Sparse Models of Natural Language Text

Dani Yogatama

CMU-LTI-15-005

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Noah A. Smith (chair), Carnegie Mellon University
Chris Dyer, Carnegie Mellon University
Alexander J. Smola, Carnegie Mellon University
Francis Bach, INRIA

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

For G.C.W.

Abstract

In statistical text analysis, many learning problems can be formulated as a minimization of a sum of a loss function and a regularization function for a vector of parameters (feature coefficients). The loss function drives the model to learn generalizable patterns from the training data, whereas the regularizer plays two important roles: to prevent the models from capturing idiosyncrasies of the training data (overfitting) and to encode prior knowledge about the model parameters.

When learning from high-dimensional data such as text, it has been empirically observed that relatively few dimensions are relevant to the predictive task (Forman, 2003). How can we capitalize on this insight and choose which dimensions are relevant in an informed and principled manner? Sparse regularizers provide a way to select relevant dimensions by means of regularization. However, past work rarely encodes non-trivial prior knowledge that yields sparse solutions through a regularization function. This thesis investigates the applications of sparse models—especially structured sparse models—as a medium to encode linguistically-motivated prior knowledge in textual models to advance NLP systems. We explore applications of sparse NLP models in temporal models of text, word embeddings, and text categorization.

Sparse models come with their own challenges, since new instantiations of sparse models often require a specialized optimization method. This thesis also presents optimization methods for the proposed instantiations of sparse models. Therefore, the goals of this thesis are twofold: (i) to show how sparsity can be used to encode linguistic information in statistical text models, and (ii) to develop efficient learning algorithms to solve the resulting optimization problems.

Acknowledgement

Noah Smith, for being a constant source of inspiration and making graduate school so much fun. Thank you for believing in me. Wherever I go from here, I am always indebted to you.

My committee—Chris Dyer, Alex Smola, and Francis Bach—for many valuable technical discussions and helpful career advice.

My fellow ARK members, past and present: Waleed Ammar, David Bamman, Dallas Card, Shay Cohen, Dipanjan Das, Jesse Dodge, Jeff Flanigan, Kevin Gimpel, Michael Heilman, Lingpeng Kong, Fei Liu, Andre Martins, Brendan O'Connor Rohan Ramanath, Bryan Routledge, Nathan Schneider, Yanchuan Sim, Swabha Swayamdipta, Sam Thomson, and Tae Yano; and other CMU friends—Manaal Faruqui, Anuj Goyal, Meghana Kshirsagar, Wang Ling, Ming Sun, and Yulia Tsvetkov.

My family—my parents and brothers—for always supporting my choice to go to graduate school in the U.S. I wish I could come home more often.

Finally, I thank Gina, without you getting a PhD would not mean as much.

Contents

Chapter 1

Introduction

What is the best way to exploit linguistic information in statistical text processing models? Much recent work in NLP has focused on error-prone and time-consuming linguistic feature engineering. For example, we can introduce higher-order n-grams (Kogan *et al.*, 2009), part-of-speech tags, dependency relations (Joshi *et al.*, 2010), Brown clusters, or LDA topics into a textual model as additional features.

In this thesis, we propose to use *sparse models* to answer this question. Sparse regularizers provide a way to define structures in a parameter space by means of regularization. Since the seminal work of Chen and Rosenfeld (2000), the importance of regularization in models of text—including language modeling, structured prediction, and classification—has been widely recognized. The emphasis, however, has largely been on one specific kind of inductive bias: avoiding large weights (i.e., coefficients in a linear model). Ridge (Hoerl and Kennard, 1970) and lasso (Tibshirani, 1996) regularizations, which penalize weights by their squared ℓ_2 and ℓ_1 norm respectively, are examples of such methods.

Recently, structured (or composite) regularization has been introduced. The most widely explored variant, group lasso (Yuan and Lin, 2006) seeks to avoid large $\ell_{1,2}$ or $\ell_{1,\infty}$ norms for *groups* of weights. Group lasso has been shown useful in a range of applications, including computational biology (Kim and Xing, 2008), signal processing (Lv *et al.*, 2011), and NLP (Eisenstein *et al.*, 2011a; Martins *et al.*, 2011b; Nelakanti *et al.*, 2013). Another variant is fused lasso (Tibshirani *et al.*, 2005), which penalizes differences in weights between successive features, when the features can be organized into a

meaningful order (e.g., chain structure, graph structure).

In statistical text analysis, we encode linguistic information by defining linguistically motivated structures in a parameter space. For example, in categorizing text documents, oftentimes only some parts of an observation are important to the task. In the "bag of words" representation, most words can typically be ignored (Forman, 2003). However, existing methods penalize these words in isolation, ignoring the sentential, syntactic, and semantic structures of text documents. Structured sparse regularizers provide a way to incorporate these structures in the form of sparsity patterns into a text model.

What are the benefits of having sparse models? First, as we will show, they can be used to encode prior knowledge in the sparsity patterns. Second, they are lightweight—requiring less memory to store and allowing faster inference and easier interpretability. Nowadays, we often start with models with hundreds of millions to billions of parameters. Sparsity provides a way to completely discard some of these parameters in an informed and principled manner, resulting in smaller model size. For example, mobile applications (e.g., Google Now, Siri, etc.) stand to benefit from smaller models since mobile phones typically have less storage and computing power than standard computers. Sparse models come with their own challenges. New varieties of sparse models often require a specialized optimization method, as we will see throughout this thesis. Last, some of the state-of-the-art methods for benchmarks tasks in various application areas such as computational biology (Kim and Xing, 2008) and computer vision are sparse models (Ranzato *et al.*, 2006; Lin and Kung, 2014), empirically demonstrating that they can also lead to statistical improvements if the prior knowledge is correct (Stojnic *et al.*, 2009).

Sparse models come with their own challenges. New varieties of sparse models often require a specialized optimization method. The goals of this thesis are twofold: (i) to show how sparsity can be used to encode linguistic information in statistical text models, and (ii) to develop efficient learning algorithms to solve the resulting optimization problems. There are many varieties of statistical text models. In this thesis, we investigate several text models, including models for text categorization, word embeddings, and temporal adaptations. We design different sparse regularizers with varying motivations depending on the characteristics of each problem. The main contents of this thesis—Chapters 3, 4, and 5—are ordered based on the complexity of structures defined

in the respective parameter space associated with each problem, from lowest to highest.

This thesis consists of five parts.

- In Chapter 2, we discuss relevant background of sparse models, including connections to feature selection.

- Chapter 3 discusses applications of sparsity in dynamic models of text. Many datasets evolve rapidly based on events in the real world, but most NLP models can be categorized as "static" models—they are not aware of natural changes in the data over time. We introduce a sparse and adaptive Bayesian prior—which can be used as structured regularizers in any generalized linear model—for time-dependent model parameters that leverages the intuition that the relationship between observable features and response variables should evolve smoothly over time. We show variational inference algorithms to perform learning and inference efficiently.

- Chapter 4 considers an application of structured regularizers in learning text representations. Specifically, we consider the problem of learning word representations (embeddings). The two predominant approaches in learning word representations are based on inducing either distributional or distributed representations. The learned dimensions in both approaches are not easily interpretable, and the word representations are dense. We propose to use structured sparse coding for learning sparse word embeddings. Our method promotes sparsity in the embedding space and organizes these lower dimensions into linguistically meaningful concepts (Collins and Quillian, 1969). We also introduce an efficient optimization method to perform sparse coding on large datasets.

- In Chapter 5, we introduce a class of structured sparse regularizers for text categorization that allows incorporation of linguistic information in the form of: sentence boundaries, parse trees, topics, or hierarchical word clusters. Our regularizers impose linguistic bias in feature weights without introducing new features, enabling us to incorporate prior knowledge into conventional bag-of-words models. Defining structures based on linguistic cues introduces a new technical challenge, since a typical corpus may contain billions of words, millions of sentences, and thousands of semantic topics. We also show how to solve the resulting optimization

challenge with the alternating directions method of multipliers for solving penalized optimization problems with thousands to millions of (possibly overlapping) structures (ADMM; Hestenes, 1969; Powell, 1969).

- The last part of this thesis (Chapter 6) outlines future directions of sparse models of natural language text.

Chapter 2

Sparsity

There are several meanings of the word "sparsity" in machine learning and natural language processing. The two most common definitions are:

- Data sparsity: the number of training examples is too small to get a good estimate of model parameters. This is sometimes referred as the "curse of dimensionality."
- Model sparsity: the assumption that given a large set of features, many dimensions of model parameters are not needed for the task at hand. We can set these parameters to zero, leading to a sparse model. For example, in a linear model, given an input example $\boldsymbol{x} \in \mathbb{R}^D$ and a parameter vector $\boldsymbol{w} \in \mathbb{R}^D$, the prediction rule is $f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{x}^\top \boldsymbol{w}$. If we assume model sparsity, we believe that $\sum_{d=1}^{D} \mathbb{I}\{w_d \neq 0\} \ll D$.

Our focus in this thesis is on model sparsity. As described previously, sparse models are desirable because they have smaller memory requirements, which leads to faster inference at runtime and easier interpretability. In terms of generalization performance, forcing a model to use only a few features is a way to discourage overfitting; so for high dimensional datasets such as text, it is natural to prefer sparse models.

In the context of model sparsity, there are two types of sparsity: unstructured sparsity and structured sparsity. Unstructured sparsity promotes sparse models with a very simple sparsity pattern: it prefers models with small number of features. Structured sparsity, on the other hand, promotes less trivial sparsity patterns such as grid sparsity, group sparsity, or graph-guided sparsity. This thesis focuses on how to encode linguistic

knowledge in the form of sparsity patterns in the parameter space, primarily through structured sparsity.

Model sparsity is closely related to feature selection, since setting the weight of a feature to zero is equivalent to discarding the feature. We discuss how it fits in the context of feature selection in the following.

2.1 Feature Selection

One of the longstanding goals in machine learning is to automate the feature selection process in an efficient and principled manner. In general, there are three types of methods to perform feature selection: filter-based methods, wrapper-based methods, and embedded methods.

2.1.1 Filter-based Methods

This is the simplest method to perform automatic feature selection. For every feature, we apply a heuristic to determine if the feature should be included in the model. For example, we can use count threshold to remove rare features (i.e., features that do not appear below a certain count threshold in our training data are removed); or other slightly more sophisticated thresholds such as mutual information between feature and labels. The main benefit of this approach is that it is very fast. However, filter-based methods generally ignore the learning algorithm, since the selection is done as a preprocessing step. An example of applications of filter-based methods in NLP is by Ratnaparkhi *et al.* (1994), which ignores features whose counts are less than 10 when building a part-of-speech tagger.

2.1.2 Wrapper-based Methods

While filter-based methods consider each feature in isolation, wrapper-based methods try to take into account possible interactions between features. These methods perform a search over all possible subsets of the feature set to evaluate the "goodness" of each subset. Davis *et al.* (1997) and Amaldi and Kann (1998) show that this is an NP-hard problem, so greedy methods are used in practice. A popular example of wrapper-based methods is the field induction algorithm of Della Pietra *et al.* (1997). The algorithm

iteratively adds one feature at a time based on a gain function, and reestimates model parameters with the new feature subset.

2.1.3 Embedded Methods

This thesis focuses on embedded methods for feature selection. Embedded methods incorporate feature selection into the learning problem by formulating the learning problem as a trade-off between minimizing a loss function to fit the training data and choosing a desirable model with no more features than needed. One common approach is to transform the model selection step as a penalty (regularization) function on the objective function:

$$\min_{w} \mathcal{L}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}) + \lambda \Omega(\boldsymbol{w}),$$

where $\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}$ are the input features, model parameters, and response variables respectively, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y})$ is the loss function, $\Omega(\boldsymbol{w})$ is the regularizer, and λ is a regularization hyperparameter. From the Bayesian perspective, the regularizer can also be seen as a log-prior over the weight $\boldsymbol{w}$. There are many choices for $\Omega(\boldsymbol{w})$ that induce sparsity (select relevant features). We discuss these choices in the following.

2.2 ℓ_0-"norm"

The most natural way to perform feature selection in embedded methods is by setting the penalty to the ℓ_0-"norm": $\Omega(\boldsymbol{w}) = \lim_{p \to 0} \|\boldsymbol{w}\|_0^0 = \|\boldsymbol{w}\|_0 = |\{i : w_i \neq 0\}|$. In other words, we let the penalty function be the number of non-zero features selected by the model. However, ℓ_0 is not an actual norm, since it is not a convex function (we refer to it as ℓ_0-"norm" to emphasize its connection with the ℓ_1-norm below). As a result, this makes the optimization problem harder, since the overall objective becomes non convex.

2.3 ℓ_1-norm

An alternative is to use the ℓ_1-norm as the penalty function: $\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_1 = \sum_{i=1}^{D} |w_i|$, where D is the total number of features. This technique is widely known as the lasso

(least absolute shrinkage and selection operator), first introduced by Tibshirani (1996). In order to understand why this penalty function yields sparsity, let use consider a very simple toy example, where we instantiate $\mathcal{L}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}) = \frac{1}{2}(w - y)^2$:

$$\hat{w} = \arg\min_{w} \frac{1}{2}(w - y)^2 + \lambda \|w\|_1$$

$$\hat{w} = \begin{cases} y - \lambda & \text{if } y > \lambda \\ 0 & \text{if } |y| \leq \lambda \\ y + \lambda & \text{if } y < \lambda \end{cases}$$

We can see that the solution here is the soft-thresholding operator, where given a threshold λ, we set the feature weight to 0 if $|y| \leq \lambda$ or shrink by λ otherwise. We can view the ℓ_1-norm as a convex surrogate for direct penalization of cardinality (ℓ_0). For example, for the above example, the solution with the ℓ_0 penalty is:

$$\hat{w} = \arg\min_{w} \frac{1}{2}(w - y)^2 + \lambda \|w\|_0$$

$$\hat{w} = \begin{cases} y & \text{if } |y| > \sqrt{2\lambda} \\ 0 & \text{if } |y| \leq \sqrt{2\lambda} \end{cases}$$

This is the hard-thresholding operator, where we set the feature weight to 0 if $|y| \leq \sqrt{2\lambda}$ and keep it as y otherwise. Therefore, while ℓ_0 performs hard thresholding, ℓ_1 approximates this by performing soft thresholding.

