, 2010).

An ideal regularizer should thus be adapted to the design (like the group Lasso), but without requiring human intervention (like the elastic net); it should thus add strong convexity only

where needed, and not modifying variables where things behave correctly. In this chapter, we propose a new norm towards this end.

More precisely we make the following contributions:

- We propose in Section 7.2 a new norm based on the trace norm (a.k.a. nuclear norm) that interpolates between the $\ell_1$-norm and the $\ell_2$-norm depending on correlations.
- We show that there is a unique minimum when penalizing with this norm in Section 7.2.1.
- We provide optimization algorithms based on reweighted least-squares and alternating direction method of multipliers in Section 7.3.
- We study the second-order expansion around independence and relate to existing work on including correlations in Section 7.4.
- We perform synthetic experiments in Section 7.5, where we show that the trace Lasso outperforms existing norms in strong-correlation regimes.

**Notations.** Let $\mathbf{M} \in \mathbb{R}^{n \times p}$. The columns of $\mathbf{M}$ are noted using superscript, i.e., $\mathbf{M}^{(i)}$ denotes the $i$-th column, while the rows are noted using subscript, i.e., $\mathbf{M}_i$ denotes the $i$-th row. For $\mathbf{M} \in \mathbb{R}^{p \times p}$, $\operatorname{diag}(\mathbf{M}) \in \mathbb{R}^p$ is the diagonal of the matrix $\mathbf{M}$, while for $\mathbf{u} \in \mathbb{R}^p$, $\operatorname{Diag}(\mathbf{u}) \in \mathbb{R}^{p \times p}$ is the diagonal matrix whose diagonal elements are the $u_i$. Let $S$ be a subset of $\{1, ..., p\}$, then $\mathbf{u}_S$ is the vector $\mathbf{u}$ restricted to the support $S$, with 0 outside the support $S$. We denote by $\mathbb{S}_p$ the set of symmetric matrices of size $p$. We will use various matrix norms, here are the notations we use:

- $||\mathbf{M}||_*$ is the trace norm, *i.e.*, the sum of the singular values of the matrix $\mathbf{M}$,
- $||\mathbf{M}||_{op}$ is the operator norm, *i.e.*, the maximum singular value of the matrix $\mathbf{M}$,
- $||\mathbf{M}||_F$ is the Frobenius norm, *i.e.*, the $\ell_2$-norm of the singular values, which is also equal to $\sqrt{\operatorname{tr}(\mathbf{M}^\top \mathbf{M})}$,
- $||\mathbf{M}||_{2,1}$ is the sum of the $\ell_2$-norm of the columns of $\mathbf{M}$: $||\mathbf{M}||_{2,1} = \sum_{i=1}^{p} ||\mathbf{M}^{(i)}||_2$.

## 7.2 Definition and properties of the trace Lasso

We consider the problem of predicting $y \in \mathbb{R}$, given a vector $\mathbf{x} \in \mathbb{R}^p$, assuming a linear model

$$y = \mathbf{w}^\top \mathbf{x} + \varepsilon,$$

where $\varepsilon$ is (Gaussian) noise with mean 0 and variance $\sigma^2$. Given a training set $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)^\top \in \mathbb{R}^{n \times p}$ and $\mathbf{y} = (y_1, ..., y_n)^\top \in \mathbb{R}^n$, a widely used method to estimate the parameter vector $\mathbf{w}$ is the penalized empirical risk minimization

$$\hat{\mathbf{w}} \in \operatorname*{argmin}_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \mathbf{w}^\top \mathbf{x}_i) + \lambda f(\mathbf{w}), \tag{7.1}$$

where $\ell$ is a loss function used to measure the error we make by predicting $\mathbf{w}^\top \mathbf{x}_i$ instead of $y_i$, while $f$ is a regularization term used to penalize complex models. This second term helps avoiding overfitting, especially in the case where we have many more parameters than observation, *i.e.*, $n \ll p$.

### 7.2.1 The ridge, the Lasso and the trace Lasso

In this section, we show that Tikhonov regularization and the Lasso penalty can be viewed as norms of the matrix $\mathbf{X}\mathrm{Diag}(\mathbf{w})$. We then introduce a new norm involving this matrix.

The solution of empirical risk minimization penalized by the $\ell_1$-norm or $\ell_2$-norm is not invariant by rescaling the predictors $\mathbf{X}^{(i)}$, so it is common to normalize the predictors. When normalizing the predictors $\mathbf{X}^{(i)}$, and penalizing by Tikhonov regularization or by the Lasso, people are implicitly using a regularization term that depends on the data or design matrix $\mathbf{X}$. In fact, there is an equivalence between normalizing the predictors and not normalizing them, using the two following reweighted $\ell_2$ and $\ell_1$-norms instead of the Tikhonov regularization and the Lasso:

$$\|\mathbf{w}\|_2^2 = \sum_{i=1}^{p} \|\mathbf{X}^{(i)}\|_2^2 \, w_i^2 \qquad \text{and} \qquad \|\mathbf{w}\|_1 = \sum_{i=1}^{p} \|\mathbf{X}^{(i)}\|_2 \, |w_i|. \tag{7.2}$$

These two norms can be expressed using the matrix $\mathbf{X}\mathrm{Diag}(\mathbf{w})$:

$$\|\mathbf{w}\|_2 = \|\mathbf{X}\mathrm{Diag}(\mathbf{w})\|_F \qquad \text{and} \qquad \|\mathbf{w}\|_1 = \|\mathbf{X}\mathrm{Diag}(\mathbf{w})\|_{2,1},$$

and a natural question arises: are there other relevant choices of functions or matrix norms? A classical measure of the complexity of a model is the number of predictors used by this model, which is equal to the size of the support of $\mathbf{w}$. This penalty being non-convex, people use its convex relaxation, which is the $\ell_1$-norm, leading to the Lasso.

Here, we propose a different measure of complexity which can be shown to be more suited in model selection settings (Hastie et al., 2001): the dimension of the subspace spanned by the selected predictors. This is equal to the rank of the selected predictors, or also to the rank of the matrix $\mathbf{X}\mathrm{Diag}(\mathbf{w})$. As for the size of the support, this function is non-convex, and we propose to replace it by a convex surrogate, the *trace norm*, leading to the following penalty that we call "trace Lasso":

$$\Omega(\mathbf{w}) = \|\mathbf{X}\mathrm{Diag}(\mathbf{w})\|_*.$$

The trace Lasso has some interesting properties: if all the predictors are orthogonal, then, it is equal to the $\ell_1$-norm. Indeed, we have the decomposition:

$$\mathbf{X}\,\mathrm{Diag}(\mathbf{w}) = \sum_{i=1}^{p} \left( ||\mathbf{X}^{(i)}||_2 w_i \right) \frac{\mathbf{X}^{(i)}}{||\mathbf{X}^{(i)}||_2} \mathbf{e}_i^\top,$$

where $\mathbf{e}_i$ are the vectors of the canonical basis. Since the predictors are orthogonal and the $\mathbf{e}_i$ are orthogonal too, this gives the singular value decomposition of $\mathbf{X}\,\mathrm{Diag}(\mathbf{w})$ and we get

$$||\mathbf{X}\,\mathrm{Diag}(\mathbf{w})||_* = \sum_{i=1}^{p} ||\mathbf{X}^{(i)}||_2 |w_i| = ||\mathbf{X}\,\mathrm{Diag}(\mathbf{w})||_{2,1}.$$

On the other hand, if all the predictors are equal to $\mathbf{X}^{(1)}$, then

$$\mathbf{X}\,\mathrm{Diag}(\mathbf{w}) = \mathbf{X}^{(1)} \mathbf{w}^\top,$$

and we get $||\mathbf{X}\,\mathrm{Diag}(\mathbf{w})||_* = ||\mathbf{X}^{(1)}||_2 ||\mathbf{w}||_2 = ||\mathbf{X}\,\mathrm{Diag}(\mathbf{w})||_F$, which is equivalent to the Tikhonov regularization. Thus when two predictors are strongly correlated, our norm will behave like the Tikhonov regularization, while for almost uncorrelated predictors, it will behave like the Lasso.

Always having a unique minimum is an important property for a statistical estimator, as it is a first step towards stability. The trace Lasso, by adding strong convexity exactly in the direction of highly correlated covariates, always has a unique minimum, and is much more stable than the Lasso.

**Proposition 2.** *If the loss function $\ell$ is strongly convex with respect to its second argument, then the solution of the empirical risk minimization penalized by the trace Lasso, i.e., Eq. (7.1), is unique.*

The technical proof of this proposition is in appendix A.2, and consists in showing that in the flat directions of the loss function, the trace Lasso is strongly convex.

### 7.2.2 A new family of penalty functions

In this section, we introduce a new family of penalties, inspired by the trace Lasso, allowing us to write the $\ell_1$-norm, the $\ell_2$-norm and the newly introduced trace Lasso as special cases. In fact, we note that $||\mathrm{Diag}(\mathbf{w})||_* = ||\mathbf{w}||_1$ and $||p^{-1/2}\mathbf{1}^\top \mathrm{Diag}(\mathbf{w})||_* = ||\mathbf{w}^\top||_* = ||\mathbf{w}||_2$. In other words, we can express the $\ell_1$ and $\ell_2$-norms of $\mathbf{w}$ using the trace norm of a given matrix times the matrix $\mathrm{Diag}(\mathbf{w})$. A natural question to ask is: what happens when using a matrix $\mathbf{P}$ other than the identity or the line vector $p^{-1/2}\mathbf{1}^\top$, and what are good choices of such matrices? Therefore, we introduce the following family of penalty functions:

**Definition 4.** *Let $\mathbf{P} \in \mathbb{R}^{k \times p}$, all of its columns having unit norm. We introduce the norm $\Omega_{\mathbf{P}}$ as*

$$\Omega_{\mathbf{P}}(\mathbf{w}) = ||\mathbf{P}\,\mathrm{Diag}(\mathbf{w})||_*.$$

*Proof.* The positive homogeneity and triangle inequality are direct consequences of the linearity of $\mathbf{w} \mapsto \mathbf{P}\mathrm{Diag}(\mathbf{w})$ and the fact that $||\cdot||_*$ is a norm. Since all the columns of $\mathbf{P}$ are not equal to zero, we have

$$\mathbf{P}\mathrm{Diag}(\mathbf{w}) = 0 \Leftrightarrow \mathbf{w} = 0,$$

and so, $\Omega_{\mathbf{P}}$ separates points and is a norm. □

As stated before, the $\ell_1$ and $\ell_2$-norms are special cases of the family of norms we just introduced. Another important penalty that can be expressed as a special case is the group Lasso, with non-overlapping groups. Given a partition $(S_j)$ of the set $\{1, ..., p\}$, the group Lasso is defined by

$$||\mathbf{w}||_{GL} = \sum_{S_j} ||\mathbf{w}_{S_j}||_2.$$

We define the matrix $\mathbf{P}^{GL}$ by

$$\mathbf{P}^{GL}_{ij} = \begin{cases} 1/\sqrt{|S_k|} & \text{if } i \text{ and } j \text{ are in the same group } S_k, \\ 0 & \text{otherwise.} \end{cases}$$