We can also get an intuition on why ℓ_1 penalty yields sparsity by approaching it from the geometrical perspective. For this example, let us instantiate $\mathcal{L}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$ for ease of illustration. Our penalized objective function is:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\Omega(\boldsymbol{w}).$$

We can rewrite this as a constrained optimization problem as follows:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 \tag{2.1}$$
$$\text{subject to } \|\boldsymbol{w}\|_1 \leq \tau,$$

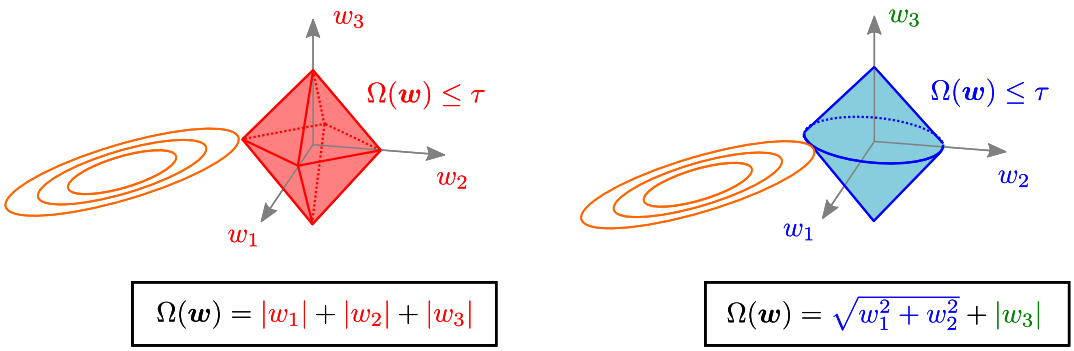


Figure 2.1: Illustrations of the solutions of the optimization problems in Eq. 2.1 and Eq. 2.2 from Martins *et al.* (2014).

where there is a one-to-one correspondence between τ and λ when the loss function is convex (which is true in our case). The solution to this constrained optimization problem intersects with a corner of the ℓ_1 norm. For example, Figure 2.1 gives an illustration in three-dimensional case. We can see that the minimizer of $\arg\min_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$ that satisfies the constraint is at a corner of the ℓ_1 ball. In other words, the optimal solution discards w_1 and w_3 (set their weights to 0) and only keeps w_2, hence yielding sparsity.

One of the main advantages of using convex surrogates such as ℓ_1 is that the resulting optimization problem is still convex, provided that $\mathcal{L}$ is convex. However, since the ℓ_1 norm is non differentiable on the axes, the optimization problem is non-smooth, so specialized optimization methods are often needed. The predominant methods to solve ℓ_1 optimization problems are described by Andrew and Gao (2007); Beck and Teboulle (2009); Xiao (2010).

From the Bayesian perspective, the ℓ_1-norm can be interpreted as zero-mean Laplacian prior on the weights: $p(w_i) \propto \exp(-\lambda|w_i|)$.

2.4 $\ell_{1,2}$-norm, $\ell_{1,\infty}$-norm

While ℓ_1-norm encourages sparse models, it promotes a very simple sparsity pattern that prefers models with small cardinality. The group lasso (Yuan and Lin, 2006) is an extension of the lasso to promote structural patterns. These patterns are defined by model designer by grouping features based on prior knowledge about the intended

patterns. The group lasso discards groups of features instead of individual features. Feature groups can be disjoint or arbitrarily overlap.

Group lasso is an instance of a bigger class of mixed-norm regularizers, often called composite absolute penalties (Zhao *et al.*, 2009). Here, we let the penalty be the mixed $\ell_{r,q}$-norm, which is defined as:

$$\Omega(\boldsymbol{w}) = \left(\sum_{g=1}^{G} \lambda_g \|\boldsymbol{w}_g\|_q^r \right)^{1/r},$$

where g indexes feature group and G is the total number of groups. Group lasso corresponds to $r = 1$. There are two choices for q that can be used to promote group sparsity: $q = 2$ and $q = \infty$. For $q = 2$, the penalty becomes $\Omega(\boldsymbol{w}) = \sum_{g=1}^{G} \lambda_g \|\boldsymbol{w}_g\|_2$, where $\|\boldsymbol{w}\|_2 = \sqrt{\sum_{i=1}^{D} |w_i|^2}$. For $q = \infty$, the penalty is $\Omega(\boldsymbol{w}) = \sum_{g=1}^{G} \lambda_g \|\boldsymbol{w}_g\|_\infty$, where $\|\boldsymbol{w}\|_\infty = \max(|w_1|, |w_2|, \ldots, |w_D|)$. In this thesis, we focus on $\ell_{1,2}$-norm, although our ideas can also be applied with $\ell_{1,\infty}$-norm. One key property of group lasso is that it promotes sparsity with respect to the groups, but it is dense inside each group.

To understand why group lasso yields group sparsity, we again can look at the geometrical interpretation of the $\ell_{1,2}$-norm. Consider the problem:

$$\begin{aligned} \hat{\boldsymbol{w}} &= \arg\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 \\ &\text{subject to } \sqrt{|w_1|^2 + |w_2|^2} + |w_3| \leq \tau, \end{aligned} \tag{2.2}$$

a group lasso problem where the group is $\{w_1, w_2\}$ and $\{w_3\}$. Figure 2.1 shows the $\ell_{1,2}$ ball constraint on the objective. The solution of this constrained optimization problem can either be at the tops of each of the two (upward and downward) cones or along the circumference of the base of the two cones. Solutions at tops corresponds to setting both w_1 and w_2 to zero (group sparsity) and setting w_3 to nonzero, whereas solutions at the base of the cone corresponds to setting w_3 to zero and setting both w_1 and w_2 to non zeros (dense within a group). In this particular example, w_3 is set to zero while w_1 and w_2 are not.

$\ell_{1,2}$ norm is a convex function, so similar to the lasso problem, the group lasso optimization problem is also convex and non smooth. Efficient optimization methods

for group lasso problems typically depend on the group structures (Jacob *et al.*, 2009; Jenatton *et al.*, 2011b; Chen *et al.*, 2011; Martins *et al.*, 2011a; Qin and Goldfarb, 2012; Yuan *et al.*, 2013). In this thesis, we propose two optimization methods to solve group lasso optimization problems efficiently for linguistically-motivated grouping structures.

From the Bayesian perspective, the **non-overlapping** $\ell_{1,2}$-norm can be interpreted as two-level hierarchical Bayes model. For each group g, we draw a group-specific variance $p(\sigma_g) = \exp(\lambda_g^2/2)$, and each feature weight in group g is drawn from $p(\boldsymbol{w}_g) \sim \mathcal{N}(0, \sigma_g \boldsymbol{I})$. Overlapping group lasso does not have a probabilistic interpretation as a prior over weights.

Chapter 3

Structured Sparsity in Temporal Models of Text

When learning from streams of data to make predictions in the future, how should we handle the timestamp associated with each instance? Ignoring timestamps and assuming data are i.i.d. is scalable but risks distracting a model with irrelevant "ancient history." On the other hand, using only the most recent portion of the data risks overfitting to current trends and missing important time-insensitive effects. Here, we seek a general approach to learning model parameters that are overall sparse, but that adapt to variation in how different effects change over time.

Our approach is a prior over parameters of an exponential family (e.g., coefficients in linear or logistic regression). We assume that parameter values shift at each timestep, with correlation between adjacent timesteps captured using a multivariate normal distribution whose precision matrix is restricted to a tridiagonal structure. We (approximately) marginalize the (co)variance parameters of this normal distribution using a Jeffreys prior, resulting in a model that allows smooth variation over time while encouraging overall sparsity in the parameters. (The parameters themselves are not given a fully Bayesian treatment.)

We demonstrate the usefulness of our model on two natural language processing tasks, showing gains over alternative approaches. The first is a text regression problem in which an economic variable (volatility of returns) is forecast from financial reports (Kogan *et al.*, 2009). The second forecasts text by constructing a language model that

conditions on highly time-dependent economic variables. This chapter is based on materials from Yogatama *et al.* (2013).

3.1 Notation

We assume data of the form $\{(x_n, y_n)\}_{n=1}^N$, where each x_n includes a timestamp denoted $t \in \{1, \ldots, T\}$.[1] The aim is to learn a predictor that maps input x_{N+1}, assumed to be at timestep T, to output y_{N+1}. In the probabilistic setting we adopt here, the prediction is MAP inference over random variable Y given $X = x$ and a model parameterized by $\boldsymbol{w} \in \mathbb{R}^I$. Learning is parameter estimation to solve:

$$\arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}) + \overbrace{\sum_{n=1}^{N} \log \underbrace{p(y_n \mid x_n, \boldsymbol{w})}_{\text{link}^{-1}(\boldsymbol{f}(x)^\top \boldsymbol{w})}}^{\mathcal{L}(\boldsymbol{w})} \tag{3.1}$$

The focus of the paper is on the prior distribution $p(\boldsymbol{w})$. Throughout, we will denote the task-specific log-likelihood (second term) by $\mathcal{L}(\boldsymbol{w})$ and assume a generalized linear model such that a feature vector function $\boldsymbol{f}$ maps inputs x into $\mathbb{R}^I$ and $\boldsymbol{f}(x)^\top \boldsymbol{w}$ is "linked" to the distribution over Y using, e.g., a logit or identity. We assume T discrete timesteps.

3.2 Time-Series Prior

Our time-series prior draws inspiration from the probabilistic interpretation of the sparsity-inducing lasso (Tibshirani, 1996) and group lasso (Yuan and Lin, 2006). In non-overlapping group lasso, features are divided into groups, and the coefficients within each group m are drawn according to:

1. Variance $\sigma_m^2 \sim$ an exponential distribution.[2]
2. $\boldsymbol{w}_m \sim \text{Normal}(\mathbf{0}, \sigma_m^2 \mathbf{I})$.

[1] In this work we assume timestamps are discretized.

[2] The exponential distribution can be replaced by the (improper) Jeffreys prior, although then the familiar Laplace distribution interpretation no longer holds (Figueiredo, 2002).

We seek a prior that lets each coefficient vary smoothly over time. A high-level intuition of our prior is that we create copies of $\boldsymbol{w}$, one at each timestep: $\langle \boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots, \boldsymbol{w}^{(T)} \rangle$. For each feature i, let the sequence $\langle w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(T)} \rangle$ form a group, denoted $\boldsymbol{w}_i$. Group lasso does not view coefficients in a group as explicitly correlated; they are independent given the variance parameter. Given the sequential structure of $\boldsymbol{w}_i$, we replace the covariance matrix $\sigma^2\mathbf{I}$ to capture autocorrelation. Specifically, we assume the vector $\boldsymbol{w}_i$ is drawn from a multivariate normal distribution with mean zero and a $T \times T$ precision matrix $\boldsymbol{\Lambda}$ with the following tridiagonal form:[3]

$$\boldsymbol{\Lambda} = \frac{1}{\lambda}\mathbf{A} = \frac{1}{\lambda}\begin{bmatrix} 1 & \alpha & 0 & 0 & \ldots \\ \alpha & 1 & \alpha & 0 & \ldots \\ 0 & \alpha & 1 & \alpha & \ldots \\ 0 & 0 & \alpha & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.2}$$

$\lambda \geq 0$ is a scalar multiplier whose role is to control sparsity in the coefficients, while α dictates the degree of correlation between coefficients in adjacent timesteps (autocorrelation). Importantly, α and λ (and hence $\mathbf{A}$ and $\boldsymbol{\Lambda}$) are allowed to be different for each group i.

We need to ensure that $\mathbf{A}$ is positive definite. Fortunately, it is easy to show that for $\alpha \in (-0.5, 0.5)$, the resulting $\mathbf{A}$ is positive definite.

Proof sketch. To show this, since $\mathbf{A}$ is a symmetric matrix, we verify that each of its principal minors have strictly positive determinants. The principal minors of $\mathbf{A}$ are uniform tridiagonal symmetric matrices, and the determinant of a uniform tridiagonal $N \times N$ matrix can be written as $\prod_{n=1}^{N} \left\{ 1 + 2\alpha \cos\left(\frac{(n+1)\pi}{N+1} \right) \right\}$ (see, e.g., Volpi, 2003 for the proof). Since $\cos(x) \in [-1, 1]$, if $\alpha \in (-0.5, 0.5)$, the determinant is always positive. Therefore, $\mathbf{A}$ is always p.d. for $\alpha \in (-0.5, 0.5)$. □

In this work, we use a tridiagonal matrix to capture temporal dependencies due to its simplicity (see §3.2.2 for discussions on scalability). Future work could consider a more general matrix form that would give longer-term dependencies.

[3] We suppress the subscript i for this discussion; each feature i has its own $\boldsymbol{\Lambda}_i$.

3.2.1 Generative Model

Our generative model for the group of coefficients $\boldsymbol{w}_i = \langle w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(T)} \rangle$ is given by:

1. $\lambda_i \sim$ an improper Jeffreys prior ($p(\lambda) \propto \lambda^{-1}$).
2. $\alpha_i \sim$ a truncated exponential prior with parameter τ. This distribution forces α_i to fall in $(-C, 0]$, so that $\mathbf{A}_i$ is p.d. and autocorrelations are always positive:

$$p(\alpha \mid \tau) = \frac{\tau \exp(-\tau(\alpha + C))\mathbf{1}\{-C < \alpha \leq 0\}}{(1 - \exp(-\tau C))}. \tag{3.3}$$

   We fix $C = \frac{1}{2} - 10^{-5}$.
3. $\boldsymbol{w}_i \sim \text{Normal}(\mathbf{0}, \boldsymbol{\Lambda}_i^{-1})$, with the precision matrix $\boldsymbol{\Lambda}_i$ as defined in Eq. 3.2.

During estimation of $\boldsymbol{w}$, each λ_i and α_i are marginalized, giving a sparse and adaptive estimate for $\boldsymbol{w}$.

3.2.2 Scalability

Our design choice of the precision matrix $\boldsymbol{\Lambda}_i$ is driven by scalability concerns. Instead of using, e.g., a random draw from a Wishart distribution, we specify the structure of the precision matrix as a tridiagonal matrix. Since the precision matrix introduces dependencies only between coefficients in adjacent timesteps (first-order dependencies), it allows the prior to scale to fine-grained timesteps more efficiently. Let N denote the number of training instances, I the number of base features, and T the number of timesteps. A single pass of our variational algorithm (discussed in §3.3) has runtime $\mathcal{O}(I(N + T))$ and space requirement $\mathcal{O}(I(N + T))$, instead of $\mathcal{O}(I(N + T^2))$ for both if each $\boldsymbol{\Lambda}_i$ is drawn from a Wishart distribution. This can make a big difference for applications with large numbers of features (I). Additionally, we choose the off-diagonal entries to be uniform, so we only need one α_i for each base feature. This design choice restricts the expressive power of the prior but still permits flexibility in adapting to trends for different coefficients, as we will see. The prior encourages sparsity at the group level, essentially performing feature selection: some feature coefficients $\boldsymbol{w}_i$ may

be driven to zero across all timesteps, while others will be allowed to vary over time, with an expectation of smooth changes.

Note that this model introduces only one hyperparameter, τ, since we marginalize $\boldsymbol{\alpha} = \langle \alpha_1, \ldots, \alpha_I \rangle$ and $\boldsymbol{\lambda} = \langle \lambda_1, \ldots, \lambda_I \rangle$.

3.3 Learning and Inference

We marginalize $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ and obtain a maximum *a posteriori* estimate for $\boldsymbol{w}$, which includes a coefficient for each base feature i at each timestep t. Specifically, the quantity that we need to maximize is:

$$\mathcal{L}(\boldsymbol{w}) + \sum_{i=1}^{I} \int d\alpha_i \int d\lambda_i \log p(\boldsymbol{w}_i \mid \alpha_i, \lambda_i) + \log p(\alpha_i \mid \tau) + \log p(\lambda_i)$$

Exact inference in this model is intractable. We use mean-field variational inference to derive a lower bound on the above log-likelihood function. We then apply a standard optimization technique to jointly optimize the variational parameters and the coefficients $\boldsymbol{w}$.

We introduce fully factored variational distributions for each λ_i and α_i. For λ_i, we use a Gamma distribution with parameters a_i, b_i as our variational distribution:

$$q_i(\lambda_i \mid a_i, b_i) = \frac{\lambda_i^{a_i - 1} \exp(-\lambda_i / b_i)}{b_i^{a_i} \Gamma(a_i)}$$

Therefore, we have $\mathbb{E}_{q_i}[\lambda_i] = a_i b_i$, $\mathbb{E}_{q_i}[\lambda_i^{-1}] = ((a_i - 1) b_i)^{-1}$, and $\mathbb{E}_{q_i}[\log \lambda_i] = \Psi(a_i) + \log b_i$ (Ψ is the digamma function).

For α_i, we choose the form of our variational distribution to be the same truncated exponential distribution as its prior, with parameter κ_i, denoting this distribution $q_i(\alpha_i \mid \kappa_i)$. We have

$$\begin{aligned} \mathbb{E}_{q_i}[\alpha_i] &= \int_{-C}^{0} \alpha_i \frac{\kappa_i \exp(-\kappa_i(\alpha_i + C))}{1 - \exp(-\kappa_i C)} d\alpha_i \\ &= \frac{1}{\kappa_i} - \frac{C}{1 - \exp(-\kappa_i C)} \end{aligned} \tag{3.4}$$

We let q denote the set of all variational distributions over $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$.

$$
\begin{aligned}
B(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\kappa}, \boldsymbol{w}) \quad \propto \quad & \mathcal{L}(\boldsymbol{w}) + \sum_{i=1}^{I} \left\{ \frac{1}{2}(-T\mathbb{E}_q[\log \lambda_i] \boxed{-\mathbb{E}_q[\log \det \mathbf{A}_i^{-1}]}) - \mathbb{E}_q[\lambda_i^{-1}] \frac{1}{2} \boldsymbol{w}_i^\top \mathbb{E}_q[\mathbf{A}_i] \boldsymbol{w}_i \right\} \\
& + \sum_{i=1}^{I} \left\{ -(\mathbb{E}_q[\alpha_i] + C)\tau - \mathbb{E}_q[\log \lambda_i] \right\} \\
& - \sum_{i=1}^{I} \left\{ (a_i - 1)\mathbb{E}_q[\log \lambda_i] - \frac{\mathbb{E}_q[\lambda_i]}{b_i} - \log \Gamma(a_i) - a_i \log b_i \right\} \\
& - \sum_{i=1}^{I} \left\{ \log \kappa_i - \kappa_i(\mathbb{E}_q[\alpha_i] + C) - \log(1 - \exp(-\kappa_i C)) \right\}
\end{aligned}
$$

Figure 3.1: The variational bound on Equation 3.1 that is maximized to learn $\boldsymbol{w}$. The boxed expression is further bounded by $-\log \det \mathbb{E}_q[\mathbf{A}_i]$ using Jensen's inequality, giving a bound we denote by B'.

The variational bound B that we seek to maximize is given in Figure 3.1. Our learning algorithm involves optimizing with respect to variational parameters $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{\kappa}$, and the coefficients $\boldsymbol{w}$. We employ the L-BFGS quasi-Newton method (Liu and Nocedal, 1989), for which we need to compute the gradient of B. We turn next to each part of this gradient.

3.3.1 Coefficients w

For $1 < t < T$, the first derivative with respect to time-specific coefficient $w_i^{(t)}$ is:

$$
\frac{\partial B}{\partial w_i^{(t)}} = \frac{\partial \mathcal{L}}{\partial w_i^{(t)}} - \frac{1}{2}\mathbb{E}[\lambda_i^{-1}] \left(\mathbb{E}[\alpha_i](w_i^{(t-1)} + w_i^{(t+1)}) + 2w_i^{(t)} \right) \tag{3.5}
$$

We can interpret the first derivative as including a penalty scaled by $\mathbb{E}[\lambda_i^{-1}]$. We rewrite this penalty as:

$$
\mathbb{E}[\lambda_i^{-1}] \left(\; (1 - \mathbb{E}[\alpha_i]) \cdot 2w_i^{(t)} + \mathbb{E}[\alpha_i] \cdot (w_i^{(t)} - w_i^{(t-1)}) + \mathbb{E}[\alpha_i] \cdot (w_i^{(t)} - w_i^{(t+1)}) \right)
$$

This form makes it clear that the penalty depends on $w_i^{(t-1)}$ and $w_i^{(t+1)}$, penalizing the difference between $w_i^{(t)}$ and these time-adjacent coefficients proportional to $\mathbb{E}[\alpha_i]$.

The form bears strong similarity to the first derivative of the time-series (log-)prior introduced in Yogatama *et al.* (2011), which depends on fixed, global hyperparameters

analogous to our α and λ. Specifically, the gradient of the regularizer in Yogatama *et al.* (2011) is:

$$2\lambda w_i^{(t)} + 2\lambda\alpha(w_i^{(t)} - w_i^{(t-1)}) + 2\lambda\alpha(w_i^{(t)} - w_i^{(t+1)}),$$

for a fixed λ and α. Because our approach does not require us to specify scalars playing the roles of "$\mathbb{E}[\lambda_i^{-1}]$" and "$\mathbb{E}[\alpha_i]$" in advance, it is possible for each feature to have its own autocorrelation. Obtaining the same effect in their model would require careful tuning of $\mathcal{O}(I)$ hyperparameters, which is not practical.

It also has some similarities to the fused lasso penalty (Tibshirani *et al.*, 2005), which is intended to encourage sparsity in the differences between features coefficients across timesteps. Our prior, on the other hand, encourages smoothness in the differences, with additional sparsity at the feature level.

3.3.2 Variational Parameters for $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$

Recall that the variational distribution for λ_i is a Gamma distribution with parameters a_i and b_i.

Precision matrix scalar $\boldsymbol{\lambda}$. The first derivative for variational parameters $\boldsymbol{a}$ is easy to compute:

$$\frac{\partial B}{\partial a_i} = (-\frac{T}{2} - a_i)\Psi_1(a_i) + \frac{\boldsymbol{w}_i^\top \mathbb{E}[\mathbf{A}_i]\boldsymbol{w}_i}{2b_i(a_i-1)^2} + 1 \tag{3.6}$$

where Ψ_1 is the trigamma function. We can solve for $\boldsymbol{b}$ in closed form given the other free variables:

$$b_i = \frac{\boldsymbol{w}_i^\top \mathbb{E}[\mathbf{A}_i]\boldsymbol{w}_i}{(a_i-1)T} \tag{3.7}$$

We therefore treat $\boldsymbol{b}$ as a function of $\boldsymbol{a}$, $\boldsymbol{\kappa}$, and $\boldsymbol{w}$ in optimization.

Off-diagonal entries $\boldsymbol{\alpha}$. First, notice that using Jensen's inequality: $\mathbb{E}[\log\det\mathbf{A}_i^{-1}] = \mathbb{E}[-\log\det\mathbf{A}_i] \geq -\log\det\mathbb{E}[\mathbf{A}_i]$ due to the fact that $-\log\det\mathbf{A}_i$ is a convex function. Furthermore, for a uniform symmetric tridiagonal matrix like $\mathbf{A}_i$, the log determinant

can be computed in closed form as follows (Volpi, 2003):

$$\begin{aligned}\log\det\mathbb{E}[\mathbf{A}_i] &= \log\left(\prod_{t=1}^{T} 1 + 2\mathbb{E}[\alpha_i]\cos\left(\frac{(t+1)\pi}{T+1}\right)\right)\\ &= \sum_{t=1}^{T}\log\left(1 + 2\mathbb{E}[\alpha_i]\cos\left(\frac{(t+1)\pi}{T+1}\right)\right)\end{aligned}$$

We therefore maximize a lower bound on B, making use of the above to calculate first derivatives with respect to κ_i:

$$\begin{aligned}\frac{\partial B'}{\partial \kappa_i} =& -\tau\frac{\partial\mathbb{E}[\alpha_i]}{\partial\kappa_i} - \frac{1}{\kappa_i} + C + \mathbb{E}[\alpha_i] + \frac{\partial\mathbb{E}[\alpha_i]}{\partial\kappa_i}\kappa_i\\ &+ \frac{C\exp(-C\kappa_i)}{1-\exp(-C\kappa_i)} + \frac{1}{2}\frac{\partial\log\det\mathbb{E}[\mathbf{A}_i]}{\partial\kappa_i}\\ &- \frac{1}{2}\mathbb{E}[\lambda_i^{-1}]\frac{\partial\boldsymbol{w}_i^\top\mathbb{E}[\mathbf{A}_i]\boldsymbol{w}_i}{\partial\kappa_i}\end{aligned}$$

The partial derivatives $\frac{\partial\mathbb{E}[\alpha_i]}{\partial\kappa_i}$, $\frac{\partial\log\det\mathbb{E}[\mathbf{A}_i]}{\partial\kappa_i}$, and $\frac{\partial\boldsymbol{w}_i^\top\mathbb{E}[\mathbf{A}_i]\boldsymbol{w}_i}{\partial\kappa_i}$ are easy to compute.

3.3.3 Implementation details

A well-known property of numerical optimizers like the one we use (L-BFGS; Liu and Nocedal, 1989) is the failure to reach optimal values exactly at zero. Although theoretically strongly sparse, our prior only produces weak sparsity in practice. Future work might consider a more principled proximal-gradient algorithm to obtain strong sparsity (Bach *et al.*, 2011; Liu and Ye, 2010; Duchi and Singer, 2009).

If we expect feature coefficients at specific timesteps to be sparse as well, it is straightforward to incorporate additional terms in the objective function that encode this prior belief (analogous to an extension from group lasso to *sparse* group lasso). For the tasks we consider in our experiments, we found that it does not substantially improve the overall performance. Therefore, we keep the simpler bound given in Figure 3.1.

3.4 Experiments

We report two sets of experiments, one with a continuous y, the other a language modeling application where y is text. Each timestep in our experiments is one year.

3.4.1 Baselines

On both tasks, we compare our approach to a range of baselines. Since this is a forecasting task, at each test year, we only used training examples that come from earlier years. Our baselines vary in how they use this earlier data and in how they regularize.

- **ridge-one**: ridge regression (Hoerl and Kennard, 1970), trained on only examples from the year prior to the test data (e.g., for the 2002 task, train on examples from 2001)
- **ridge-all**: ridge regression trained on the full set of past examples (e.g., for the 2002 task, train on examples from 1996–2001)
- **ridge-ts**: the non-adaptive time-series ridge model of Yogatama *et al.* (2011)
- **lasso-one**: lasso regression (Tibshirani, 1996), trained on only examples from the year prior to the test data[4]
- **lasso-all**: lasso regression trained on the full set of past examples

In all cases, we tuned hyperparameters on a development data. Note that, of the above baselines, only **ridge-ts** replicates the coefficients at different timesteps (i.e., IT parameters); the others have only I time-insensitive coefficients.

The model with our prior always uses all training examples that are available up to the test year (this is equivalent to a sliding window of size infinity). Like **ridge-ts** our model trusts more recent data more, allowing coefficients farther in the past to drift farther away from those most relevant for prediction at time $T + 1$. Our model, however, adapts the "drift" of each coefficient separately rather than setting a global hyperparameter.

3.4.2 Forecasting Risk from Text

In the first experiment, we apply our prior to a forecasting task. We consider the task of predicting volatility of stock returns from financial reports of publicly-traded companies, similar to Kogan *et al.* (2009).

[4]Anonymous (personal communication) has established the superiority of the lasso to the support vector regression method of Kogan *et al.* (2009) on this dataset; lasso is a strong baseline for this problem.

Table 3.1: MSE on the 10-K dataset (various test sets). The first test year (2002) was used as our development data. Our model uses the sparse adaptive prior described in §3.2. The overall differences between our model and all competing models are statistically significant (Wilcoxon signed-rank test, $p < 0.01$).

year	# examples	ridge-one	ridge-all	ridge-ts	lasso-one	lasso-all	our model
2002(dev)	2,845	0.182	0.176	0.171	0.165	**0.156**	0.158
2003	3,611	0.185	0.173	0.171	**0.164**	0.176	0.164
2004	3,558	0.125	0.137	0.129	0.116	0.119	**0.113**
2005	3,474	0.135	0.133	0.136	0.124	0.124	**0.122**
overall	13,488	0.155	0.154	0.151	0.141	0.143	**0.139**

In finance, *volatility* refers to a measure of variation in a quantity over time; for stock returns, it is measured using the standard deviation during a fixed period (here, one year). Volatility is used as a measure of financial risk. Consider a linear regression model for predicting the log volatility[5] of a stock from a set of features (see the dataset paragraph below for a complete description of our features). We can interpret a linear regression model probabilistically as drawing $y \in \mathbb{R}$ from a normal distribution with $\boldsymbol{w}^\top \boldsymbol{f}(x)$ as the mean of the normal. Therefore, in this experiment: $\mathcal{L}(\boldsymbol{w}) = -\sum_{t=1}^{T}\sum_{i=1}^{N_t}(y_i^{(t)} - \boldsymbol{w}^{(t)\top}\boldsymbol{f}(x_i^{(t)}))^2$. We apply the time-series prior to the feature coefficients $\boldsymbol{w}$. When making a prediction for the test data, we use $\boldsymbol{w}^{(T)}$, the set of feature coefficients for the last timestep in the training data.

Dataset We used a collection of Securities Exchange Commission-mandated annual reports from 10,492 publicly traded companies in the U.S. There are 27,159 reports over a period of ten years from 1996–2005 in the corpus. These reports are known as "Form 10-K."[6] For the feature set, we downcased and tokenized the texts and selected the 101st–10,101st most frequent words as binary features. The feature set was kept the same across experiments for all models. It is widely known in the financial community that the past history of volatility of stock returns is a good indicator of the future volatility. Therefore, we also included the log volatility of the stocks twelve months prior to the report as a feature. Our response variable y is the log volatility of stock returns over a

[5]Similar to Kogan *et al.* (2009) and as also the standard practice in finance, we perform a log transformation, since log volatilities are typically close to normally distributed.

[6]See Kogan *et al.* (2009) for a complete description of the dataset; it is available at `http://www.ark.cs.cmu.edu/10K`.

period of twelve months after the report is published.

Results The first test year (i.e., 2002) was used as our development data for hyper-parameter tuning (τ was selected to be 1.0). We initialized all the feature coefficients by the coefficients from training a lasso regression on the last year of the training data (**lasso-one**). Table 5.3 provides a summary of experimental results. We report the results in mean squared error on the test set: $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$, where y_i is the true response for instance i and $\hat{y}_i$ is the predicted response.

Our model consistently outperformed ridge variants, including the one with a time-series penalty (Yogatama *et al.*, 2011). It also outperformed the lasso variants without any time-series penalty, on average and in three out of four test sets apiece.