Then,

$$\mathbf{P}^{GL}\mathrm{Diag}(\mathbf{w}) = \sum_{S_j} \frac{\mathbf{1}_{S_j}}{\sqrt{|S_j|}} \mathbf{w}_{S_j}^\top. \tag{7.3}$$

Using the fact that $(S_j)$ is a partition of $\{1, ..., p\}$, the vectors $\mathbf{1}_{S_j}$ are orthogonal and so are the vectors $\mathbf{w}_{S_j}$. Hence, after normalizing the vectors, Eq. (7.3) gives a singular value decomposition of $\mathbf{P}^{GL}\mathrm{Diag}(\mathbf{w})$ and so the group Lasso penalty can be expressed as a special case of our family of norms:

$$||\mathbf{P}^{GL}\mathrm{Diag}(\mathbf{w})||_* = \sum_{S_j} ||\mathbf{w}_{S_j}||_2 = ||\mathbf{w}||_{GL}.$$

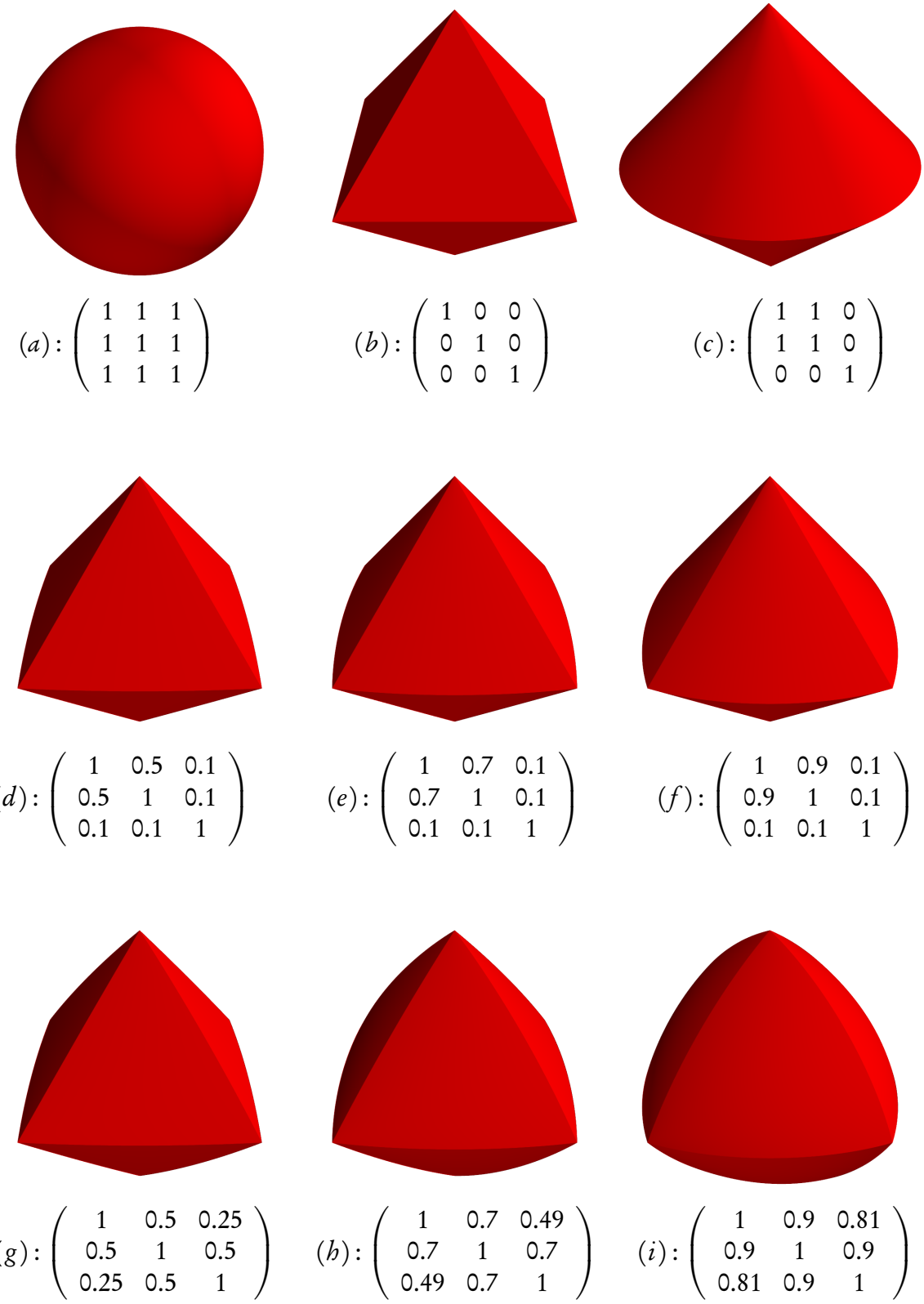
In the following proposition, we show that our norm only depends on the value of $\mathbf{P}^\top\mathbf{P}$. This is an important property for the trace Lasso, where $\mathbf{P} = \mathbf{X}$, since it underlies the fact that this penalty only depends on the correlation matrix $\mathbf{X}^\top\mathbf{X}$ of the covariates.

**Proposition 3.** *Let $\mathbf{P} \in \mathbb{R}^{k \times p}$, all of its columns having unit norm. We have*

$$\Omega_{\mathbf{P}}(\mathbf{w}) = ||(\mathbf{P}^\top\mathbf{P})^{1/2}\mathrm{Diag}(\mathbf{w})||_*.$$

We plot the unit ball of our norm for various values of $\mathbf{P}^\top\mathbf{P}$ (see figure (7.2)). We plot the unit balls of the special cases corresponding to the ridge regression (a), the Lasso (b) and the group Lasso (c). We also plot unit balls of our norm, for the following values of $\mathbf{P}^\top\mathbf{P}$:

$$\begin{pmatrix} 1 & \rho & 0.1 \\ \rho & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}$$



$$(a): \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad (b): \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (c): \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$(d): \begin{pmatrix} 1 & 0.5 & 0.1 \\ 0.5 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix} \quad (e): \begin{pmatrix} 1 & 0.7 & 0.1 \\ 0.7 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix} \quad (f): \begin{pmatrix} 1 & 0.9 & 0.1 \\ 0.9 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix}$$

$$(g): \begin{pmatrix} 1 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 1 \end{pmatrix} \quad (h): \begin{pmatrix} 1 & 0.7 & 0.49 \\ 0.7 & 1 & 0.7 \\ 0.49 & 0.7 & 1 \end{pmatrix} \quad (i): \begin{pmatrix} 1 & 0.9 & 0.81 \\ 0.9 & 1 & 0.9 \\ 0.81 & 0.9 & 1 \end{pmatrix}$$

Figure 7.2: Unit balls for various value of $\mathbf{P}^\top\mathbf{P}$.

for $\rho \in \{0.5, 0.7, 0.9\}$. The first case corresponds to a group of two correlated variables (d, e, f). The second case correspond to a Toeplitz design matrix (g, h, i).

We can lower bound and upper bound our norms by the $\ell_2$-norm and $\ell_1$-norm respectively. This shows that, as for the elastic net, our norms interpolate between the $\ell_1$-norm and the $\ell_2$-norm. But the main difference between the elastic net and our norms is the fact that our norms are *adaptive*, and require a single regularization parameter to tune. In particular for the trace Lasso, when two covariates are strongly correlated, it will be close to the $\ell_2$-norm, while when two covariates are almost uncorrelated, it will behave like the $\ell_1$-norm. This is a behavior close to the one of the pairwise elastic net (Lorbert et al., 2010).

**Proposition 4.** *Let $\mathbf{P} \in \mathbb{R}^{k \times p}$, all of its columns having unit norm. We have*

$$||\mathbf{w}||_2 \leq \Omega_{\mathbf{P}}(\mathbf{w}) \leq ||\mathbf{w}||_1.$$

### 7.2.3 Dual norm

The dual norm is an important quantity for both optimization and theoretical analysis of the estimator. Unfortunately, we are not able in general to obtain a closed form expression of the dual norm for the family of norms we just introduced. However we can obtain a bound, which is exact for some special cases:

**Proposition 5.** *The dual norm, defined by $\Omega^*_{\mathbf{P}}(\mathbf{u}) = \max_{\Omega_{\mathbf{P}}(\mathbf{v}) \leq 1} \mathbf{u}^\top \mathbf{v}$, can be bounded by:*

$$\Omega^*_{\mathbf{P}}(\mathbf{u}) \leq ||\mathbf{P}\,\mathrm{Diag}(\mathbf{u})||_{op}.$$

*Proof.* Using the fact that $\mathrm{diag}(\mathbf{P}^\top \mathbf{P}) = \mathbf{1}$, we have

$$\begin{aligned} \mathbf{u}^\top \mathbf{v} &= \mathrm{tr}\left(\mathrm{Diag}(\mathbf{u})\mathbf{P}^\top \mathbf{P}\,\mathrm{Diag}(\mathbf{v})\right) \\ &\leq ||\mathbf{P}\,\mathrm{Diag}(\mathbf{u})||_{op} ||\mathbf{P}\,\mathrm{Diag}(\mathbf{v})||_*, \end{aligned}$$

where the inequality comes from the fact that the operator norm $||\cdot||_{op}$ is the dual norm of the trace norm. The definition of the dual norm then gives the result. □

As a corollary, we can bound the dual norm by a constant times the $\ell_\infty$-norm:

$$\Omega^*_{\mathbf{P}}(\mathbf{u}) \leq ||\mathbf{P}\,\mathrm{Diag}(\mathbf{u})||_{op} \leq ||\mathbf{P}||_{op} ||\mathrm{Diag}(\mathbf{u})||_{op} = ||\mathbf{P}||_{op} ||\mathbf{u}||_\infty.$$

Using proposition (4), we also have the inequality $\Omega^*_{\mathbf{P}}(\mathbf{u}) \geq ||\mathbf{u}||_\infty$.

## 7.3 Optimization algorithms

In this section, we introduce two algorithms to estimatre the parameter vector $\mathbf{w}$. The first one belongs to the family of iteratively reweighted least square algorithm (IRLS), and thus requires that the square loss is used. The second algorithm that we propose is based on the alternating directions method of multipliers (ADMM), and only requires the loss to be convex and differentiable. In the following, we will present both algorithms for the trace Lasso, but it is straighforward to extend them to the family of norms indexed by a matrix $\mathbf{P}$.

We recall that the problem we consider is

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{X}\mathrm{Diag}(\mathbf{w})||_*.$$

We could optimize this cost function by subgradient descent, but this is quite inefficient: the rate of convergence of subgradient descent is quite slow and computing the subgradient of the trace Lasso is expensive. Indeed, the following proposition implies that at each step of the subgradient descent for the trace Lasso, the singular value decomposition of an $n \times p$ matrix has to be computed:

**Proposition 6.** *Let* $\mathbf{U}\mathrm{Diag}(\mathbf{s})\mathbf{V}^\top$ *be the singular value decomposition of* $\mathbf{X}\mathrm{Diag}(\mathbf{w})$. *Then, the subgradient of the trace Lasso regularization is given by*

$$\partial\Omega(\mathbf{w}) = \left\{\mathrm{diag}\left(\mathbf{X}^\top(\mathbf{U}\mathbf{V}^\top + \mathbf{M})\right) \mid ||\mathbf{M}||_2 \leq 1, \mathbf{U}^\top\mathbf{M} = 0 \text{ and } \mathbf{M}\mathbf{V} = 0\right\}.$$

### 7.3.1 Iteratively reweighted least squares

The first optimization algorithm we consider belongs to the family of iteratively reweighted least-squares methods. First, we need to introduce a well-known variational formulation for the trace norm (Argyriou et al., 2007):

**Proposition 7.** *Let* $\mathbf{M} \in \mathbb{R}^{n\times p}$. *The trace norm of* $\mathbf{M}$ *is equal to:*

$$||\mathbf{M}||_* = \frac{1}{2}\inf_{\mathbf{S}\succeq 0} \mathrm{tr}\left(\mathbf{M}^\top\mathbf{S}^{-1}\mathbf{M}\right) + \mathrm{tr}(\mathbf{S}),$$

*and the infimum is attained for* $\mathbf{S} = \left(\mathbf{M}\mathbf{M}^\top\right)^{1/2}$.

Using this proposition, we can reformulate the previous optimization problem as

$$\min_{\mathbf{w}} \inf_{S\succeq 0} \frac{1}{2}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}\mathbf{w}^\top \mathrm{Diag}\left(\mathrm{diag}(\mathbf{X}^\top\mathbf{S}^{-1}\mathbf{X})\right)\mathbf{w} + \frac{\lambda}{2}\mathrm{tr}(\mathbf{S}).$$

This problem is jointly convex in $(\mathbf{w}, \mathbf{S})$ (Boyd and Vandenberghe, 2004). In order to optimize this objective function by alternating the minimization over $\mathbf{w}$ and $\mathbf{S}$, we need to

add a term $\frac{\lambda\mu_i}{2}\operatorname{tr}(\mathbf{S}^{-1})$. Otherwise, the infimum over $\mathbf{S}$ could be attained at a non invertible $\mathbf{S}$, leading to a non convergent algorithm. The infimum over $\mathbf{S}$ is then attained for $\mathbf{S} = \left(\mathbf{X}\operatorname{Diag}(\mathbf{w})^2\mathbf{X}^\top + \mu_i\mathbf{I}\right)^{1/2}$.

Optimizing over $\mathbf{w}$ is a least-squares problem penalized by a reweighted $\ell_2$-norm equal to $\mathbf{w}^\top\mathbf{D}\mathbf{w}$, where $\mathbf{D} = \operatorname{Diag}\left(\operatorname{diag}(\mathbf{X}^\top\mathbf{S}^{-1}\mathbf{X})\right)$. It is equivalent to solving the linear system

$$(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{D})\mathbf{w} = \mathbf{X}^\top\mathbf{y}. \tag{7.4}$$

Due to the structure of the matrix $\mathbf{A} = \mathbf{X}^\top\mathbf{X} + \lambda\mathbf{D}$, a rank deficient matrix plus a diagonal matrix, computing the product of the matrix $\mathbf{A}$ with a vector is quite cheap. Thus, the best choice for solving the linear system 7.4 is to use an iterative method only using matrix vector multiplication, such as the conjugate gradient algorithm (Golub and Van Loan, 1996). We now summarize the algorithm:

---

IRLS ALGORITHM FOR ESTIMATING $\mathbf{w}$

---

**Input:** the design matrix $\mathbf{X}$, the initial guess $\mathbf{w}^0$, number of iteration $N$, sequence $\mu_i$.
**For** $i = 1...N$:

- Compute the eigenvalue decomposition $\mathbf{U}\operatorname{Diag}(s_k)\mathbf{U}^\top$ of $\mathbf{X}\operatorname{Diag}(\mathbf{w}^{i-1})^2\mathbf{X}^\top$.
- Set $\mathbf{D} = \operatorname{Diag}(\operatorname{diag}(\mathbf{X}^\top\mathbf{S}^{-1}\mathbf{X}))$, where $\mathbf{S}^{-1} = \mathbf{U}\operatorname{Diag}(1/\sqrt{s_k + \mu_i})\mathbf{U}^\top$.
- Set $\mathbf{w}^i$ by solving the system $(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{D})\mathbf{w} = \mathbf{X}^\top\mathbf{y}$.

---

For the sequence $\mu_i$, we use a decreasing sequence converging to ten times the machine precision.

**Complexity of the IRLS algorithm.**

1. The first step has a complexity of $\mathcal{O}\left(n^2(n+p)\right)$: computing the matrix product has a complexity of $pn^2$ and computing the eigenvalue decomposition of an $n \times n$ matrix has a complexity of $n^3$.
2. Computing the product $\mathbf{X}^\top\mathbf{U}$ has a compexity of $\mathcal{O}(n^2 p)$, while the other operations have a complexity of $\mathcal{O}(np)$.
3. The complexity of computing a matrix-vector multiplication $(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{D})\mathbf{w}$ is $\mathcal{O}(np)$. Moreover, using the conjugate gradient algorithm to solve the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} = \mathbf{I} + \mathbf{E}$ and $\mathbf{E}$ is a positive definite matrix of rank $r$ takes at most $r + 1$ iterations (see

Golub and Van Loan, 1996, Theorem 10.2.5). Hence, preconditioning by $\mathbf{D}^{-1/2}$ ensures that the complexity of this step is at most $\mathcal{O}(n^2 p)$. The use of warm restart can even speed up this step.

The complexity of one iteration of our IRLS algorithm is thus $\mathcal{O}(n^2 p)$, if $p \geq n$.

### 7.3.2 Alternating direction method of multipliers

We now introduce a second optimization technique for the trace Lasso, based on the alternating direction method of multipliers (See Boyd et al., 2011, for an introduction). Even if this method can be used with any convex and differentiable loss, we will present it using the square loss. We first introduce the dummy variable $\mathbf{M} \in \mathbb{R}^{n \times p}$ and obtain the following equivalent optimization problem:

$$\begin{cases} \min\limits_{\mathbf{w},\mathbf{M}} & \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{M}||_* \\ \text{such that} & \mathbf{M} = \mathbf{X}\mathrm{Diag}(\mathbf{w}). \end{cases}$$

The corresponding augmented Lagrangian is

$$\mathcal{L}_\rho(\mathbf{w},\, \mathbf{M},\, \Lambda) = \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{M}||_* + \mathrm{tr}(\Lambda^\top(\mathbf{X}\mathrm{Diag}(\mathbf{w}) - \mathbf{M})) + \frac{\rho}{2}||\mathbf{X}\mathrm{Diag}(\mathbf{w}) - \mathbf{M}||_F^2.$$

Then, the alternating direction method of multipliers, which is an extension of the augmented Lagrangian consists of the iterations:

$$\begin{aligned} \mathbf{w}^{(k+1)} &= \operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w},\, \mathbf{M}^{(k)},\, \Lambda^{(k)}), \\ \mathbf{M}^{(k+1)} &= \operatorname*{argmin}_{\mathbf{M}} \mathcal{L}(\mathbf{w}^{(k+1)},\, \mathbf{M},\, \Lambda^{(k)}), \\ \Lambda^{(k+1)} &= \Lambda^{(k)} + \rho\left(\mathbf{X}\mathrm{Diag}(\mathbf{w}^{(k+1)}) - \mathbf{M}^{(k+1)}\right). \end{aligned}$$

**Optimization w.r.t. w.** The optimization problem with respect to $\mathbf{w}$ is equivalent to

$$\min_{\mathbf{w}} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \mathrm{tr}(\Lambda^\top\mathbf{X}\mathrm{Diag}(\mathbf{w})) + \frac{\rho}{2}\mathrm{tr}(\mathrm{Diag}(\mathbf{w})\mathbf{X}^\top\mathbf{X}\mathrm{Diag}(\mathbf{w}) - 2\mathbf{M}^\top\mathbf{X}\mathrm{Diag}(\mathbf{w})),$$

which is equivalent to

$$\min_{\mathbf{w}} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \frac{\rho}{2}\mathbf{w}^\top\mathbf{w} + \mathrm{diag}\left((\Lambda - \rho\mathbf{M})^\top\mathbf{X}\right)^\top\mathbf{w}.$$

Thus, computing the optimal $\mathbf{w}$ is easily done by solving the linear system:

$$\left(\mathbf{X}^\top\mathbf{X} + \rho\mathbf{I}\right)\mathbf{w} = \mathbf{X}^\top\mathbf{y} - \mathrm{diag}\left((\Lambda - \rho\mathbf{M})^\top\mathbf{X}\right).$$

**Optimization w.r.t. M.** The optimization problem with respect to $\mathbf{M}$ is equivalent to

$$\min_{\mathbf{M}} \lambda||\mathbf{M}||_* - \text{tr}(\Lambda^\top \mathbf{M}) + \frac{\rho}{2}||\mathbf{X}\text{Diag}(\mathbf{w}) - \mathbf{M}||_F^2,$$

which is equivalent to

$$\min_{\mathbf{M}} \lambda||\mathbf{M}||_* + \frac{\rho}{2}||(\mathbf{X}\text{Diag}(\mathbf{w}) + \frac{1}{\rho}\Lambda) - \mathbf{M}||_F^2.$$

This problem is the proximal operator of the trace norm, which has a closed form solution requiring to compute a singular value decomposition.

**Complexity of the ADMM algorithm.**

1. Solving the linear system to minimize the Lagrangian with respect to $\mathbf{w}$ has a complexity of $\mathcal{O}(n^2 p)$. Indeed, this linear system has the same structure as the linear system of the IRLS algorithm, and thus can be solved in the same way.

2. Computing the optimal matrix $\mathbf{M}$ is equivalent to taking the proximal operator of the trace norm of an $n \times p$ matrix. This proximal operator is computed by taking the SVD of that matrix, whose complexity is $\mathcal{O}(n^2 p)$.

3. The gradient step with respect to $\Lambda$ has a complexity of $\mathcal{O}(np)$.

Thus, like the IRLS algorithm, one step of the ADMM algorithm has a complexity of $\mathcal{O}(n^2 p)$, if $p \geq n$.