One of the major challenges in working with time-series data is to choose the right window size, in which the data is still relevant to current predictions. Our model automates this process with a Bayesian treatment of the strength of each feature coefficient's autocorrelation. The results indicate that our model was able to learn when to trust a longer history of training data, and when to trust a shorter history of training data, demonstrating the adaptiveness of our prior. Figure 3.2 shows the distribution of the expected values of the autocorrelation paramaters under the variational distributions $\mathbb{E}_{q_i}[\alpha_i]$ for 10,002 features, learned by our model from the last run (test year 2005).

In future work, an empirical Bayesian treatment of the hyperprior τ, fitting it to improve the variational bound, might lead to further improvements.

3.4.3 Text Modeling in Context

In the second experiment, we consider a hard task of modeling a collection of texts over time conditioned on economic measurements. The goal is to predict the probability of words appearing in a document, based on the "state of the world" at the time the document was authored. Given a set of macroeconomic variables in the U.S. (e.g., unemployment rate, inflation rate, average housing prices, etc.), we want to predict what kind of texts will be produced at a specific timestep. These documents can be written by either the government or publicly-traded companies directly or indirectly affected by the current economic situation.

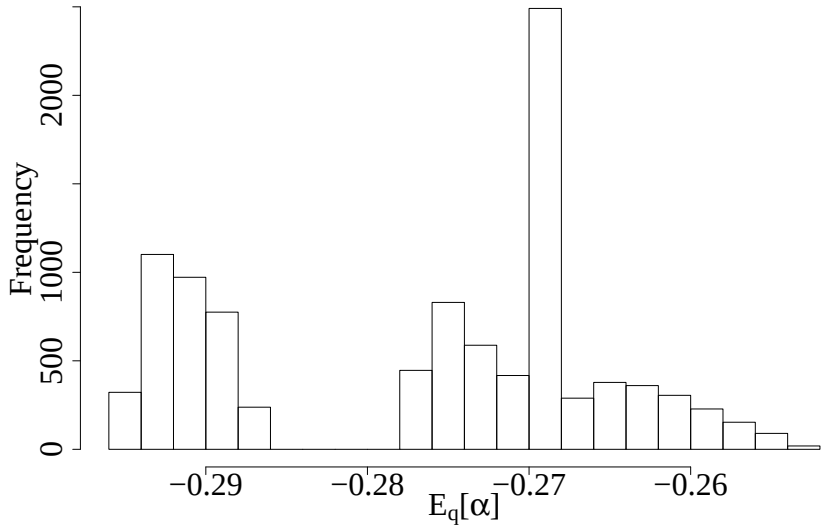


Figure 3.2: The distribution of expected values of the autocorrelation paramaters under the variational distributions $\mathbb{E}_{q_i}[\alpha_i]$ for 10,002 features used in our experiments (10,000 unigram features, the previous year log volatility feature, and a bias feature).

Model Our text model is a sparse additive generative model (SAGE; Eisenstein *et al.*, 2011b). In SAGE, there is a background lexical distribution that is perturbed additively in the log-space. When the effects are due to a (sole) feature $f(x)$, the probability of a word is:

$$p(w \mid \boldsymbol{\theta}, \boldsymbol{w}, x) = \frac{\exp(\theta_w + w_w f(x))}{\sum_{w' \in V} \exp(\theta_{w'} + w_{w'} f(x))}$$

where V is the vocabulary, $\boldsymbol{\theta}$ (always observed) is the vector of background log-frequencies of words in the corpus, $f(x)$ (observed) is the feature derived from the context x, and $\boldsymbol{w}$ is the feature-specific deviation.

Notice that the formulation above is easily extended to multiple effects with coefficients $\boldsymbol{w}$. In our experiment, we have 117 effects (features), each with its own $\boldsymbol{w}_i$. The first 50 correspond to U.S. states, plus an additional feature for the entire U.S., and they are observed for each text since each text is associated with a known set of states (discussed below). We assume that texts that are generated in different states have distinct characteristics; for each state, we have a binary indicator feature. The other 66 features depend on observed macroeconomics variables at each timestep (e.g., unemployment

rate, inflation rate, house price index, etc.). Given an economic state of the world, we hypothesize that there are certain words that are more likely to be used, and each economic variable has its own (sparse) deviation from the background word frequencies. The generative story for a word at timestep t associated with (observed) features $\boldsymbol{f}(x^{(t)})$ is:

- Given observed real-world observed variables $x^{(t)}$, draw word w from a multinomial distribution $p(w \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{w}^{(t)}, x^{(t)}) \propto \exp(\theta_w^{(t)} + \boldsymbol{w}_w^{(t)\top} \boldsymbol{f}(x^{(t)}))$.

Our $\mathcal{L}(\boldsymbol{w})$ is simply the negative log-loss function commonly used in multiclass logistic regression:

$$\mathcal{L}(\boldsymbol{w}) = \sum_{t=1}^{T} \sum_{i=1}^{N_t} \log p(\boldsymbol{w}_i^{(t)} \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{w}^{(t)}, x_i^{(t)}).$$

We apply our time-series prior from §3.2 to $\boldsymbol{w}$. $\boldsymbol{\theta}^{(t)}$ is fixed to be the log frequencies of words at timestep t. For a single feature, coefficients over time for different classes (words) are assumed to be generated from the same prior.

Dataset There is a great deal of text that is produced to describe current macroeconomic events. We conjecture that the connection between the economy and the text will have temporal dependencies (e.g., the amount of discussion about housing or oil prices might vary over time). We use three sources of text commentary on the economy. The first is a subset of the 10-K reports we used in §3.4.2. We selected the 10-K reports of 200 companies chosen randomly from the top quintile of size (measured by beginning-of-sample market capitalization). This gives us a sample of the largest U.S. companies. Each report is associated with the state in which the company's head office is located. Our next two data sources come from the Federal Reserve System, the primary body responsible for monetary policy in the U.S.[7] The Federal Open Market Committee (FOMC) meets roughly eight times per year to discuss economic conditions and set monetary policy. Prior to each meeting, each of the twelve regional banks writes an informal "anecdotal" description of economic activity in their region as well as a national summary. This "Beige Book" is akin to a blog of economic activity released

[7] For an overview of the Federal Reserve System, see the Federal Reserve's "Purpose and Functions" document at `http://www.federalreserve.gov/pf/pf.htm`.

prior to each meeting. Each FOMC meeting also produces a transcript of the discussion. For our experiments here, we focus on text from 1996–2006.[8] As a result, we have 2,075 documents in the final corpus, consisting of 842 documents of the 10-K reports, 89 documents of the FOMC meeting transcripts, and 1,144 documents of the Beige Book summaries.

We use the 501st–5,501st most frequent words in the dataset. We associated the FOMC meeting transcripts with all states. The "Beige Book" texts were produced by the Federal Reserve Banks. There are twelve Federal Reserve Banks in the United States, each serving a collection of states. We associated texts from a Federal Reserve Bank with the states that it serves.

Table 3.2: Negative log-likelihood of the documents on various test sets (lower is better). The first test year (2003) was used as our development data. Our model uses the sparse adaptive prior in §3.2.

year	# tokens ($\times 10^6$)	ridge-one ($\times 10^3$)	ridge-all ($\times 10^3$)	ridge-ts ($\times 10^3$)	lasso-one ($\times 10^3$)	lasso-all ($\times 10^3$)	our model ($\times 10^3$)
2003(dev)	1.1	2,736	2,765	2,735	2,736	2,765	**2,735**
2004	1.5	2,975	3,004	2,975	2,975	3,004	**2,974**
2005	1.9	2,999	3,027	2,997	2,998	3,027	**2,997**
2006	2.3	2,916	2,922	2,913	2,912	2,922	**2,912**
overall	6.8	11,626	11,718	11,619	11,620	11,718	**11,618**

Quantitative U.S. macroeconomic data was obtained from the Federal Reserve Bank of St. Louis data repository ("FRED"). We used standard measures of economic activity focusing on output (GDP), employment, and specific markets (e.g., housing).[9] We use equity market returns for the U.S. market as a whole and various industry and characteristic portfolios.[10] They are used as $f(x)$ in our model; in addition to state indicator variables, there are 66 macroeconomic variables in total.

We compare our model to the baselines in §3.4.1. The lasso variants are analogous to the original formulation of SAGE (Eisenstein *et al.*, 2011b), except that our model directly conditions on macroeconomic variables instead of a Dirichlet-multinomial compound.

[8] All the text is freely available at `http://www.federalreserve.gov`. The Beige Book is released to the public prior to each meeting. The transcripts are released five years after the meetings.

[9] For growing output series, like GDP, we calculate growth rates as log differences.

[10] Returns are monthly, excess of the risk-free rate, and continuously compounded. The data are from CRSP and are available for these portfolios at `http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`.

Results We score models by computing the negative log-likelihood on the test dataset:[11]

$$-\sum_{i=1}^{N} \log p(\boldsymbol{w}_i^{(T+1)} \mid \boldsymbol{\theta}^{(T)}, \boldsymbol{w}^{(T)}, x_i^{(T+1)}).$$

We initialized all the feature coefficients by the coefficients by training a lasso regression on the last year of the training data (**lasso-one**). The first test year (i.e., 2003) was used as our development data for hyperparameter tuning (τ was selected to be .001). Table 3.2 shows the results for the six models we compared. Similar to the forecasting experiments, at each test year, we trained only on documents from earlier years. When we collapsed all the training data and ignored the temporal dimension (ridge-all and lasso-all), the background log-frequencies $\boldsymbol{\theta}^{(t)}$ are computed using the entire training data, which is different compared to the background log-frequencies for only the last timestep of the training data. Our model outperformed all ridge and lasso variants, including the one with a time-series penalty (Yogatama *et al.*, 2011), in terms of negative log-likelihood on unseen dataset.

In addition to improving predictive accuracy, the prior also allows us to discover trends in the feature coefficients and gain insight. We manually examined the model from the last run (test year 2006). Examples of temporal trends learned by our model are shown in Figure 3.3. The plot illustrates feature coefficients for words that contain the string `employ`. For comparison, we also included the percentage of unemployment rate in the U.S. (which was used as one of the features $f(x)$), scaled by 10^{-16} to fit into the plot. We can see that there is a correlation between feature coefficients for the word `unemployment` and the actual unemployment rate. On the other hand, the correlations are less evident for other words.

3.4.4 Discussion

Our model is related to autoregressive integrated moving average approaches to time-series data (Box *et al.*, 2008), but we never have access to *direct* observations of the time-series. Instead, we observe data (x and y) assumed to have been sampled using time-series-generated variables as *coefficients* ($\boldsymbol{w}$). During learning, we therefore use probabilistic inference to reason about the variables at all timesteps together. In §3.3,

[11] Out-of-vocabulary items are ignored.

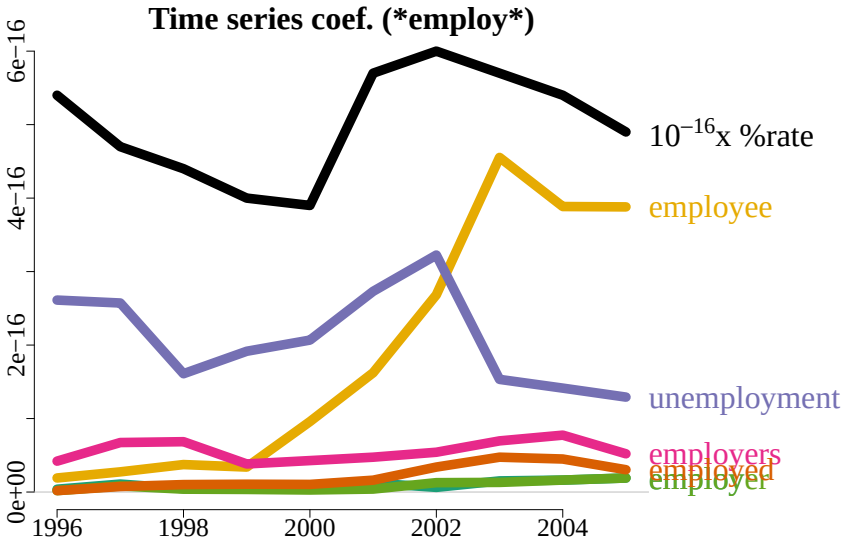


Figure 3.3: Temporal trends learned by our model for the words that contains `employ` in our dataset, as well as the actual unemployment rate (scaled by 10^{-16} for ease of reading). The y-axis denotes coefficients and the x-axis is years. See the text for explanation.

we describe a scalable variational inference algorithm for inferring coefficients at all timesteps, enabling prediction of future data and inspection of trends.

We follow Yogatama *et al.* (2011) in creating time-specific copies of the base coefficients, so that $\boldsymbol{w} = \langle \boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots, \boldsymbol{w}^{(T)} \rangle$. As a prior over $\boldsymbol{w}$, they used a multivariate Gaussian imposing non-zero covariance between each $w_i^{(t)}$ and its time-adjacent copies $w_i^{(t-1)}$ and $w_i^{(t+1)}$. The strength of that covariance was set for each base feature by a global hyperparameter, which was tuned on held-out development data along with the global variance hyperparameter. Yogatama et al.'s model can be obtained from ours by fixing the same α and λ for all features i. Our approach differs in that (i) we marginalize the hyperparameters, (ii) we allow each coefficient its own autocorrelation, and (iii) we encourage sparsity.

There are many related Bayesian approaches for time-varying model parameters (Belmonte *et al.*, 2012; Nakajima and West, 2012; Caron *et al.*, 2012), as well as work on time-varying signal estimation (Angelosante and Giannakis, 2009; Angelosante *et al.*, 2009; Charles and Rozell, 2012). Each provides a different probabilistic interpretation of parameter generation. Our model has a distinctive generative story in that correlations between parameters of successive timesteps are encoded in a precision matrix.

Additionally, unlike these fully Bayesian approaches that infer full posterior distributions, we only obtain posterior mode estimates of coefficients, which has computational advantages at prediction time (e.g., straightforward MAP inference and sparsity) and interpretability of $\boldsymbol{w}$.

As noted, our grouping together of each feature's instantiations at all timesteps, $\langle w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(T)} \rangle$ and seeking sparsity, bears clear similarity to group lasso (Yuan and Lin, 2006) and our work in Chapter 4 and Chapter 5, which encourages whole groups of coefficients to collectively go to zero. A probabilistic interpretation for lasso as a two level exponential-normal distribution that generalizes to (non-overlapping) group lasso was introduced by Figueiredo (2002). He also showed that the exponential distribution prior can be replaced with an improper Jeffreys prior for a parameter-free model, a step we follow as well. Our model is also related to the fused lasso (Tibshirani *et al.*, 2005), which penalizes a loss function by the ℓ_1-norm of the coefficients and their differences. Our prior has a more clear probabilistic interpretation and adapts the degree autocorrelation for each coefficient, based on the data. Future work could try to evaluate how much adaptiveness and sparsity help in isolation in order to seek ways to improve each component independently.

Zhang and Yeung (2010) proposed a regularization method using a matrix-variate normal distribution prior to model task relationships in multitask learning. If we consider timesteps as tasks, the technique resembles our regularizer. Their method jointly optimizes the covariance matrix with the feature coefficients; we choose a Bayesian treatment and encode our prior belief to the (inverse) covariance matrix, while still allowing the learned feature coefficients to modify the matrix by posterior inference. As a result, our method allows different base features to have different matrices.

3.5 Conclusion

We presented a time-series prior for the parameters of probabilistic models; it produces sparse models and adapts the strength of temporal effects on each coefficient separately, based on the data, without an explosion in the number of hyperparameters. We showed how to do inference under this prior using variational approximations. We evaluated the prior for the task of forecasting volatility of stock returns from financial reports, and

demonstrated that it outperforms other competing models. We also evaluated the prior for the task of modeling a collection of texts over time, i.e., predicting the probability of words given some observed real-world variables. We showed that the prior achieved state-of-the-art results as well.

Chapter 4

Structured Sparsity in Learning Word Representations

When applying machine learning to text, the classic categorical representation of words as indices of a vocabulary fails to capture syntactic and semantic similarities that are easily discoverable in data (e.g., *pretty*, *beautiful*, and *lovely* have similar meanings, opposite to *unattractive*, *ugly*, and *repulsive*). In contrast, recent approaches to word representation learning apply neural networks to obtain dense, low-dimensional, continuous embeddings of words (Bengio *et al.*, 2003; Mnih and Teh, 2012; Collobert *et al.*, 2011; Huang *et al.*, 2012; Mikolov *et al.*, 2010, 2013b; Pennington *et al.*, 2014).

In this work, we propose an alternative approach based on decomposition of a high-dimensional matrix capturing surface statistics of association between a word and its "contexts" with sparse coding. As in past work, contexts are words that occur nearby in running text (Turney and Pantel, 2010). Learning is performed by minimizing a reconstruction loss function to find the best factorization of the input matrix.

The key novelty in our method is to govern the relationships among dimensions of the learned word vectors, introducing a hierarchical organization imposed through a structured penalty known as the group lasso (Yuan and Lin, 2006). The idea of regulating the order in which variables enter a model was first proposed by Zhao *et al.* (2009), and it has since been shown useful for other applications (Jenatton *et al.*, 2011a). Our approach is motivated by coarse-to-fine organization of words' meanings often found in the field of lexical semantics (see §4.2 for a detailed description), which mirrors evidence

for distributed nature of hierarchical concepts in the brain (Raposo *et al.*, 2012). Related ideas have also been explored in syntax (Petrov and Klein, 2008). It also has a foundation in cognitive science, where hierarchical structures have been proposed as representations of semantic cognition (Collins and Quillian, 1969). We propose a stochastic proximal algorithm for hierarchical sparse coding that is suitable for problems where the input matrix is very large and sparse. Our algorithm enables application of hierarchical sparse coding to learn word representations from a corpus of billions of word tokens and 400,000 word types.

On standard evaluation tasks—word similarity ranking, analogies, sentence completion, and sentiment analysis—we find that our method outperforms or is competitive with the best published representations. This chapter is based on materials from Yogatama *et al.* (2015).

4.1 Notation

The observable representation of word v is taken to be a vector $\boldsymbol{x}_v \in \mathbb{R}^C$ of cooccurrence statistics with C different contexts. Most commonly, each context is a possible neighboring word within a fixed window.[1] Following many others, we let $x_{v,c}$ be the pointwise mutual information (PMI) between the occurrence of context word c within a five-word window of an occurrence of word v (Turney and Pantel, 2010; Murphy *et al.*, 2012; Faruqui and Dyer, 2014; Levy and Goldberg, 2014).

In sparse coding, the goal is to represent each input vector $\boldsymbol{x} \in \mathbb{R}^C$ as a sparse linear combination of basis vectors. Given a stacked input matrix $\boldsymbol{X} \in \mathbb{R}^{C \times V}$, where V is the number of words, we seek to minimize:

$$\arg\min_{\boldsymbol{D} \in \mathcal{D}, \boldsymbol{A}} \|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{A}\|_F^2 + \lambda \Omega(\boldsymbol{A}), \tag{4.1}$$

where $\boldsymbol{D} \in \mathbb{R}^{C \times M}$ is the dictionary of basis vectors, $\mathcal{D}$ is the set of matrices whose columns have small (e.g., less than or equal to one) ℓ_2 norm, $\boldsymbol{A} \in \mathbb{R}^{M \times V}$ is the code matrix (each column of $\boldsymbol{A}$ represents a word), λ is a regularization hyperparameter, and

[1] Others include: global context (Huang *et al.*, 2012), multilingual context (Faruqui and Dyer, 2014), geographic context (Bamman *et al.*, 2014), brain activation (Fyshe *et al.*, 2014), and second-order context (Schutze, 1998).

Ω is the regularizer. Here, we use the squared loss for the reconstruction error, but other loss functions could also be used (Lee *et al.*, 2009). Note that it is not necessary, although typical, for M to be less than C (when $M > C$, it is often called an overcomplete representation). The most common regularizer is the ℓ_1 penalty, which results in sparse codes. While structured regularizers are associated with sparsity as well (e.g., the group lasso encourages group sparsity), our motivation is to use Ω to encourage a coarse-to-fine organization of latent dimensions of the learned representations of words.

4.2 Structured Sparse Coding for Learning Word Representations

For $\Omega(\boldsymbol{A})$, we design a forest-structured regularizer that encourages the model to use some dimensions in the code space before using other dimensions. Consider the trees in Figure 4.1. In this example, there are 13 variables in each tree, and 26 variables in total (i.e., $M = 26$), each corresponding to a latent dimension for *one particular word*. These trees describe the order in which variables "enter the model" (i.e., take nonzero values). In general, a node may take a nonzero value only if its ancestors also do. For example, nodes 3 and 4 may only be nonzero if nodes 1 and 2 are also nonzero. Our regularizer for column v of $\boldsymbol{A}$, denoted by $\boldsymbol{a}_v$ (in this example, $\boldsymbol{a}_v \in \mathbb{R}^{26}$), for the trees in Figure 4.1 is:[2]

$$\Omega(\boldsymbol{a}_v) = \sum_{i=1}^{26} \|[a_{v,i}, \quad \boldsymbol{a}_{v,\text{Descendants}(i)}]\|_2$$

where Descendants(i) returns the (possibly empty) set of descendants of node i, and $[.]$ returns the subvector of $\boldsymbol{a}_v$ by considering only $a_{v,i}$ and $\boldsymbol{a}_{v,\text{Descendants}(i)}$.[3] Jenatton *et al.* (2011a) proposed a related penalty with only one tree for learning image and document representations.

Let us analyze why organizing the code space this way is helpful in learning better

[2] $\Omega(\boldsymbol{A})$ is computed by adding components of $\Omega(\boldsymbol{a}_v)$ for all columns of $\boldsymbol{A}$.

[3] Note that if $\|[a_{v,i}, \boldsymbol{a}_{v,\text{Descendants}(i)}]\|_2$ is below a regularization threshold ($a_{v,i}$ is a zero node), $\|[\boldsymbol{a}_{v,\text{Descendants}(i)}]\|_2$ is also below the threshold (all its descendants are zero nodes as well). Conversely, if $\|[a_{v,i}, \boldsymbol{a}_{v,\text{Descendants}(i)}]\|_2$ is above the threshold ($a_{v,i}$ is a nonzero node), $\|[a_{v,\text{Parent}(i)}, a_{v,i}, \boldsymbol{a}_{v,\text{Descendants}(i)}]\|_2$ is also above the threshold ($a_{v,\text{Parent}(i)}$ is also a nonzero node).

word representations. Recall that the goal is to have a good dictionary $\boldsymbol{D}$ and code matrix $\boldsymbol{A}$. We apply the structured penalty to each column of $\boldsymbol{A}$. When we use the same structured penalty in these columns, we encode an additional shared constraint that the dimensions of $\boldsymbol{a}_v$ that correspond to top level nodes should focus on "general" contexts that are present in most words. In our case, this corresponds to contexts with extreme PMI values for many words, since they are the ones that incur the largest losses. As we go down the trees, more word-specific contexts can then be captured. As a result, we have better organization *across* words when learning their representations, which also translates to a more structured dictionary $\boldsymbol{D}$. Contrast this with the case when we use unstructured regularizers that penalize each dimension of $\boldsymbol{A}$ independently (e.g., lasso). In this case, each dimension of $\boldsymbol{a}_v$ has more flexibility to pay attention to any contexts (the only constraint that we encode is that the cardinality of the model should be small). We hypothesize that this is less appropriate for learning word representations, since the model has excessive freedom when learning $\boldsymbol{A}$ on noisy PMI values, which translates to poor $\boldsymbol{D}$.

The intuitive motivation for our regularizer comes from the field of lexical semantics, which often seeks to capture the relationships between words' meanings in hierarchically-organized lexicons. The best-known example is WordNet (Miller, 1995). Words with the same (or close) meanings are grouped together (e.g., *professor* and *prof* are synonyms), and fine-grained meaning groups ("synsets") are nested under coarse-grained ones (e.g., *professor* is a hyponym of *academic*). Our hierarchical sparse coding approach is still several steps away from inducing such a lexicon, but it seeks to employ the dimensions of a distributed word representation scheme in a similar coarse-to-fine way. In cognitive science, such hierarchical organization of semantic representations was first proposed by Collins and Quillian (1969).

4.3 Learning

Learning is accomplished by minimizing the function in Eq. 4.1, with the group lasso regularization function described in §4.2. The function is not convex with respect to $\boldsymbol{D}$ and $\boldsymbol{A}$, but it is convex with respect to each when the other is fixed. Alternating minimization routines have been shown to work reasonably well in practice for such

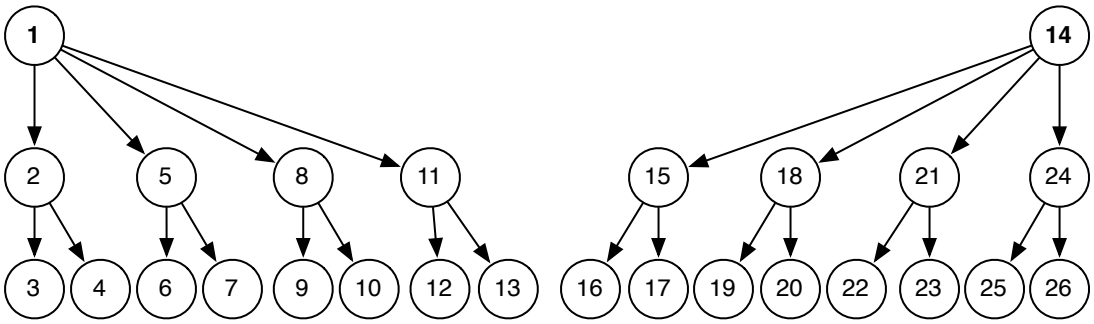


Figure 4.1: An example of a regularization forest that governs the order in which variables enter the model. In this example, 1 needs to be selected (nonzero) for $2, 3, \ldots, 13$ to be selected. However, $1, 2, \ldots, 13$ have nothing to do with the variables in the second tree: $14, 15, \ldots, 26$. See text for details.

problems (Lee *et al.*, 2007), but they are too expensive here due to:

- The size of $\boldsymbol{X} \in \mathbb{R}^{C \times V}$ (C and V are each on the order of 10^5).
- The many overlapping groups in the structured regularizer $\Omega(\boldsymbol{A})$.

One possible solution is based on the online dictionary learning method of Mairal *et al.* (2010). For T iterations, we:

- Sample a mini-batch of words and (in parallel) solve for each one's $\boldsymbol{a}$ using proximal methods or alternating directions method of multipliers, shown to work well for overlapping group lasso problems (Jenatton *et al.*, 2011a; Qin and Goldfarb, 2012; Yogatama and Smith, 2014b).
- Update $\boldsymbol{D}$ using the block coordinate descent algorithm of Mairal *et al.* (2010).

Finally, we parallelize solving for all columns of $\boldsymbol{A}$, which are separable once $\boldsymbol{D}$ is fixed. In our experiments, we use this algorithm for a medium-sized corpus.

The main difficulty of learning word representations with hierarchical sparse coding is, again, that the size of the input matrix can be very large. When we use neighboring words as the contexts, the numbers of rows and columns are the size of the vocabulary. For a medium-sized corpus with hundreds of millions of word tokens, we typically have one or two hundred thousand unique words, so the above algorithm is still applicable. For a large corpus with billions of word tokens, this number can easily double or triple,

making learning very expensive. We propose an alternative learning algorithm for such cases.

4.3.1 Stochastic Proximal Methods

We rewrite Eq. 4.1 as:

$$\arg\min_{\boldsymbol{D},\boldsymbol{A}} \sum_{c,v} (x_{c,v} - \boldsymbol{d}_c \cdot \boldsymbol{a}_v)^2 + \lambda\Omega(\boldsymbol{A}) + \tau \sum_m \|\boldsymbol{d}_m\|_2^2$$

where (abusing notation) $\boldsymbol{d}_c$ denotes the c-th row vector of $\boldsymbol{D}$ and $\boldsymbol{d}_m$ denotes the m-th column vector of $\boldsymbol{D}$ (recall that $\boldsymbol{D} \in \mathbb{R}^{C \times M}$). At each iteration, we sample an entry $x_{c,v}$ and perform gradient updates to the corresponding row $\boldsymbol{d}_c$ and column $\boldsymbol{a}_v$. Instead of considering all elements of the input matrix, our algorithm allows approximating the solution by using only some (e.g., nonzero) entries of the input matrix $\boldsymbol{X}$ if necessary.

We directly penalize columns of $\boldsymbol{D}$ by their squared ℓ_2 norm as an alternative to constraining columns of $\boldsymbol{D}$ to have unit ℓ_2 norm. The advantage of this transformation is that we have eliminated a projection step for columns of $\boldsymbol{D}$. Instead, we can include the gradient of the penalty term in the stochastic gradient update. We apply the proximal operator associated with $\Omega(\boldsymbol{a}_v)$ as a composition of elementary proximal operators with no group overlaps, similar to Jenatton *et al.* (2011a). This can be done by recursively visiting each node of a tree and applying the proximal operator for the group lasso penalty associated with that node (i.e., the group lasso penalty where the node is the topmost node and the group consists of the node and all of its descendants). The proximal operator associated with node m, denoted by $\text{prox}_{\Omega_m,\lambda}$, is simply the block-thresholding operator for node m and all its descendants.

Since each entry $x_{c,v}$ only depends on $\boldsymbol{d}_c$ and $\boldsymbol{a}_v$, we can sample multiple entries and perform the updates in parallel as long as they do not share c and v. In our case, where C and V are on the order of hundreds of thousands and we only have tens or hundreds of processors, finding elements that do not violate this constraint is easy. For example, there are typically a huge number of nonzero entries (on the order of billions). Using a sampling procedure that favors entries with higher (absolute) PMI values can lead to reasonably good word representations faster, so we can sample an entry with

Algorithm 1 Fast algorithm for learning word representations with the forest regularizer.