### 7.3.3 Choice of $\lambda$

We now give a method to choose the regularization path. In fact, we know that the vector $0$ is solution if and only if $\lambda \geq \Omega^*(\mathbf{X}^\top \mathbf{y})$ (Bach et al., 2012). Thus, we need to start the path at $\lambda = \Omega^*(\mathbf{X}^\top \mathbf{y})$, corresponding to the empty solution $0$, and then decrease $\lambda$. Using the inequalities on the dual norm we obtained in the previous section, we get

$$||\mathbf{X}^\top \mathbf{y}||_\infty \leq \Omega^*(\mathbf{X}^\top \mathbf{y}) \leq ||\mathbf{X}||_{op}||\mathbf{X}^\top \mathbf{y}||_\infty.$$

Therefore, starting the path at $\lambda = ||\mathbf{X}||_{op}||\mathbf{X}^\top \mathbf{y}||_\infty$ is a good choice.

## 7.4 Approximation around the Lasso

In this section, we compute the second order approximation of our norm around the special case corresponding to the Lasso. We recall that when $\mathbf{P} = \mathbf{I} \in \mathbb{R}^{p \times p}$, our norm is equal to the

$\ell_1$-norm. We add a small perturbation $\Delta \in \mathbb{S}_p$ to the identity matrix, and using proposition 8, we obtain the following second order approximation:

$$||(\mathbf{I}+\Delta)\mathrm{Diag}(\mathbf{w})||_* = ||\mathbf{w}||_1 + \mathrm{diag}(\Delta)^\top|\mathbf{w}| + \sum_{|w_i|>0}\sum_{|w_j|>0}\frac{(\Delta_{ji}|w_i| - \Delta_{ij}|w_j|)^2}{4(|w_i|+|w_j|)} + \sum_{|w_i|=0}\sum_{|w_j|>0}\frac{(\Delta_{ij}|w_j|)^2}{2|w_j|} + o(||\Delta||^2).$$

We can rewrite this approximation as

$$||(\mathbf{I}+\Delta)\mathrm{Diag}(\mathbf{w})||_* = ||\mathbf{w}||_1 + \mathrm{diag}(\Delta)^\top|\mathbf{w}| + \sum_{i,j}\frac{\Delta_{ij}^2(|w_i|-|w_j|)^2}{4(|w_i|+|w_j|)} + o(||\Delta||^2),$$

using a slight abuse of notation, considering that the last term is equal to 0 when $w_i = w_j = 0$. The second order term is quite interesting: it shows that when two covariates are correlated, the effect of the trace Lasso is to shrink the corresponding coefficients toward each other. Another interesting remark is the fact that this term is very similar to pairwise elastic net penalties, which are of the form $|\mathbf{w}|^\top\mathbf{P}|\mathbf{w}|$, where $\mathbf{P}_{ij}$ is a decreasing function of $\Delta_{ij}$.

## 7.5 Experiments

In this section, we perform experiments on synthetic data to illustrate the behavior of the trace Lasso and other classical penalties when there are highly correlated covariates in the design matrix. First, we present how the synthetic data is generated, we then perform experiments in order to compare the two proposed optimization algorithms, and finally we compare the trace Lasso with other sparsity inducing norms.

### 7.5.1 Generation of synthetic data

The support $S$ of $\mathbf{w}$ is equal to $\{1,...,k\}$, where $k$ is the size of the support. For $i$ in the support of $\mathbf{w}$, $w_i$, is independently drawn from a uniform distribution over $[-1,1]$. The observations $\mathbf{x}_i$ are drawn from a multivariate Gaussian with mean 0 and covariance matrix $\Sigma$. For the first setting, $\Sigma$ is set to the identity, for the second setting, $\Sigma$ is block diagonal with blocks equal to $0.2\mathbf{I} + 0.8\mathbf{1}\mathbf{1}^\top$ corresponding to clusters of eight variables, finally for the third setting, we set $\Sigma_{ij} = 0.95^{|i-j|}$, corresponding to a Toeplitz design. Finally, we generate the response variables $y_i$ according to

$$y_i = \mathbf{w}^\top\mathbf{x}_i + \varepsilon_i,$$

where $\varepsilon_i$ is a zero mean Gaussian random variable with its variance set such that the signal-to-noise ratio is equal to 11.

### 7.5.2 Comparison of optimization algorithms

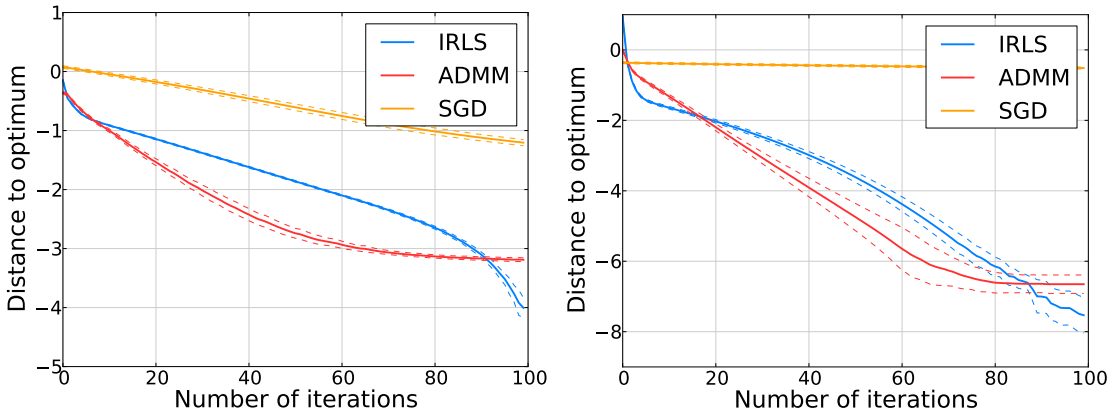
In this section, we compare the speed of convergence of the various algorithms we introduce to optimize the trace Lasso. For all experiments, we have $p = 256$, $n = 128$ and the support size $k = 16$. We consider a low correlations setting, corresponding to $\Sigma = \mathbf{I}$, and a strong correlations setting, corresponding to the Toeplitz setting. In the case of the ADMM algorithm, we use a conjugate gradient algorithm to optimize with respect to $\mathbf{w}$, as for the IRLS algorithm. We replicate the experiment over 10 runs, where only the noise vector $\varepsilon$ changes.

**Comments.** First, we observe that subgradient descent method is extremely slow to converge, and is thus not usable in practice. Second, we observe that there is not clear winner between the iteratively reweighted least-squares algorithm and the alternative direction method of multipliers. IRLS algorithm speed of convergence is slower during the first iterations, but then it converges faster to a high accuracy solution. ADMM should thus be preferred if high accuracy is not needed, and IRLS should be preferred otherwise. It should be noted that performance of ADMM might be improved by varying the parameter $\rho$ during the optimization (See Boyd et al., 2011, section 3.4.1 for an example of such scheme).

### 7.5.3 Comparison with other estimators

We now compare the trace Lasso with the ridge regression estimator, the Lasso, the elastic net and the pairwise elastic net. For each method, we choose the best $\lambda$. We perform a first sequence of experiments ($p = 1024$, $n = 256$) for which we report the estimation error. For the second serie of experiments ($p = 512$, $n = 128$), we report the Hamming distance between the estimated support and the true support.

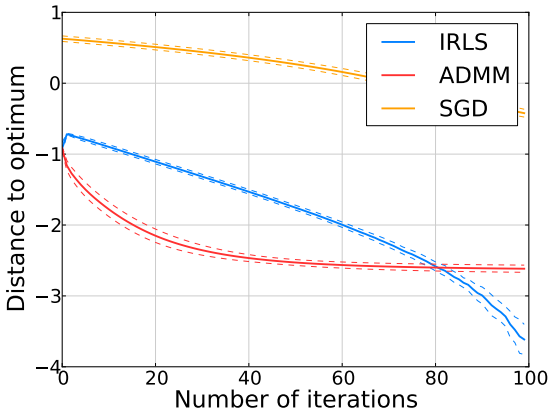
**Comments.** In all six graphs of Figure 7.4, we observe behaviors that are typical of the Lasso, ridge and elastic net: the Lasso performs very well on very sparse models but its performance degrades for denser models. The elastic net performs better than the Lasso for settings where there are strongly correlated covariates, thanks to its strongly convex $\ell_2$ term. In setting 1, since the variables are uncorrelated, there is no reason to couple their selection. This suggests that the Lasso should be the most appropriate convex regularization. The trace Lasso approaches the Lasso when $n$ is much larger than $p$, but the weak coupling induced by empirical correlations is sufficient to slightly decrease its performance compared to that of the Lasso. By contrast, in settings 2 and 3, the trace Lasso outperforms other methods (including the pairwise elastic net) since variables that should be selected together are indeed correlated. As for the penalized elastic net, since it takes into account the correlations between variables, it is not surprising that in experiments 2 and 3 it performs better than methods that do not. We do not have a compelling explanation for its superior performance in experiment 1.



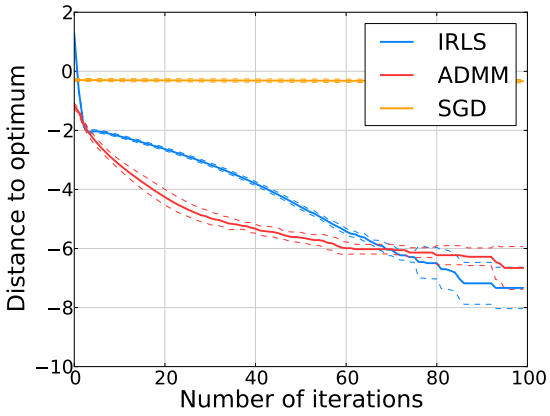

(a) Low corr., high reg.




(b) Low corr., low reg.




(c) High corr., high reg.




(d) High corr., low reg.

Figure 7.3: **Top:** low correlations, **bottom:** strong correlations. **Left:** strong regularization, **right:** low regularization. `IRLS` stands for iteratively reweighted least squares, `ADMM` stands for alternating direction method of multipliers and `SGD` stands for subgradient descent. The scale for the distance to optimum is logarithmic.

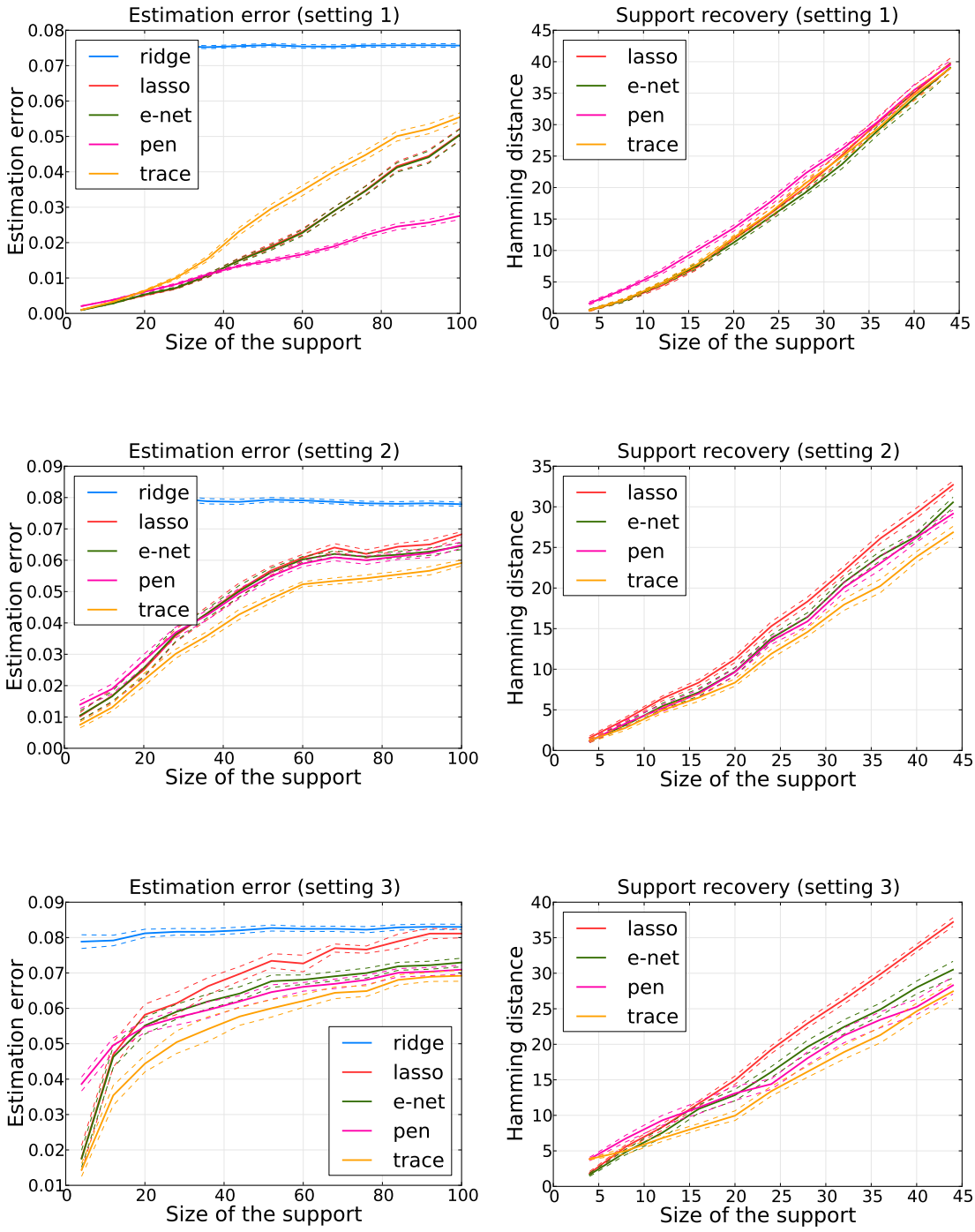



Figure 7.4: Left: estimation error ($p = 1024$, $n = 256$), right: support recovery ($p = 512$, $n = 128$). (Best seen in color. `e-net` stands for elastic net, `pen` stands for pairwise elastic net and `trace` stands for trace Lasso. Error bars are obtained over 20 runs.)

## 7.6 Conclusion

We introduce a new penalty function, the trace Lasso, which takes advantage of the correlation between covariates to add strong convexity exactly in the directions where needed, unlike the elastic net for example, which blindly adds a squared $\ell_2$-norm term in every directions. We show on synthetic data that this adaptive behavior leads to better estimation performance. In the future, we want to show that if a dedicated norm using prior knowledge such as the group Lasso can be used, the trace Lasso will behave similarly and its performance will not degrade too much, providing theoretical guarantees to such adaptivity. Finally, we will seek applications of this estimator in inverse problems such as deblurring, where the design matrix exhibits strong correlation structure.

# CHAPTER A



# SOME FACTS ABOUT THE TRACE NORM

## A.1 Perturbation of the trace norm

We follow the technique used in Bach (2008c) to obtain an approximation of the trace norm.

### A.1.1 Jordan-Wielandt matrices

Let $\mathbf{M} \in \mathbb{R}^{n \times p}$ of rank $r$. We note $s_1 \geq s_2 \geq ... \geq s_r > 0$, the strictly positive singular values of $\mathbf{M}$ and $\mathbf{u}_i$, $\mathbf{v}_i$ the associated left and right singular vectors. We introduce the Jordan-Wielandt matrix

$$\tilde{\mathbf{M}} = \begin{pmatrix} 0 & \mathbf{M} \\ \mathbf{M}^\top & 0 \end{pmatrix} \in \mathbb{R}^{(n+p)\times(n+p)}.$$

The singular values of $\mathbf{M}$ and the eigenvalues of $\tilde{\mathbf{M}}$ are related: $\tilde{\mathbf{M}}$ has eigenvalues $s_i$ and $s_{-i} = -s_i$ associated to eigenvectors

$$\mathbf{w}_i = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{u}_i \\ \mathbf{v}_i \end{pmatrix} \quad \text{and} \quad \mathbf{w}_{-i} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{u}_i \\ -\mathbf{v}_i \end{pmatrix}.$$

The remaining eigenvalues of $\tilde{\mathbf{M}}$ are equal to 0 and are associated to eigenvectors of the form

$$\mathbf{w} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \quad \text{and} \quad \mathbf{w} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{u} \\ -\mathbf{v} \end{pmatrix},$$

where $\forall\, i \in \{1, ..., r\},\ \mathbf{u}^\top \mathbf{u}_i = \mathbf{v}^\top \mathbf{v}_i = 0$.

### A.1.2 Cauchy residue formula

Let $\mathscr{C}$ be a closed curve that does not go through the eigenvalues of $\tilde{\mathbf{M}}$. We define

$$\Pi_{\mathscr{C}}(\tilde{\mathbf{M}}) = \frac{1}{2i\pi} \int_{\mathscr{C}} \lambda (\lambda \mathbf{I} - \tilde{\mathbf{M}})^{-1} d\lambda.$$

We have

$$\begin{aligned}\Pi_{\mathscr{C}}(\tilde{\mathbf{M}}) &= \frac{1}{2i\pi}\oint \sum_j \frac{\lambda}{\lambda - s_j}\mathbf{w}_j\mathbf{w}_j^\top d\lambda \\ &= \frac{1}{2i\pi}\oint \sum_j \left(1 + \frac{s_j}{\lambda - s_j}\right)\mathbf{w}_j\mathbf{w}_j^\top d\lambda \\ &= \sum_{s_j \in \mathscr{C}} s_j \mathbf{w}_j\mathbf{w}_j^\top.\end{aligned}$$

### A.1.3 Perturbation analysis

Let $\boldsymbol{\Delta} \in \mathbb{R}^{n\times p}$ be a perturbation matrix such that $||\boldsymbol{\Delta}||_{op} < s_r/4$, and let $\mathscr{C}$ be a closed curve around the $r$ largest eigenvalues of $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{M}}+\tilde{\boldsymbol{\Delta}}$. We can study the perturbation of the strictly positive singular values of $\mathbf{M}$ by computing the trace of $\Pi_{\mathscr{C}}(\tilde{\mathbf{M}}+\tilde{\boldsymbol{\Delta}}) - \Pi_{\mathscr{C}}(\tilde{\mathbf{M}})$. Using the fact that $(\lambda\mathbf{I} - \tilde{\mathbf{M}} - \tilde{\boldsymbol{\Delta}})^{-1} = (\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1} + (\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}\tilde{\boldsymbol{\Delta}}(\lambda\mathbf{I} - \tilde{\mathbf{M}} - \tilde{\boldsymbol{\Delta}})^{-1}$, we have

$$\begin{aligned}\Pi_{\mathscr{C}}(\tilde{\mathbf{M}}+\tilde{\boldsymbol{\Delta}}) - \Pi_{\mathscr{C}}(\tilde{\mathbf{M}}) = &\frac{1}{2i\pi}\oint \lambda(\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}\tilde{\boldsymbol{\Delta}}(\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}d\lambda \\ &+ \frac{1}{2i\pi}\oint \lambda(\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}\tilde{\boldsymbol{\Delta}}(\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}\tilde{\boldsymbol{\Delta}}(\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}d\lambda \\ &+ \frac{1}{2i\pi}\oint \lambda(\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}\tilde{\boldsymbol{\Delta}}(\lambda\mathbf{I} - \tilde{\mathbf{M}})^{-1}\tilde{\boldsymbol{\Delta}}(\lambda\mathbf{I} - \tilde{\mathbf{M}} - \tilde{\boldsymbol{\Delta}})^{-1}d\lambda.\end{aligned}$$

We note $A$ and $B$ the first two terms of the right hand side of this equation. We have

$$\begin{aligned}\mathrm{tr}(A) &= \sum_{j,k}\mathrm{tr}(\mathbf{w}_j\mathbf{w}_j^\top\tilde{\boldsymbol{\Delta}}\mathbf{w}_k\mathbf{w}_k^\top)\frac{1}{2i\pi}\oint_{\mathscr{C}}\frac{\lambda d\lambda}{(\lambda - s_j)(\lambda - s_k)} \\ &= \sum_j \mathrm{tr}(\mathbf{w}_j^\top\tilde{\boldsymbol{\Delta}}\mathbf{w}_j)\frac{1}{2i\pi}\oint_{\mathscr{C}}\frac{\lambda d\lambda}{(\lambda - s_j)^2} \\ &= \sum_j \mathrm{tr}(\mathbf{w}_j^\top\tilde{\boldsymbol{\Delta}}\mathbf{w}_j) \\ &= \sum_j \mathrm{tr}(\mathbf{u}_j^\top\boldsymbol{\Delta}\mathbf{v}_j),\end{aligned}$$

and

$$\begin{aligned}\mathrm{tr}(B) &= \sum_{j,k,l}\mathrm{tr}(\mathbf{w}_j\mathbf{w}_j^\top\tilde{\boldsymbol{\Delta}}\mathbf{w}_k\mathbf{w}_k^\top\tilde{\boldsymbol{\Delta}}\mathbf{w}_l\mathbf{w}_l^\top)\frac{1}{2i\pi}\oint_{\mathscr{C}}\frac{\lambda d\lambda}{(\lambda - s_j)(\lambda - s_k)(\lambda - s_l)} \\ &= \sum_{j,k}\mathrm{tr}(\mathbf{w}_j\tilde{\boldsymbol{\Delta}}\mathbf{w}_k\mathbf{w}_k\tilde{\boldsymbol{\Delta}}\mathbf{w}_j)\frac{1}{2i\pi}\oint_{\mathscr{C}}\frac{\lambda d\lambda}{(\lambda - s_j)^2(\lambda - s_k)}.\end{aligned}$$