Require: matrix $\boldsymbol{X}$, regularization constant λ and τ, learning rate sequences $\eta_0, \ldots, \eta_T$, number of iterations T
  Initialize $\boldsymbol{D}_0$ and $\boldsymbol{A}_0$ randomly
  for $t = 1, \ldots, T$ {can be parallelized, see text for details} **do**
    Sample $x_{c,v}$ with probability proportional to its (absolute) value
    $\boldsymbol{d}_c = \boldsymbol{d}_c + 2\eta_t(\boldsymbol{a}_v(x_{c,v} - \boldsymbol{d}_c \cdot \boldsymbol{a}_v) - \tau \boldsymbol{d}_c)$
    $\boldsymbol{a}_v = \boldsymbol{a}_v + 2\eta_t(\boldsymbol{d}_c(x_{c,v} - \boldsymbol{d}_c \cdot \boldsymbol{a}_v))$
    for $m = 1, \ldots, M$ **do**
      $\text{prox}_{\Omega_m,\lambda}(\boldsymbol{a}_v)$, where $\Omega_m = \|\langle a_{v,m}, \ \ \boldsymbol{a}_{v,\text{Descendants}(m)}\rangle\|_2$
    end for
  end for

probability proportional to its absolute value.[4] Algorithm 1 summarizes our method.

4.3.2 Convergence Analysis

We analyze the convergence of Algorithm 1 for the basic setting where we uniformly sample elements of the input matrix $\boldsymbol{X}$. Similar to Mairal *et al.* (2010), we can rewrite our optimization problem as: $\arg\min_{\boldsymbol{A}} \sum_{t=1}^{T} \mathcal{L}^t(\boldsymbol{A}) + \lambda\Omega(\boldsymbol{A})$, where $\mathcal{L}^t(\boldsymbol{A}) = \|\boldsymbol{X} - \boldsymbol{D}^t\boldsymbol{A}\|_F^2 + \tau \sum_m \|\boldsymbol{d}_m^t\|_2^2$, and $\boldsymbol{D}^t = \boldsymbol{D}^{t-1} + 2\eta_t((\boldsymbol{X} - \boldsymbol{D}^{t-1}\boldsymbol{A}^{t-1})\boldsymbol{A}^{t-1\top} - \tau\boldsymbol{D}^{t-1})$. Note that $\mathcal{L}^t(\boldsymbol{A})$ is a nonconvex (with respect to $\boldsymbol{A}$) continuously differentiable function, which is the loss at timestep t after performing the dictionary update step. For ease of exposition, in the following, we assume $\boldsymbol{A}$ is a vector formed by stacking together columns of the matrix $\boldsymbol{A}$.

Let us denote $\mathcal{L}(\boldsymbol{A}) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}^t(\boldsymbol{A})$. We show convergence of Algorithm 1 to a stationary point under the assumption that we have an unbiased estimate of the gradient with respect to $\boldsymbol{A}$: $\mathbb{E}[\bigtriangledown\mathcal{L}^t(\boldsymbol{A})] = \bigtriangledown\mathcal{L}(\boldsymbol{A})$. This can also be stated as $\mathbb{E}[\|\epsilon^t\|_2] = 0$, where $\|\epsilon^t\|_2 = \| \bigtriangledown \mathcal{L}(\boldsymbol{A}) - \bigtriangledown\mathcal{L}^t(\boldsymbol{A})\|_2$.

Our convergence proof uses the following definition of a stationary point and relies on a lemma from Sra (2012).

Definition 1. *A point $\boldsymbol{A}^*$ is a stationary point if and only if: $\boldsymbol{A}^* = prox_{\Omega,\lambda}(\boldsymbol{A}^* - \eta \bigtriangledown \ \mathcal{L}(\boldsymbol{A}^*))$.*

[4] In practice, we can use an even faster approximation of this sampling procedure by uniformly sampling a nonzero entry and multiplying its gradient by a scaling constant proportional to its absolute PMI value.

Lemma 2. *(Sra, 2012) Let F be a function with a (locally) Lipschitz continuous gradient with constant $L > 0$.*

$$F(\boldsymbol{A}^t) - F(\boldsymbol{A}^{t+1}) \geq \frac{2 - L\eta_t}{2\eta_t}\|\boldsymbol{A}^{t+1} - \boldsymbol{A}^t\|_2^2 - \|\epsilon^t\|_2\|\boldsymbol{A}^{t+1} - \boldsymbol{A}^t\|_2. \tag{4.2}$$

Theorem 3. *Let the assumption of an unbiased estimate of the gradient be satisfied and the learning rate satisfies $0 < \eta_t < \frac{2}{L}$. Algorithm 1 converges to a stationary point in expectation.*

Proof. We first show that $\mathcal{L}(\boldsymbol{A}^t) - \mathcal{L}(\boldsymbol{A}^{t+1})$ is bounded below in expectation. Since $\mathcal{L}$ has a Lipschitz continuous gradient, Lemma 2 already bounds $\mathcal{L}(\boldsymbol{A}^t) - \mathcal{L}(\boldsymbol{A}^{t+1})$. Let us denote the Lipschitz constant of $\mathcal{L}$ by L. Given our assumption about the error of the stochastic gradient (vanishing error), we have:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\boldsymbol{A}^t) - \mathcal{L}(\boldsymbol{A}^{t+1})] &\geq \frac{2 - L\eta_t}{2\eta_t}\mathbb{E}[\|\boldsymbol{A}^{t+1} - \boldsymbol{A}^t\|_2^2] \\ &= \frac{2 - L\eta_t}{2\eta_t}\mathbb{E}[\|\text{prox}_{\Omega,\lambda}(\boldsymbol{A}^t - \eta_t \bigtriangledown \mathcal{L}^t(\boldsymbol{A}^t)) - \boldsymbol{A}^t\|_2^2]\end{aligned}$$

Since our learning rate satisfies $0 < \eta_t < \frac{2}{L}$, it is easy to show that the above is never negative. In order to show convergence, we then bound this quantity:

$$\begin{aligned}&\sum_{t=1}^{T}\frac{2 - L\eta_t}{2\eta_t}\mathbb{E}[\|\text{prox}_{\Omega,\lambda}(\boldsymbol{A}^t - \eta_t \bigtriangledown \mathcal{L}^t(\boldsymbol{A}^t)) - \boldsymbol{A}^t\|_2^2] \\ &\leq \sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\boldsymbol{A}^t) - \mathcal{L}(\boldsymbol{A}^{t+1})] = \mathbb{E}[\mathcal{L}(\boldsymbol{A}^1) - \mathcal{L}(\boldsymbol{A}_{T+1})] \\ &\leq \mathbb{E}[\mathcal{L}(\boldsymbol{A}^1) - \mathcal{L}(\boldsymbol{A}^*)]\end{aligned}$$

The right hand side (third line) is a positive constant and the left hand side (first line) is never negative, so $\mathbb{E}[\|\text{prox}_{\Omega,\lambda}(\boldsymbol{A}^t - \eta_t \bigtriangledown \mathcal{L}^t(\boldsymbol{A}^t)) - \boldsymbol{A}^t\|_2^2] \to 0$, which means that $\boldsymbol{A}_t$ converges to a stationary point in expectation based on the definition of a stationary point in Definition 1. □

4.4 Experiments

We present a controlled comparison of the forest regularizer against several strong baseline word representations learned on a fixed dataset, across several tasks. In §4.4.4 we compare to publicly available word vectors trained on different data.

4.4.1 Setup and Baselines

We use the WMT-2011 English news corpus as our training data.[5] The corpus contains about 15 million sentences and 370 million words. The size of our vocabulary is 180,834.[6]

In our experiments, we use forests similar to those in Figure 4.1 to organize the latent word space. Note that the example has 26 nodes (2 trees). We choose to evaluate performance with $M = 52$ (4 trees) and $M = 520$ (40 trees).[7] We denote the sparse coding method with regular ℓ_1 penalty by SC, and our method with structured regularization (§4.2) by FOREST. We set $\lambda = 0.1$. In this first set of experiments with a medium-sized corpus, we use the online learning algorithm of Mairal *et al.* (2010).

We compare with the following baseline methods:

- Turney and Pantel (2010): principal component analysis (PCA) by truncated singular value decomposition on $\boldsymbol{X}^\top$. Note that this is also the same as minimizing the squared reconstruction loss in Eq. 4.1 without any penalty on $\boldsymbol{A}$.
- Mikolov *et al.* (2010): a recursive neural network (RNN) language model. We obtain an implementation from `http://rnnlm.org/`.
- Mnih and Teh (2012): a log bilinear model that predicts a word given its context, trained using noise-contrastive estimation (NCE, Gutmann and Hyvarinen, 2010). We use our own implementation for this model.
- Mikolov *et al.* (2013b): a log bilinear model that predicts a word given its context (continuous bag of words, CBOW), trained using negative sampling (Mikolov

[5] `http://www.statmt.org/wmt11/`

[6] We replace words with frequency less than 10 with #rare# and numbers with #number#.

[7] In preliminary experiments we explored binary tree structures and found they did not work as well. We hypothesized that better organizations of word meanings are flatter, as opposed to deeper. We leave a more extensive exploration of tree structures to future work.

et al., 2013a). We obtain an implementation from `https://code.google.com/p/word2vec/`.

- Mikolov *et al.* (2013b): a log bilinear model that predicts context words given a target word (skip gram, SG), trained using negative sampling (Mikolov *et al.*, 2013a). We obtain an implementation from `https://code.google.com/p/word2vec/`.

- Pennington *et al.* (2014): a log bilinear model that is trained using AdaGrad (Duchi *et al.*, 2011) to minimize a weighted square error on global (log) cooccurrence counts (global vectors, GV). We obtain an implementation from `http://nlp.stanford.edu/projects/glove/`.

Our focus here is on comparisons of model architectures. For a fair comparison, we train all competing methods on the same corpus using a context window of five words (left and right). For the baseline methods, we use default settings in the provided implementations (or papers, when implementations are not available and we reimplement the methods). We also trained the two baseline methods introduced by Mikolov *et al.* (2013b) with hierarchical softmax using a binary Huffman tree instead of negative sampling; consistent with Mikolov *et al.* (2013a), we found that negative sampling performs better and show hierarchical softmax results in §4.4.6.

4.4.2 Evaluation

We evaluate on the following benchmark tasks.

Word similarity The first task evaluates how well the representations capture word similarity. For example *beautiful* and *lovely* should be closer in distance than *beautiful* and *science*. We evaluate on a suite of word similarity datasets, subsets of which have been considered in past work: WordSim 353 (Finkelstein *et al.*, 2002), rare words (Luong *et al.*, 2013), and many others. We use the following word similarity datasets in our experiments:

- Finkelstein *et al.* (2002): WordSimilarity dataset (353 pairs).

- Agirre *et al.* (2009): a subset of WordSimilarity dataset for evaluating similarity (203 pairs).

- Agirre *et al.* (2009): a subset of WordSimilarity dataset for evaluating relatedness (252 pairs).
- Miller and Charles (1991): semantic similarity dataset (30 pairs)
- Rubenstein and Goodenough (1965): contains only nouns (65 pairs)
- Luong *et al.* (2013): rare words (2,034 pairs)
- Bruni *et al.* (2012): frequent words (3,000 pairs)
- Radinsky *et al.* (2011): MTurk-287 dataset (287 pairs)
- Halawi and Dror (2014): MTurk-771 dataset (771 pairs)
- Yang and Powers (2006): contains only verbs (130 pairs)

Following standard practice, for each competing model, we compute cosine distances between word pairs in word similarity datasets, then rank and report Spearman's rank correlation coefficient (Spearman, 1904) between the model's rankings and human rankings.

Syntactic and semantic analogies The second evaluation dataset is two analogy tasks proposed by Mikolov *et al.* (2013b). These questions evaluate syntactic and semantic relations between words. There are 10,675 syntactic questions (e.g., *walking : walked :: swimming : swam*) and 8,869 semantic questions (e.g., *Athens : Greece :: Oslo :: Norway*). In each question, one word is missing, and the task is to correctly predict the missing word. We use the vector offset method (Mikolov *et al.*, 2013b) that computes the vector $\boldsymbol{b} = \boldsymbol{a}_{\text{Athens}} - \boldsymbol{a}_{\text{Greece}} + \boldsymbol{a}_{\text{Oslo}}$. We only consider a question to be answered correctly if the returned vector ($\boldsymbol{b}$) has the highest cosine similarity to the correct answer (in this example, $\boldsymbol{a}_{\text{Norway}}$).

Sentence completion The third evaluation task is the Microsoft Research sentence completion challenge (Zweig and Burges, 2011). In this task, the goal it to choose from a set of five candidate words which one best completes a sentence. For example: *Was she his {client, musings, discomfiture, choice, opportunity}, his friend, or his mistress?* (*client* is the

Table 4.1: Summary of results. We report Spearman's correlation coefficient for the word similarity task and accuracies (%) for other tasks. Higher values are better (higher correlation coefficient or higher accuracy). The last two methods (columns) are new to this paper, and our proposed method is in the last column.

M	Task	PCA	RNN	NCE	CBOW	SG	GV	SC	FOREST
52	Word similarity	0.39	0.26	0.48	0.43	0.49	0.43	0.49	**0.52**
	Syntactic analogies	18.88	10.77	24.83	23.80	26.69	**27.40**	11.84	24.38
	Semantic analogies	8.39	2.84	25.29	8.45	19.49	**26.23**	4.50	9.86
	Sentence completion	27.69	21.31	**30.18**	25.60	26.89	25.10	25.10	28.88
	Sentiment analysis	74.46	64.85	70.84	68.48	71.99	72.60	75.51	**75.83**
520	Word similarity	0.50	0.31	0.59	0.53	0.58	0.51	0.58	**0.66**
	Syntactic analogies	40.67	22.39	33.49	52.20	**54.64**	44.96	22.02	48.00
	Semantic analogies	28.82	5.37	**62.76**	12.58	39.15	55.22	15.46	41.33
	Sentence completion	30.58	23.11	33.07	26.69	26.00	33.76	28.59	**35.86**
	Sentiment analysis	81.70	72.97	78.60	77.38	79.46	79.40	78.20	**81.90**

Table 4.2: Results on the syntactic and semantic analogies tasks with a bigger corpus ($M = 260$).

Task	CBOW	SG	GV	FOREST
Syntactic	61.37	63.61	65.56	**65.63**
Semantic	23.13	54.41	**74.35**	52.88

correct answer). We choose the candidate with the highest average similarity to every other word in the sentence.[8]

Sentiment analysis The last evaluation task is sentence-level sentiment analysis. We use the movie reviews dataset from Socher *et al.* (2013). The dataset consists of 6,920 sentences for training, 872 sentences for development, and 1,821 sentences for testing. We train ℓ_2-regularized logistic regression to predict binary sentiment, tuning the regularization strength on development data. We represent each example (sentence) as an M-dimensional vector constructed by taking the average of word representations of words appearing in that sentence.

The analogy, sentence completion, and sentiment analysis tasks are evaluated on prediction accuracy.

[8]We note that unlike matrix decomposition based approaches, some of the neural network based models can directly compute the scores of context words given a possible answer (Mikolov *et al.*, 2013b). We choose to use average similarities for a fair comparison of the representations.

Table 4.3: Comparison to previously published word representations. The five rightmost columns correspond to the tasks described above; parenthesized values are the number of in-vocabulary items that could be evaluated.

Models	M	V	W. Sim.	Syntactic	Semantic	Sentence	Sentiment
CW	50	130,000	(6,225) 0.51	(10,427) 12.34	(8,656) 9.33	(976) 24.59	69.36
RNN-DC		100,232	(6,137) 0.32	(10,349) 10.94	(7,853) 2.60	(964) 19.81	67.76
HLBL		246,122	(6,178) 0.11	(10,477) 8.98	(8,446) 1.74	(990) 19.90	62.33
NNSE		34,107	(3,878) 0.23	(5,114) 1.47	(1,461) 2.46	(833) 0.04	64.80
HPCA		178,080	(6,405) 0.29	(10,553) 10.42	(8,869) 3.36	(993) 20.14	67.49
FOREST	52	180,834	(6,525) **0.52**	(10,675) **24.38**	(8,733) **9.86**	(1,004) **28.88**	**75.83**

4.4.3 Results

Table 4.1 shows results on all evaluation tasks for $M = 52$ and $M = 520$. Runtime will be discussed in §4.4.5. In the similarity ranking and sentiment analysis tasks, our method performed the best in both low and high dimensional embeddings. In the sentence completion challenge, our method performed best in the high-dimensional case and second-best in the low-dimensional case. Importantly, FOREST outperforms PCA and unstructured sparse coding (SC) on every task. We take this collection of results as support for the idea that coarse-to-fine organization of latent dimensions of word representations captures the relationships between words' meanings better than unstructured organization.

Analogies Our results on the syntactic and semantic analogies tasks for all models are below state-of-the-art performance from previous work. We hypothesize that this is because performing well on these tasks requires training on a bigger corpus (in general, the bigger the training corpus is, the better the models will be for all tasks). We combine our WMT-2011 corpus with other news corpora and Wikipedia to obtain a corpus of 6.8 billion words. The size of the vocabulary of this corpus is 401,150. We retrain four models that are scalable to a corpus of this size: CBOW, SG, GV, and FOREST.[9] We select $M = 260$ to balance the trade-off between training time and performance ($M = 52$ does not perform as well, and $M = 520$ is computationally expensive). For FOREST, we use the fast learning algorithm in §4.3, since the online learning algorithm of Mairal *et al.* (2010) does not scale to a problem of this size. We report accuracies on the syntactic

[9]Our NCE implementation is not optimized and therefore not scalable.

and semantic analogies tasks in Table 4.2. All models benefit significantly from a bigger corpus, and the performance levels are now comparable with previous work. On the syntactic analogies task, FOREST is competitive with GV and both models outperformed SG and CBOW. On the semantic analogies task, GV outperformed SG, FOREST, and CBOW.

4.4.4 Other Comparisons

In Table 4.3, we compare with five other baseline methods for which we do not train on our training data but pre-trained 50-dimensional word representations are available:

- Collobert *et al.* (2011): a neural network language model trained on Wikipedia data for 2 months (CW).[10]
- Huang *et al.* (2012): a neural network model that uses additional global document context (RNN-DC).[11]
- Mnih and Hinton (2008): a log bilinear model that predicts a word given its context, trained using hierarchical softmax (HLBL).[12]
- Murphy *et al.* (2012): a word representation trained using non-negative sparse embedding (NNSE) on dependency relations and document cooccurrence counts.[13] These vectors were learned using sparse coding, but using different contexts (dependency and document cooccurrences), a different training method, and with a nonnegativity constraint. Importantly, there is no hierarchy in the code space, as in FOREST.[14]
- Lebret and Collobert (2014): a word representation trained using Hellinger PCA (HPCA).[15]

These methods were all trained on different corpora, so they have different vocabularies that do not always include all of the words found in the tasks. We estimate performance

[10] `http://ronan.collobert.com/senna/`
[11] `http://goo.gl/Wujc5G`
[12] `http://metaoptimize.com/projects/wordreprs/` (Turian *et al.*, 2010)
[13] `http://www.cs.cmu.edu/~bmurphy/NNSE/`.
[14] We found that NNSE trained using our contexts performed very poorly; see additional results in §4.4.6.
[15] `http://lebret.ch/words/`

Table 4.4: Training time comparisons for skip gram (SG), glove (GV), and FOREST. For the medium corpus (370 million words), we learn FOREST with Mairal *et al.* (2010). This algorithm consists of two major steps: online learning of $\boldsymbol{D}$ and parallel learning of $\boldsymbol{A}$ with fixed $\boldsymbol{D}$ (see §4.3). "$*$" indicates that we only use 640 cores for the second step, since the first step only takes less than 2 hours even for $M = 520$. We can also see from this table that it becomes too expensive to use this algorithm for a bigger corpus. For the bigger corpus (6.8 billion words), we use Algorithm 1.

Models	Corpus	M	Cores	Time
SG	370M	52	16	1.5 hours
GV	370M	52	16	0.4 hours
FOREST	370M	52	640*	2.5 hours
SG	370M	520	16	5 hours
GV	370M	520	16	2.5 hours
FOREST	370M	520	640*	20 hours
SG	6.8B	260	16	6.5 hours
GV	6.8B	260	16	4 hours
FOREST	6.8B	260	16	4 hours

on the items for which prediction is possible, and show the count for each method in Table 4.3. This comparison should be interpreted cautiously since many experimental variables are conflated; nonetheless, FOREST performs strongly.

4.4.5 Discussion

In terms of running time, FOREST is reasonably fast to learn. See Table 4.4 for comparisons with other state-of-the-art methods.

Our method produces sparse word representations with exact zeros. We observe that the sparse coding method without a structured regularizer produces sparser representations, but it performs worse on our evaluation tasks, indicating that it zeroes out meaningful dimensions. For FOREST with $M = 52$ and $M = 520$, the average numbers of nonzero entries are 91% and 85% respectively. While our word representations are not extremely sparse, this makes intuitive sense since we try to represent about 180,000 contexts in only 52 (520) dimensions. We also did not tune λ. As we increase M, we get sparser representations.

We visualize our $M = 52$ word representations (FOREST) related to animals (10 words) and countries (10 words). We show the coefficient patterns for these words

in Figure 4.2. We can see that in both cases, there are dimensions where the coefficient signs (positive or negative) agree for all 10 words (they are mostly on the right and left sides of the plots). Note that the dimensions where all the coefficients agree are not the same in animals and countries. The larger magnitude of the vectors for more abstract concepts (*animal*, *animals*, *country*, *countries*) is reminiscent of neural imaging studies that have found evidence of more global activation patterns for processing superordinate terms (Raposo *et al.*, 2012). In Figure 4.3, we show tree visualizations of coefficients of word representations for *animal*, *horse*, and *elephant*. We show one tree for $M = 52$ (there are four trees in total, but other trees exhibit similar patterns). Coefficients that differ in sign mostly correspond to leaf nodes, validating our motivation that top level nodes should focus more on "general" contexts (for which they should be roughly similar for *animal*, *horse*, and *elephant*) and leaf nodes focus on word-specific contexts. One of the leaf nodes for animal is driven to zero, suggesting that more abstract concepts require fewer dimensions to explain.

For FOREST, SG, and NCE with $M = 520$, we project the learned word representations into two dimensions using the t-SNE tool (van der Maaten and Hinton, 2008).[16] We show projections of words related to the concept "good" vs. "bad" and "man" vs. "woman" in Figure 4.4.[17]

4.4.6 Additional Results

In Table 4.5, we compare FOREST with three additional baselines:

- Murphy *et al.* (2012): a word representation trained using non-negative sparse embedding (NNSE) on our corpus. Similar to the authors, we use an NNSE implementation from `http://spams-devel.gforge.inria.fr/` (Mairal *et al.*, 2010).
- Mikolov *et al.* (2013b): a log bilinear model that predicts a word given its context, trained using hierarchical softmax with a binary Huffman tree (continuous bag of words, CBOW-HS). We use an implementation from `https://code.google.com/p/word2vec/`.

[16] `http://homepage.tudelft.nl/19j49/t-SNE.html`

[17] Since t-SNE is a non-convex method, we run it 10 times and choose the plots with the lowest t-SNE error.

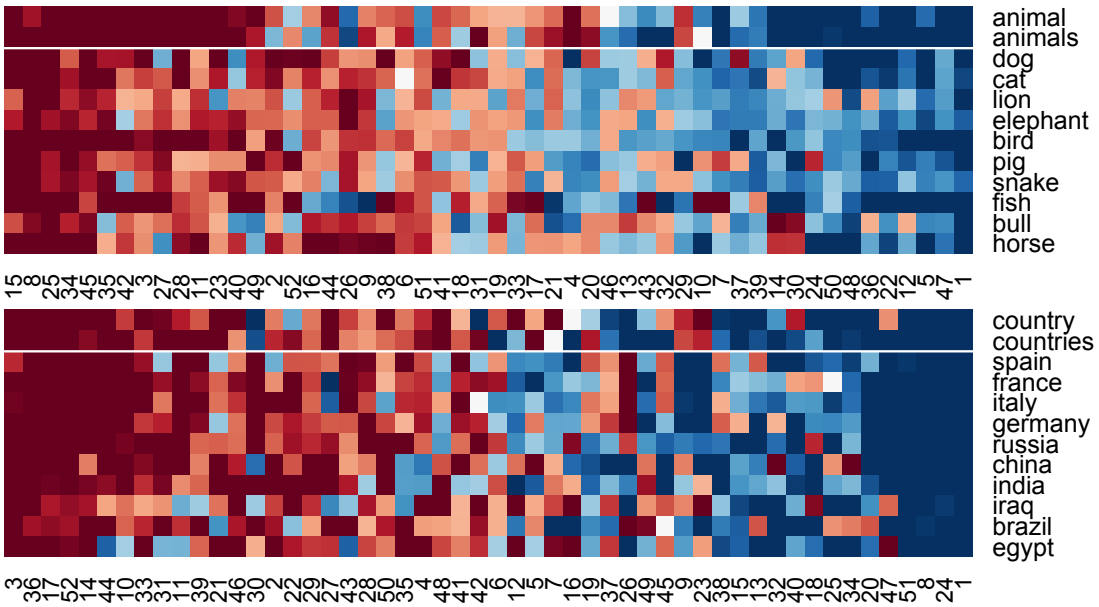
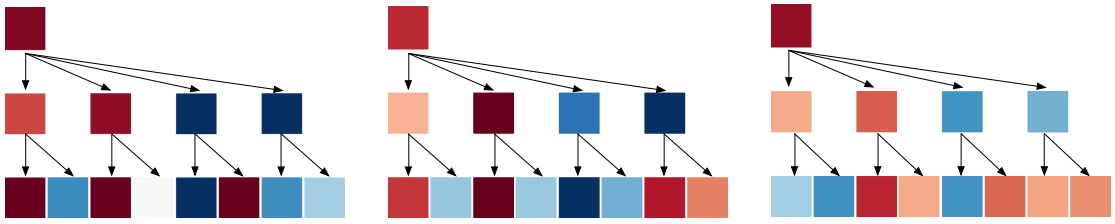


Figure 4.2: Heatmaps of word representations for 10 animals (top) and 10 countries (bottom) for $M = 52$ from FOREST. Red indicates negative values, blue indicates positive values (darker colors correspond to more extreme values); white denotes exact zero. The x-axis shows the original dimension index, we show the dimensions from the most negative (left) to the most positive (right), within each block, for readability.



(a) animal (b) horse (c) elephant

Figure 4.3: Tree visualizations of word representations for *animal* (left), *horse* (center), *elephant* (right) for $M = 52$. We use the same color coding scheme as in Figure 4.2. Here, we only show one tree (out of four), but other trees exhibit similar patterns.

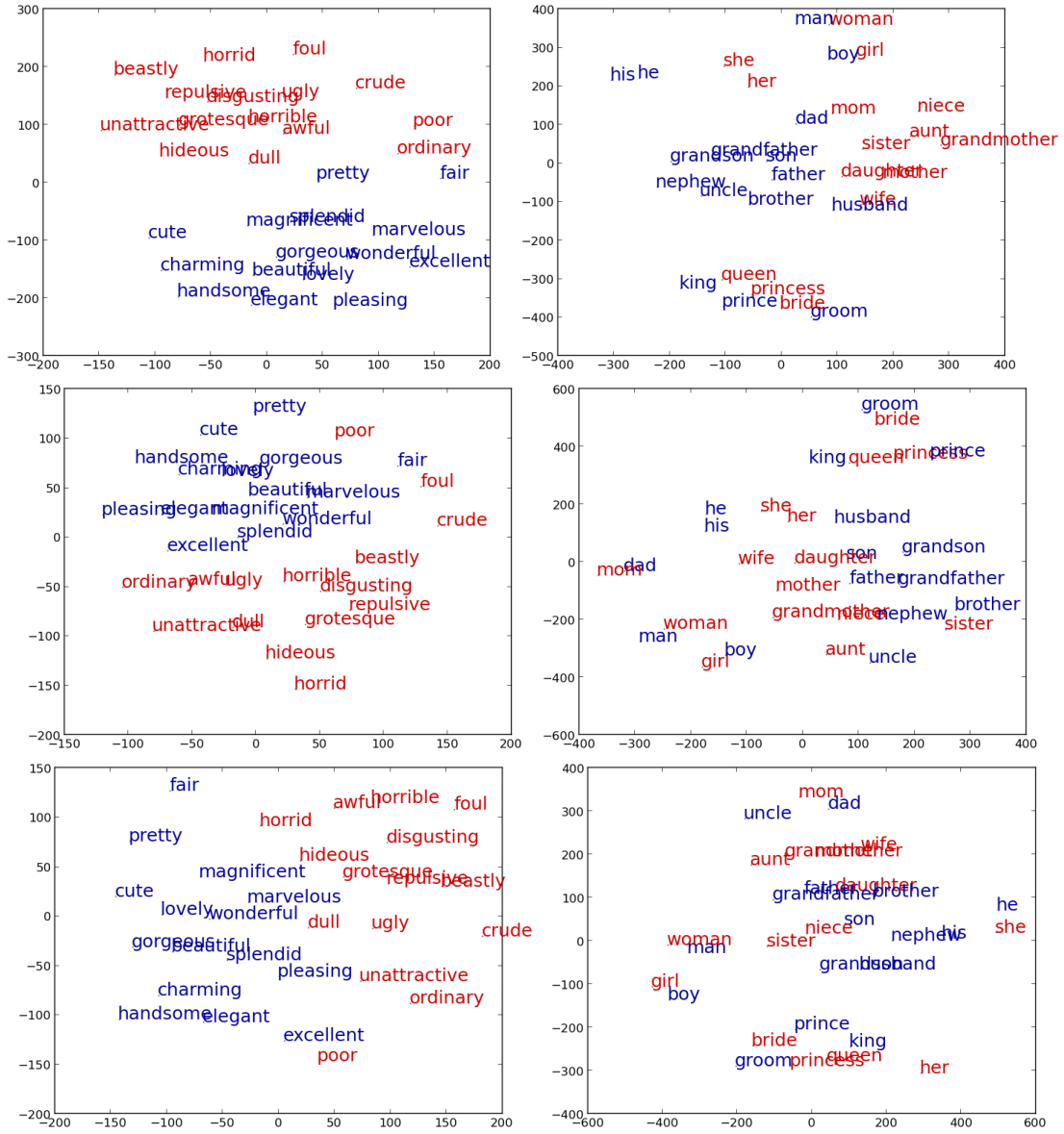


Figure 4.4: Two dimensional projections of the FOREST (top), NCE (middle), and NCE (bottom) word representations using the t-SNE tool (van der Maaten and Hinton, 2008). Words associated with "good" (left) and "man" (right) are colored in blue, words associated with "bad" (left) and "woman" (right) are colored in red. The two plots on the top left are the same plots shown in the paper.