If $s_j = s_k$, the integral is nul. Otherwise, we have

$$\frac{\lambda}{(\lambda - s_j)^2(\lambda - s_k)} = \frac{a}{\lambda - s_j} + \frac{b}{\lambda - s_k} + \frac{c}{(\lambda - s_j)^2},$$

where

$$a = \frac{-s_k}{(s_k - s_j)^2},$$
$$b = \frac{s_k}{(s_k - s_j)^2},$$
$$c = \frac{s_j}{s_j - s_k}.$$

Therefore, if $s_j$ and $s_k$ are both inside or outside the interior of $\mathscr{C}$, the integral is equal to zero. So

$$\begin{aligned}
\operatorname{tr}(B) &= \sum_{s_j>0}\sum_{s_k\le 0} \frac{-s_k(\mathbf{w}_j^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k)^2}{(s_j - s_k)^2} + \sum_{s_j\le 0}\sum_{s_k>0} \frac{s_k(\mathbf{w}_j^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k)^2}{(s_j - s_k)^2} \\
&= \sum_{s_j>0}\sum_{s_k>0} \frac{s_k(\mathbf{w}_j^\top \tilde{\mathbf{\Delta}} \mathbf{w}_{-k})^2}{(s_j + s_k)^2} + \sum_{s_j>0}\sum_{s_k>0} \frac{s_k(\mathbf{w}_{-j}^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k)^2}{(s_j + s_k)^2} + \sum_{s_j=0}\sum_{s_k>0} \frac{(\mathbf{w}_j^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k)^2}{s_k} \\
&= \sum_{s_j>0}\sum_{s_k>0} \frac{(\mathbf{w}_{-j}^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k)^2}{s_j + s_k} + \sum_{s_j=0}\sum_{s_k>0} \frac{(\mathbf{w}_j^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k)^2}{s_k}.
\end{aligned}$$

For $s_j > 0$ and $s_k > 0$, we have

$$\mathbf{w}_{-j}^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k = \frac{1}{2}\left(\mathbf{u}_j^\top \mathbf{\Delta} \mathbf{v}_k - \mathbf{u}_k^\top \mathbf{\Delta} \mathbf{v}_j\right),$$

and for $s_j = 0$ and $s_k > 0$, we have

$$\mathbf{w}_j^\top \tilde{\mathbf{\Delta}} \mathbf{w}_k = \frac{1}{2}\left(\pm \mathbf{u}_k^\top \mathbf{\Delta} \mathbf{v}_j + \mathbf{u}_j^\top \mathbf{\Delta} \mathbf{v}_k\right).$$

So

$$\operatorname{tr}(B) = \sum_{s_j>0}\sum_{s_k>0} \frac{(\mathbf{u}_j^\top \mathbf{\Delta} \mathbf{v}_k - \mathbf{u}_k^\top \mathbf{\Delta} \mathbf{v}_j)^2}{4(s_j + s_k)} + \sum_{s_j=0}\sum_{s_k>0} \frac{(\mathbf{u}_k^\top \mathbf{\Delta} \mathbf{v}_j)^2 + (\mathbf{u}_j^\top \mathbf{\Delta} \mathbf{v}_k)^2}{2s_k}.$$

Now, let $\mathscr{C}_0$ be the circle of center 0 and radius $s_r/2$. We can study the perturbation of the singular values of $\mathbf{M}$ equal to zero by computing the trace norm of $\Pi_{\mathscr{C}_0}(\tilde{\mathbf{M}} + \tilde{\mathbf{\Delta}}) - \Pi_{\mathscr{C}_0}(\tilde{\mathbf{M}})$. We

have

$$\Pi_{\mathscr{C}_0}(\tilde{\mathbf{M}}+\tilde{\mathbf{\Delta}})-\Pi_{\mathscr{C}_0}(\tilde{\mathbf{M}})=\frac{1}{2i\pi}\oint_{\mathscr{C}_0}\lambda(\lambda\mathbf{I}-\tilde{\mathbf{M}})^{-1}\tilde{\mathbf{\Delta}}(\lambda\mathbf{I}-\tilde{\mathbf{M}})^{-1}d\lambda$$
$$+\frac{1}{2i\pi}\oint_{\mathscr{C}_0}\lambda(\lambda\mathbf{I}-\tilde{\mathbf{M}})^{-1}\tilde{\mathbf{\Delta}}(\lambda\mathbf{I}-\tilde{\mathbf{M}})^{-1}\tilde{\mathbf{\Delta}}(\lambda\mathbf{I}-\tilde{\mathbf{M}})^{-1}d\lambda$$
$$+\frac{1}{2i\pi}\oint_{\mathscr{C}_0}\lambda(\lambda\mathbf{I}-\tilde{\mathbf{M}})^{-1}\tilde{\mathbf{\Delta}}(\lambda\mathbf{I}-\tilde{\mathbf{M}})^{-1}\tilde{\mathbf{\Delta}}(\lambda\mathbf{I}-\tilde{\mathbf{M}}-\tilde{\mathbf{\Delta}})^{-1}d\lambda.$$

Then, if we note the first integral $C$ and the second one $D$, we get

$$C=\sum_{j,k}\mathbf{w}_j\mathbf{w}_j^\top\tilde{\mathbf{\Delta}}\mathbf{w}_k\mathbf{w}_k^\top\frac{1}{2i\pi}\oint_{\mathscr{C}_0}\frac{\lambda d\lambda}{(\lambda-s_j)(\lambda-s_k)}.$$

If both $s_j$ and $s_k$ are outside $int(\mathscr{C}_0)$, then the integral is equal to zero. If one of them is inside, say $s_j$, then $s_j=0$ and the integral is equal to

$$\oint_{\mathscr{C}_0}\frac{d\lambda}{\lambda-s_k}$$

Then this integral is non nul if and only if $s_k$ is also inside $int(\mathscr{C}_0)$. Thus

$$\begin{aligned}C&=\sum_{j,k}\mathbf{w}_j\mathbf{w}_j^\top\tilde{\mathbf{\Delta}}\mathbf{w}_k\mathbf{w}_k^\top\mathbf{1}\{s_j\in int(\mathscr{C}_0)\mathbf{1}\{s_k\in int(\mathscr{C}_0)\\&=\sum_{s_j=0}\sum_{s_k=0}\mathbf{w}_j\mathbf{w}_j^\top\tilde{\mathbf{\Delta}}\mathbf{w}_k\mathbf{w}_k^\top\\&=\mathbf{W}_0\mathbf{W}_0^\top\tilde{\mathbf{\Delta}}\mathbf{W}_0\mathbf{W}_0^\top,\end{aligned}$$

where $\mathbf{W}_0$ are the eigenvectors associated to the eigenvalue 0. We have

$$D=\sum_{j,k,l}\mathbf{w}_j\mathbf{w}_j^\top\tilde{\mathbf{\Delta}}\mathbf{w}_k\mathbf{w}_k^\top\tilde{\mathbf{\Delta}}\mathbf{w}_l\mathbf{w}_l^\top\frac{1}{2i\pi}\oint_{\mathscr{C}_0}\frac{\lambda d\lambda}{(\lambda-s_j)(\lambda-s_k)(\lambda-s_l)}.$$

The integral is not equal to zero if and only if exactly one eigenvalue, say $s_i$, is outside $int(\mathscr{C}_0)$. The integral is then equal to $-1/s_i$. Thus

$$D=-\mathbf{W}_0\mathbf{W}_0^\top\tilde{\mathbf{\Delta}}\mathbf{W}_0\mathbf{W}_0^\top\tilde{\mathbf{\Delta}}\mathbf{W}S^{-1}\mathbf{W}^\top-\mathbf{W}S^{-1}\mathbf{W}^\top\tilde{\mathbf{\Delta}}\mathbf{W}_0\mathbf{W}_0^\top\tilde{\mathbf{\Delta}}\mathbf{W}_0\mathbf{W}_0^\top$$
$$-\mathbf{W}_0\mathbf{W}_0^\top\tilde{\mathbf{\Delta}}\mathbf{W}S^{-1}\mathbf{W}^\top\tilde{\mathbf{\Delta}}\mathbf{W}_0\mathbf{W}_0^\top,$$

where $S=\mathrm{Diag}(-\mathbf{s},\ \mathbf{s})$. Finally, putting everything together, we get

**Proposition 8.** *Let* $\mathbf{M} = \mathbf{U}\text{Diag}(\mathbf{s})\mathbf{V}^\top \in \mathbb{R}^{n\times p}$*, the singular value decomposition of* $\mathbf{M}$*, with* $\mathbf{U} \in \mathbb{R}^{n\times r}$, $\mathbf{V} \in \mathbb{R}^{p\times r}$*. Let* $\mathbf{\Delta} \in \mathbb{R}^{n\times p}$*. We have*

$$||\mathbf{M}+\mathbf{\Delta}||_* = ||\mathbf{M}||_* + ||\mathbf{Q}||_* + \text{tr}(\mathbf{V}\mathbf{U}^\top\mathbf{\Delta}) + \sum_{s_j>0}\sum_{s_k>0}\frac{(\mathbf{u}_j^\top\mathbf{\Delta}\mathbf{v}_k - \mathbf{u}_k^\top\mathbf{\Delta}\mathbf{v}_j)^2}{4(s_j+s_k)} + \sum_{s_j=0}\sum_{s_k>0}\frac{(\mathbf{u}_k^\top\mathbf{\Delta}\mathbf{v}_{0_j})^2 + (\mathbf{u}_{0_j}^\top\mathbf{\Delta}\mathbf{v}_k)^2}{2s_k} + o(||\mathbf{\Delta}||^2),$$

*where*

$$\mathbf{Q} = \mathbf{U}_0^\top\mathbf{\Delta}\mathbf{V}_0 - \mathbf{U}_0^\top\mathbf{\Delta}\mathbf{V}_0\mathbf{V}_0^\top\mathbf{\Delta}^\top\mathbf{U}\text{Diag}(\mathbf{s})^{-1} - \text{Diag}(\mathbf{s})^{-1}\mathbf{V}^\top\mathbf{\Delta}^\top\mathbf{U}_0\mathbf{U}_0^\top\mathbf{\Delta}\mathbf{V}_0 - \mathbf{U}_0^\top\mathbf{\Delta}\mathbf{V}\text{Diag}(\mathbf{s})^{-1}\mathbf{U}^\top\mathbf{\Delta}\mathbf{V}_0.$$

## A.2 Proof of proposition 2

In this section, we prove that if the loss function is strongly convex with respect to its second argument, then the solution of the penalized empirical risk minimization is unique.

Let $\hat{\mathbf{w}} \in \text{argmin}_\mathbf{w}\sum_{i=1}^n \ell(y_i, \mathbf{w}^\top\mathbf{x}_i) + \lambda||\mathbf{X}\text{Diag}(\mathbf{w})||_*$. If $\hat{\mathbf{w}}$ is in the nullspace of $\mathbf{X}$, then $\hat{\mathbf{w}} = 0$ and the minimum is unique. From now on, we suppose that the minima are not in the nullspace of $\mathbf{X}$.

Let $\mathbf{u}, \mathbf{v} \in \text{argmin}_\mathbf{w}\sum_{i=1}^n \ell(y_i, \mathbf{w}^\top\mathbf{x}_i) + \lambda||\mathbf{X}\text{Diag}(\mathbf{w})||_*$ and $\delta = \mathbf{v} - \mathbf{u}$. By convexity of the objective function, all the $\mathbf{w} = \mathbf{u} + t\delta$, for $t \in ]0,1[$ are also optimal solutions, and so, we can choose an optimal solution $\mathbf{w}$ such that $w_i \neq 0$ for all $i$ in the support of $\delta$. Because the loss function is strongly convex outside the nullspace of $\mathbf{X}$, $\delta$ is in the nullspace of $\mathbf{X}$.

Let $\mathbf{X}\text{Diag}(\mathbf{w}) = \mathbf{U}\text{Diag}(\mathbf{s})\mathbf{V}^\top$ be the SVD of $\mathbf{X}\text{Diag}(\mathbf{w})$. We have the following development around $\mathbf{w}$:

$$||\mathbf{X}\text{Diag}(\mathbf{w}+t\delta)||_* = ||\mathbf{X}\text{Diag}(\mathbf{w})||_* + \text{tr}(\text{Diag}(t\delta)\mathbf{X}^\top\mathbf{U}\mathbf{V}^\top) + \sum_{s_i>0}\sum_{s_j>0}\frac{\text{tr}(\text{Diag}(t\delta)\mathbf{X}^\top(\mathbf{u}_i\mathbf{v}_j^\top - \mathbf{u}_j\mathbf{v}_i^\top))^2}{4(s_i+s_j)} + \sum_{s_i>0}\sum_{s_j=0}\frac{\text{tr}(\text{Diag}(t\delta)\mathbf{X}^\top\mathbf{u}_i\mathbf{v}_j^\top)^2}{2s_i} + o(t^2).$$

We note $S$ the support of $\mathbf{w}$. Using the fact that the support of $\delta$ is included in $S$, we have $\mathbf{X}\text{Diag}(t\delta) = \mathbf{X}\text{Diag}(\mathbf{w})\text{Diag}(t\gamma)$, where $\gamma_i = \frac{\delta_i}{w_i}$ for $i \in S$ and 0 otherwise. Then:

$$||\mathbf{X}\text{Diag}(\mathbf{w}+t\delta)||_* = ||\mathbf{X}\text{Diag}(\mathbf{w})||_* + t\gamma^\top\text{diag}(\mathbf{V}\text{Diag}(\mathbf{s})\mathbf{V}^\top) + \sum_{s_i>0}\sum_{s_j>0}\frac{t^2\,\text{tr}\left((s_i-s_j)\text{Diag}(\gamma)\mathbf{v}_i\mathbf{v}_j^\top\right)^2}{4(s_i+s_j)} + \sum_{s_i>0}\sum_{s_j=0}\frac{t^2\,\text{tr}\left(s_i\text{Diag}(\gamma)\mathbf{v}_i\mathbf{v}_j^\top\right)^2}{2s_i} + o(t^2).$$

For small $t$, $\mathbf{w} + t\delta$ is also a minimum, and therefore, we have:

$$\forall\ s_i > 0,\ s_j > 0, \quad (s_i - s_j)\operatorname{tr}\left(\operatorname{Diag}(\gamma)\mathbf{v}_i\mathbf{v}_j^\top\right) = 0, \tag{A.1}$$

$$\forall\ s_i > 0,\ s_j = 0, \quad \operatorname{tr}\left(\operatorname{Diag}(\gamma)\mathbf{v}_i\mathbf{v}_j^\top\right) = 0. \tag{A.2}$$

This could be summarized as

$$\forall\ s_i \neq s_j, \quad \mathbf{v}_i^\top(\operatorname{Diag}(\gamma)\mathbf{v}_j) = 0. \tag{A.3}$$

This means that the eigenspaces of $\operatorname{Diag}(\mathbf{w})\mathbf{X}^\top\mathbf{X}\operatorname{Diag}(\mathbf{w})$ are stable by the matrix $\operatorname{Diag}(\gamma)$. Therefore, $\operatorname{Diag}(\mathbf{w})\mathbf{X}^\top\mathbf{X}\operatorname{Diag}(\mathbf{w})$ and $\operatorname{Diag}(\gamma)$ are simultaneously diagonalizable and so, they commute. Therefore:

$$\forall\ i, j \in S, \quad \sigma_{ij}\gamma_i = \sigma_{ij}\gamma_j \tag{A.4}$$

where $\sigma_{ij} = [\mathbf{X}^\top\mathbf{X}]_{ij}$. We define a partition $(S_k)$ of $S$, such that $i$ and $j$ are in the same set $S_k$ if there exists a path $i = a_1, ..., a_m = j$ such that $\sigma_{a_n, a_{n+1}} \neq 0$ for all $n \in \{1, ..., m-1\}$. Then, using equation (A.4), $\gamma$ is constant on each $S_k$. $\delta$ being in the nullspace of $\mathbf{X}$, we have:

$$0 = \delta^\top\mathbf{X}^\top\mathbf{X}\delta \tag{A.5}$$

$$= \sum_{S_k}\sum_{S_l}\delta_{S_k}^\top\mathbf{X}^\top\mathbf{X}\delta_{S_l} \tag{A.6}$$

$$= \sum_{S_k}\delta_{S_k}^\top\mathbf{X}^\top\mathbf{X}\delta_{S_k} \tag{A.7}$$

$$= \sum_{S_k}||\mathbf{X}\delta_{S_k}||_2^2. \tag{A.8}$$

So for all $S_i$, $\mathbf{X}\delta_{S_i} = 0$. Since a predictor $\mathbf{X}_i$ is orthogonal to all the predictors belonging to other groups defined by the partition $(S_k)$, we can decompose the norm $\Omega$:

$$||\mathbf{X}\operatorname{Diag}(\mathbf{w})||_* = \sum_{S_k}||\mathbf{X}\operatorname{Diag}(\mathbf{w}_{S_k})||_*. \tag{A.9}$$

We recall that $\gamma$ is constant on each $S_k$ and so $\delta_{S_k}$ is colinear to $\mathbf{w}_{S_i}$, by definition of $\gamma$. If $\delta_{S_i}$ is not equal to zero, this means that $\mathbf{w}_{S_i}$, which is not equal to zero, is in the nullspace of $\mathbf{X}$. Replacing $\mathbf{w}_{S_i}$ by 0 will not change the value of the data fitting term but it will strictly decreases the value of the norm $\Omega$. This is a contradiction with the optimality of $\mathbf{w}$. Thus all the $\delta_{S_i}$ are equal to zero and the minimum is unique.

## A.3 Proof of proposition 3

For the first inequality, we have

$$\begin{aligned}||\mathbf{w}||_2 &= ||\mathbf{P}\operatorname{Diag}(\mathbf{w})||_F \\ &\leq ||\mathbf{P}\operatorname{Diag}(\mathbf{w})||_*.\end{aligned}$$

For the second inequality, we have

$$\begin{aligned}
||\mathbf{P}\,\mathrm{Diag}(\mathbf{w})||_* &= \max_{||\mathbf{M}||_{op}\leq 1} \mathrm{tr}\left(\mathbf{M}^\top \mathbf{P}\,\mathrm{Diag}(\mathbf{w})\right) \\
&= \max_{||\mathbf{M}||_{op}\leq 1} \mathrm{diag}\left(\mathbf{M}^\top \mathbf{P}\right)^\top \mathbf{w} \\
&\leq \max_{||\mathbf{M}||_{op}\leq 1} \sum_{i=1}^{p} |\mathbf{M}^{(i)\top}\mathbf{P}^{(i)}|\,|w_i| \\
&\leq ||\mathbf{w}||_1.
\end{aligned}$$

The first equality is the fact that the dual norm of the trace norm is the operator norm and the second inequality uses the fact that all matrices of operator norm smaller than one have columns of $\ell_2$ norm smaller than one.

# BIBLIOGRAPHY


Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2012). Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559*.

Argyriou, A., Evgeniou, T., and Pontil, M. (2007). Multi-task feature learning. *Advances in neural information processing systems*, 19:41.

Bach, F. (2008a). Bolasso: model consistent Lasso estimation through the bootstrap. In *Proceedings of the 25th international conference on Machine learning*, pages 33–40. ACM.

Bach, F. (2008b). Consistency of the group Lasso and multiple kernel learning. *The Journal of Machine Learning Research*, 9:1179–1225.

Bach, F. (2008c). Consistency of trace norm minimization. *The Journal of Machine Learning Research*, 9:1019–1048.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 3(2-3).

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the zurich acm-gamm conference. *Proceedings of the International Comference on Information Processing, 1959*.

Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Baroni, M. and Lenci, A. (2011). How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics.

Baroni, M. and Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Bickel, P., Ritov, Y., and Tsybakov, A. (2009). Simultaneous analysis of Lasso and Dantzig selector. *The Annals of Statistics*, 37(4):1705–1732.

Blei, D., Griffiths, T., Jordan, M., and Tenenbaum, J. (2004). Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:106–114.

Blei, D. M. and Lafferty, J. D. (2006). Correlated topic models. In *Advances in neural information processing systems*, pages 147–154.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge Univ Pr.

Boyd-Graber, J., Blei, D. M., and Zhu, X. (2007). A topic model for word sense disambiguation. In *EMNLP*.

Boyd-Graber, J. L. and Blei, D. (2009). Syntactic topic models. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 185–192.

Brown, P. F., deSouza, P. V., Mercer, R. L., Della Pietra, V. J., and Lai, J. C. (1992). Class-based *n*-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Buntine, W. (2002). Variational extensions to em and multinomial pca. In *Machine Learning: ECML 2002*, pages 23–34. Springer.

Candito, M., Nivre, J., Denis, P., and Anguiano, E. H. (2010). Benchmarking of statistical dependency parsers for french. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 108–116. Association for Computational Linguistics.

Cappé, O. and Moulines, E. (2009). On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613.

Chen, S. and Donoho, D. (1994). Basis pursuit. In *Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on*, volume 1, pages 41–44. IEEE.

Chen, S. S., Donoho, D. L., and Saunders, M. A. (1998). Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 20(1):33–61.

Chomsky, N. (1956). Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124.

Ciaramita, M. and Altun, Y. (2006). Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602, Sydney, Australia. Association for Computational Linguistics.

Combettes, P. L. and Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer.

Curran, J. R. and Moens, M. (2002). Scaling context space. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 231–238. Association for Computational Linguistics.

Daubechies, I., DeVore, R., Fornasier, M., and Güntürk, C. S. (2010). Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38.

Davis, G., Mallat, S., and Avellaneda, M. (1997). Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98.

Davis, G., Mallat, S., and Zhang, Z. (1994). Adaptive time-frequency decompositions with matching pursuit. *Optical Engineering*, 33(7).

De Marneffe, M. C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

De Marneffe, M. C. and Manning, C. D. (2008). The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*.

Donoho, D. L. and Johnstone, J. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499.

Eisenstein, J., O'Connor, B., Smith, N. A., and Xing, E. P. (2010). A latent variable model for geographic lexical variation. In *EMNLP*.

Faruqui, M., Padó, S., and Sprachverarbeitung, M. (2010). Training and evaluating a German named entity recognizer with semantic generalization. *Semantic Approaches in Natural Language Processing*.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2001). Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Firth, J. R. (1957). *A synopsis of linguistic theory, 1930-1955*.

Freitag, D. (2004). Trained named entity recognition using distributional clusters. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Gaussier, E. and Goutte, C. (2005). Relation between plsa and nmf and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602. ACM.

Golub, G. and Van Loan, C. (1996). *Matrix computations*. Johns Hopkins Univ Pr.

Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Grandvalet, Y. and Canu, S. (1999). Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. *Advances in Neural Information Processing Systems 11: Proceedings of the 1998 Confernce*, 11:445.

Grave, E., Obozinski, G., and Bach, F. (2013a). Domain adaptation for sequence labeling using hidden Markov models. In *Proceedings of the NIPS Workshop: new directions in transfer and multi-task: learning across domains and tasks*.

Grave, E., Obozinski, G., and Bach, F. (2013b). Hidden Markov tree models for semantic class induction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 94–103, Sofia, Bulgaria. Association for Computational Linguistics.

Grave, E., Obozinski, G. R., and Bach, F. (2011). Trace lasso: a trace norm regularization for correlated designs. In *Advances in Neural Information Processing Systems (NIPS)*.

Grefenstette, E. and Sadrzadeh, M. (2011). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Griffiths, T. L., Steyvers, M., Blei, D. M., and Tenenbaum, J. B. (2005). Integrating topics and syntax. *Advances in Neural Information Processing Systems*, 17:537–544.

Guevara, E. (2010). A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 33–37. Association for Computational Linguistics.

Guevara, E. (2011). Computing semantic compositionality in distributional semantics. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, pages 135–144. Citeseer.

Haffari, G., Razavi, M., and Sarkar, A. (2011). An ensemble model that combines syntactic and semantic clustering for discriminative dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Harris, Z. S. (1954). *Distributional structure*. Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). The elements of statistical learning.

Hoerl, A. and Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*.

Huang, F., Ahuja, A., Downey, D., Yang, Y., Guo, Y., and Yates, A. (2013). Learning representations for weakly supervised natural language processing tasks. *Computational Linguistics*.

Huang, F., Yates, A., Ahuja, A., and Downey, D. (2011). Language models as representations for weakly supervised nlp tasks. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 125–134, Portland, Oregon, USA. Association for Computational Linguistics.

Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440. ACM.

Jenatton, R., Audibert, J.-Y., and Bach, F. (2011). Structured variable selection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824.

Jenatton, R., Le Roux, N., Bordes, A., Obozinski, G., et al. (2012). A latent factor model for highly multi-relational data. In *NIPS 2012-Neural Information Processing Systems*.

Jurafsky, D. and Martin, J. H. (2000). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall.

Kneser, R. and Ney, H. (1993). Improved clustering techniques for class-based statistical language modelling. In *Third European Conference on Speech Communication and Technology*.

Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning*.

Landauer, T. K. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Lavergne, T., Cappé, O., and Yvon, F. (2010). Practical very large scale crfs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513. Association for Computational Linguistics.

Li, W. and McCallum, A. (2005). Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of the National Conference on Artificial Intelligence*.

Liang, P. (2005). Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Liang, P. and Klein, D. (2009). Online EM for unsupervised models. In *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 611–619.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 768–774. Association for Computational Linguistics.

Liu, H., Roeder, K., and Wasserman, L. (2010). Stability approach to regularization selection (stars) for high dimensional graphical models. *Advances in Neural Information Processing Systems*, 23.

Lorbert, A., Eis, D., Kostina, V., Blei, D. M., and Ramadge, P. J. (2010). Exploiting covariate similarity in sparse regression via the pairwise elastic net. *JMLR - Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 9:477–484.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*.

Mallat, S. and Zhang, Z. (1993a). Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415.

Mallat, S. G. and Zhang, Z. (1993b). Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL*.

Misra, H., Yvon, F., Jose, J. M., and Cappe, O. (2009). Text segmentation via topic modeling: an analytical study. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1553–1556. ACM.

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *ACL*, pages 236–244.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, 34(8).

Montague, R. (1970). Universal grammar. *Theoria*, 36(3):373–398.

Montague, R. (1973). The proper treatment of quantification in ordinary english. In *Approaches to natural language*, pages 221–242. Springer.

Navigli, R. (2009). Word Sense Disambiguation: a survey. *ACM Computing Surveys*, 41(2):1–69.

O'Connor, B., Stewart, B. M., and Smith, N. A. (2013). Learning to extract international relations from political context. In *Association of Computational Linguistics*.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Pal, C., Sutton, C., and McCallum, A. (2006). Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *ICASSP 2006 Proceedings.*, volume 5, pages V–V. IEEE.

Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE.

Petrov, S. (2009). *Coarse-to-Fine Natural Language Processing*. PhD thesis, University of California at Bekeley, Berkeley, CA, USA.

Petrov, S. and McDonald, R. (2012). Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Sandhaus, E. (2008). The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Schutze, H. (1992). Dimensions of meaning. In *Supercomputing'92. Proceedings*, pages 787–796. IEEE.

Séaghdha, D. O. (2010). Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics.

Seeger, M. (2008). Bayesian inference and optimal design for the sparse linear model. *The Journal of Machine Learning Research*, 9:759–813.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4).

Täckström, O., McDonald, R., and Uszkoreit, J. (2012). Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics*.

Tesnière, L. (1959). *Eléments de syntaxe structurale*, volume 1965. Klincksieck Paris.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. In *Soviet Math. Dokl.*, volume 5, page 1035.

Titov, I. and Klementiev, A. (2012). A Bayesian approach to unsupervised semantic role induction. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France.

Toutanova, K. and Johnson, M. (2007). A bayesian lda-based model for semi-supervised part-of-speech tagging. In *NIPS*, pages 1521–1528.

Tratz, S. and Hovy, E. (2011). A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Uszkoreit, J. and Brants, T. (2008). Distributed word clustering for large scale class-based language modeling in machine translation. *Proceedings of ACL-08: HLT*.

Van de Cruys, T. (2010). *Mining for Meaning. The Extraction of Lexico-Semantic Knowledge from Text*. PhD thesis, University of Groningen, The Netherlands.

Van de Cruys, T., Poibeau, T., and Korhonen, A. (2013). A tensor-based factorization model of semantic compositionality. In *Proceedings of NAACL-HLT*, pages 1142–1151.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.

Vecchi, E. M., Baroni, M., and Zamparelli, R. (2011). (linear) maps of the impossible: Capturing semantic anomalies in distributional space. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 1–9, Portland, Oregon, USA. Association for Computational Linguistics.

Wainwright, M. (2009). Sharp thresholds for noisy and high-dimensional recovery of sparsity using $\ell_1$-constrained quadratic programming (lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Wright, S. J., Nowak, R. D., and Figueiredo, M. A. (2009). Sparse reconstruction by separable approximation. *Signal Processing, IEEE Transactions on*, 57(7):2479–2493.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Yuan, M. and Lin, Y. (2007). On the non-negative garrotte estimator. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):143–161.

Zhang, T. (2008). Adaptive forward-backward greedy algorithm for sparse learning with linear models. *Advances in Neural Information Processing Systems*, 22.

Zhao, P., Rocha, G., and Yu, B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497.

Zhao, P. and Yu, B. (2006). On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

**Abstract:** This thesis, which is organized in two independent parts, presents work on distributional semantics and on variable selection.

In the first part, we introduce a new method for learning good word representations using large quantities of unlabeled sentences. The method is based on a probabilistic model of sentence, using a hidden Markov model and a syntactic dependency tree. The latent variables, which correspond to the nodes of the dependency tree, aim at capturing the meanings of the words. We develop an efficient algorithm to perform inference and learning in those models, based on online EM and approximate message passing. We then evaluate our models on intrinsic tasks such as predicting human similarity judgements or word categorization, and on two extrinsic tasks: named entity recognition and supersense tagging.

In the second part, we introduce, in the context of linear models, a new penalty function to perform variable selection in the case of highly correlated predictors. This penalty, called the trace Lasso, uses the trace norm of the selected predictors, which is a convex surrogate of their rank, as the criterion of model complexity. The trace Lasso interpolates between the $\ell_1$-norm and $\ell_2$-norm. In particular, it is equal to the $\ell_1$-norm if all predictors are orthogonal and to the $\ell_2$-norm if all predictors are equal. We propose two algorithms to compute the solution of least-squares regression regularized by the trace Lasso, and perform experiments on synthetic datasets to illustrate the behavior of the trace Lasso.

KEYWORDS: distributional semantics; hidden Markov model; dependency tree; word representation; semantic class; variable selection; trace Lasso.

**Résumé :** Cette thèse, organisée en deux parties indépendantes, a pour objet la semantique distributionnelle et la sélection de variables.

Dans la première partie, nous introduisons une nouvelle méthode pour l'apprentissage de représentations de mots à partir de grandes quantités de texte brut. Cette méthode repose sur un modèle probabiliste de la phrase, utilisant modèle de Markov caché et arbre de dépendance. Nous présentons un algorithme efficace pour réaliser l'inférence et l'apprentissage dans un tel modèle, fondé sur l'algorithme EM en ligne et la propagation de message approchée. Nous évaluons les modèles obtenus sur des taches intrinsèques, telles que prédire des jugements de similarité humains ou catégoriser des mots et deux taches extrinsèques : la reconnaissance d'entités nommées et l'étiquetage en supersens.

Dans la seconde partie, nous introduisons, dans le contexte des modèles linéaires, une nouvelle pénalité pour la sélection de variables en présence de prédicteurs fortement corrélés. Cette pénalité, appelée *trace Lasso*, utilise la norm trace des prédicteurs sélectionnés, qui est une relaxation convexe de leur rang, comme critère de complexité. Le trace Lasso interpole les normes $\ell_1$ et $\ell_2$. En particulier, lorsque tous les prédicteurs sont orthogonaux, il est égal à la norme $\ell_1$, tandis que lorsque tous les prédicteurs sont égaux, il est égal à la norme $\ell_2$. Nous proposons deux algorithmes pour calculer la solution du problème de régression aux moindres carrés regularisé par le trace Lasso et réalisons des expériences sur des données synthétiques.

MOTS CLÉS : sémantique distributionnelle ; modèle de Markov caché ; arbre de dépendance ; représentation de mots ; classe sémantique ; sélection de variables ; trace Lasso.