- Mikolov *et al.* (2013b): a log bilinear model that predicts context words given a target word, trained using hierarchical softmax with a binary Huffman tree

Table 4.5: Summary of results for non-negative sparse embedding (NNSE), continuous bag-of-words and skip gram models trained with hierarchical softmax (CBOW-HS and SG-HS). Higher number is better (higher correlation coefficient or higher accuracy).

M	Task	NNSE	CBOW-HS	SG-HS	FOREST
52	Word similarity	0.04	0.38	0.47	**0.52**
	Syntactic analogies	0.10	19.50	**24.87**	24.38
	Semantic analogies	0.01	5.31	**14.77**	9.86
	Sentence completion	0.01	22.51	28.78	**28.88**
	Sentiment analysis	61.12	68.92	71.72	**75.83**
520	Word similarity	0.05	0.50	0.57	**0.66**
	Syntactic analogies	0.81	46.00	**50.40**	48.00
	Semantic analogies	0.57	8.00	31.05	**41.33**
	Sentence completion	22.81	25.80	27.79	**35.86**
	Sentiment analysis	67.05	78.50	79.57	**81.90**

(skip gram, SG-HS). We use an implementation from `https://code.google.com/p/word2vec/`.

4.5 Conclusion

In this chapter, we introduced a new method for learning word representations based on hierarchical sparse coding. The regularizer encourages hierarchical organization of the latent dimensions of vector-space word embeddings. We showed that our method outperforms state-of-the-art methods on word similarity ranking, sentence completion, syntactic analogies, and sentiment analysis tasks.

Chapter 5

Structured Sparsity in Text Categorization

In many high-dimensional learning problems, only some parts of an observation are important to the prediction task; for example, the cues to correctly categorizing a document (sentence) may lie in a handful of its sentences (phrases). In the case of text analysis, this idea was exploited by Yessenalina *et al.* (2010) and Tackstrom and McDonald (2011) using latent variable models that explicitly encode which sentences in a document are relevant to a polarity judgment (e.g., is the author's sentiment toward a film positive or negative?). Such models require sacrifices: convexity during parameter estimation and simplicity of prediction algorithms (compared to linear models). In this chapter, we introduce a class of linguistically-motivated sparse regularizers for text categorization that exploits this intuition by encoding it in a penalty function. This framework enables efficient incorporation of linguistic biases (e.g., syntactic parse trees, thematic topics, hierarchical word clusterings, sentence boundaries) into conventional bag-of-words models without sacrificing convexity. We show how to efficiently solve the resulting optimization challenge using the alternating directions method of multipliers. We demonstrate that our approach consistently achieves more accurate models than lasso, ridge, and elastic net regularized baselines. More generally, our method is applicable to many high-dimensional learning problems where only some parts of an

observation are relevant to the prediction task.[1] This chapter is based on materials from Yogatama and Smith (2014b) and Yogatama and Smith (2014a).

5.1 Notation

We denote the feature vector to represent a document by $\boldsymbol{x} \in \mathbb{R}^V$, where V is the vocabulary size, and we represent documents as vectors of word frequencies (i.e., "bags of words"). Each document is associated with a response (output) variable y. For simplicity and without loss of generality, we assume $y \in \{-1, 1\}$. The parameter vector that we want to learn is denoted by $\boldsymbol{w}$. We use feature coefficients and feature weights interchangeably to describe $\boldsymbol{w}$. We denote the loss function by $\mathcal{L}(\boldsymbol{x}, \boldsymbol{w}, y)$; in this section it is the log loss:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{w}, y) = \log(1 + \exp(-y\boldsymbol{w}^\top \boldsymbol{x}))$$

The general framework can be extended to continuous responses (i.e., linear regression) and to other loss functions (e.g., SVMs' hinge loss).

The goal of the learning procedure is to estimate $\boldsymbol{w}$ for a given set of training documents $\{\boldsymbol{x}_d, y_d\}_{d=1}^D$ by minimizing the penalized training data loss:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \Omega(\boldsymbol{w}) + \sum_{d=1}^D \mathcal{L}(\boldsymbol{x}_d, \boldsymbol{w}, y_d)$$

where Ω is a regularization penalty to encourage models with small weight vectors. We defer explanations on how to efficiently learn models with linguistically motivated structured regularizers to §5.3.

[1] For example, in computer vision, we might be interested in classifying an image. We know that an image is composed of multiple entities. If we can detect these entities and we assume that not all of these entities are relevant to predicting the label of the image, we can apply our regularizer to this problem.

5.2 Linguistic Structured Sparsity

5.2.1 Sentence Regularization

Considerable study has been devoted to structure in text, both for purposes of theoretical linguistics and for practical applications. All this work builds on the idea that more accurate interpretation can be obtained by explicitly representing rhetorical, semantic, or syntactic structures that relate word tokens to each other. Here we consider one very simple kind of structure that can easily be recovered with high accuracy in documents: sentences.

Our basic idea is to define, for every sentence in the training data, a **group** of the features that are present (i.e., nonzero) in that sentence. (In our models, these features are all word frequencies.) These groups, in turn, serve to define a group lasso regularization term, which we call the "sentence regularizer":

$$\Omega_{sen}(\boldsymbol{w}) = \sum_{d=1}^{D} \sum_{s=1}^{S_d} \lambda_{d,s} \|\boldsymbol{w}_{d,s}\|_2,$$

where d ranges over documents (as before) and s over sentences within a document. S_d is the number of sentences in document d. $\boldsymbol{w}_{d,s}$ corresponds to the subvector of $\boldsymbol{w}$ such that the corresponding features are present in sentence s of document d. The regularizer can take into account the length of the sentence by encoding it in $\lambda_{d,s}$. In the following, for simplicity and without loss of generality, we assume $\forall d, \forall s, \lambda_{d,s} = \lambda_{sen}$.

To gain an intuition for this regularizer, consider the case where we apply the penalty only for a single document, d_0, which happens (unrealistically) never to use the same word more than once (i.e., $\|\boldsymbol{x}_{d_0}\|_\infty = 1$). Because it instantiates group lasso, the sentence regularizer will encourage some groups to go to zero (especially groups whose sentences contain no words strongly associated with a label in the rest of the corpus). The effect is that only some sentences in d_0 will be selected as relevant (i.e., $\{s : \boldsymbol{w}_{d_0,s} \neq \mathbf{0}\}$), and the rest will have $\boldsymbol{w}_{d_0,s} = \mathbf{0}$ and therefore will have no effect on the prediction for d_0. Further, the words deemed not relevant in d_0 will have no effect on the prediction for other documents.

Of course, in typical documents, many words will occur in more than one sentence, and we create a group for every sentence in the training corpus. This means that our

groups are heavily *overlapping*; a word that occurs in k sentences in the corpus will force its corresponding weight to associate with k groups. As a result, the regularizer mainly acts as a proxy to encourage group behavior of words appearing in the same sentences.

Comparison to latent variable models. Seen this way, we can draw connections between our model and latent variable models for sentiment analysis that explicitly "select" relevant sentences (Yessenalina *et al.*, 2010; Tackstrom and McDonald, 2011). Latent variables complicate inference, because prediction algorithms must reason about the additional variables. This sometimes leads to mixed inference problems (i.e., maximizing over y while marginalizing latent variables). Our method, by contrast, does not change the linear model bag-of-words family, so the prediction algorithm is unchanged. At inference time, there is no notion of "relevant" sentences.

More importantly, latent variable models lead to non-convex objective functions, so that learning methods' performance hinges on clever (or lucky) initialization (e.g., Yessenalina *et al.*, 2010). Our approach maintains convexity of the objective function, allowing for familiar guarantees about the parameter estimate.

5.2.2 Parse Tree Regularizer

Sentence boundaries are a rather superficial kind of linguistic structure; syntactic parse trees provide more fine-grained information. We introduce the parse tree regularizer, in which groups are defined for every constituent in every parse of a training data sentence. Nowadays, constitutent-level annotations are easy and fast to obtain. For example, we can use the mehod in Kong *et al.* (2015) to get them accurately from dependeny annotations.

Figure 5.1 illustrates the group structures derived from an example sentence from the Stanford sentiment treebank (Socher *et al.*, 2013). This regularizer captures the idea that *phrases* might be selected as relevant or (in most cases) irrelevant to a task, and is expected to be especially useful in sentence-level prediction tasks.

The parse-tree regularizer (omitting the group coefficients and λ) for one sentence

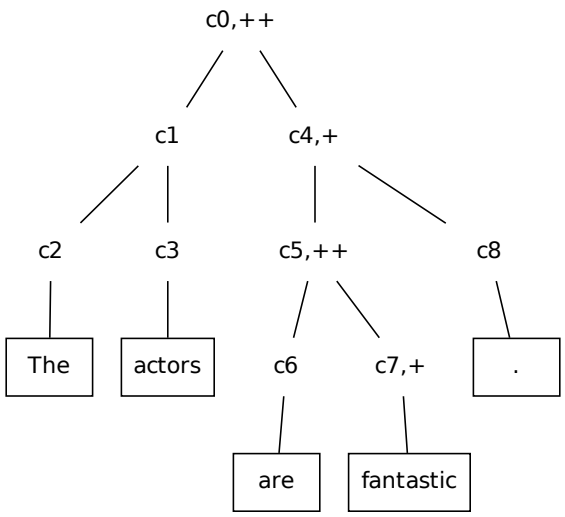


Figure 5.1: An example of a parse tree from the Stanford sentiment treebank, which annotates sentiment at the level of every constituent (indicated here by $+$ and $++$; no marking indicates neutral sentiment). The sentence is The actors are fantastic. Our regularizer constructs nine groups for this sentence, corresponding to $c_0, c_1, \ldots, c_8$. g_{c_0} consists of 5 weights—$\langle w_{the}, w_{actors}, w_{are}, w_{fantastic}, w_. \rangle$, exactly the same as the group in the sentence regularizer—g_{c_1} consists of 2 words, g_{c_4} of 3 words, etc. Notice that c_2, c_3, c_6, c_7, and c_8 each consist of only 1 word. The Stanford sentiment treebank has an annotation of sentiments at the constituent level. As in this example, most constituents are annotated as neutral.

with the parse tree shown in Figure 5.1 is:

$$\Omega_{tree}(\boldsymbol{w}) = \sqrt{|w_{the}|^2 + |w_{actors}|^2 + |w_{are}|^2 + |w_{fantastic}|^2 + |w_.|^2} + \sqrt{|w_{are}|^2 + |w_{fantastic}|^2 + |w_.^2|}$$
$$+ \sqrt{|w_{the}|^2 + |w_{actors}|^2} + \sqrt{|w_{are}|^2 + |w_{fantastic}|^2} + |w_{the}| + |w_{actors}| + |w_{are}| + |w_{fantastic}| + |w_.|$$

The groups have a tree structure, in that assigning zero values to the weights in a group corresponding to a higher-level constituent implies the same for those constituents that are dominated by it. This resembles the tree-guided group lasso in Kim and Xing (2008), although the leaf nodes in their tree represent tasks in multi-task regression.

Of course, in a corpus there are many parse trees (one per sentence, so the number

of parse trees is the number of sentences). Formally, the parse-tree regularizer:

$$\Omega_{tree}(\boldsymbol{w}) = \sum_{d=1}^{D} \sum_{s=1}^{S_d} \sum_{c=1}^{C_{d,s}} \lambda_{d,s,c} \|\boldsymbol{w}_{d,s,c}\|_2,$$

where $\lambda_{d,s,c} = \lambda_{glas} \times \sqrt{size(g_{d,s,c})}$, d ranges over (training) documents and c ranges over constituents in the parse of sentence s in document d. Similar to the sentence regularizer, the parse-tree regularizer operates on word tokens. Note that, since each word token is itself a constituent, the parse tree regularizer includes terms just like the lasso naturally, penalizing the absolute value of each word's weight in isolation. For the lasso-like penalty on each word, instead of defining the group weights to be $1 \times$ the number of tokens for each word type, we tune one group weight for all word types on a development data. As a result, besides λ_{glas}, we have an additional hyperparameter, denoted by λ_{las}.

To gain an intuition for this regularizer, consider the case where we apply the penalty only for a single tree (sentence), which for ease of exposition is assumed to never to use the same word more than once (i.e., $\|\boldsymbol{x}\|_\infty = 1$). Because it instantiates the tree-structured group lasso, the regularizer will require bigger constituents to be "included" (i.e., their words given nonzero weight) before smaller constituents can be included. The result is that some words may not be included. Of course, in some sentences, some words will occur more than once, and the parse tree regularizer instantiates groups for constituents in every sentence in the training corpus, and these groups may work against each other. The parse tree regularizer should therefore be understood as encouraging group behavior of syntactically grouped words, or sharing of information by syntactic neighbors.

In sentence level prediction tasks, such as sentence-level sentiment analysis, it is known that most constituents (especially those that correspond to shorter phrases) in a parse tree are uninformative (neutral sentiment). This was verified by Socher *et al.* (2013) when annotating phrases in a sentence for building the Stanford sentiment treebank. Our regularizer incorporates our prior expectation that most constituents should have no effect on prediction.

5.2.3 LDA Regularizer

Another type of structure to consider is topics. For example, if we want to predict whether a paper will be cited or not (Yogatama *et al.*, 2011), the model can perform better if it knows beforehand the collections of words that represent certain themes (e.g., in ACL papers, these might include machine translation, parsing, etc.). As a result, the model can focus on which topics will increase the probability of getting citations, and penalize weights for words in the same topic together, instead of treating each word separately.

We do this by inferring topics in the training corpus by estimating the latent Dirichlet allocation (LDA) model (Blei *et al.*, 2003). Note that LDA is an unsupervised method, so we can infer topical structures from *any* collection of documents that are considered related to the target corpus (e.g., training documents, text from the web, etc.). This contrasts with typical semi-supervised learning methods for text categorization that combine unlabeled and labeled data within a generative model, such as multinomial naïve Bayes, via expectation-maximization (Nigam *et al.*, 2000) or semi-supervised frequency estimate (Su *et al.*, 2011). Our method does not use unlabeled data to obtain more training documents or estimate the joint distributions of words better, but it allows the use of unlabeled data to induce topics. We leave comparison with other semi-supervised methods for future work.

There are many ways to associate inferred topics with group structure. In our experiments, we choose the R most probable words given a topic and create a group for them.[2] The LDA regularizer can be written as:

$$\Omega_{lda}(\boldsymbol{w}) = \sum_{k=1}^{K} \lambda_k \|\boldsymbol{w}_k\|_2,$$

where k ranges over the K topics. Similar to our earlier notation, $\boldsymbol{w}_k$ corresponds to the subvector of $\boldsymbol{w}$ such that the corresponding features are present in topic k. Note that in this case we can also have overlapping groups, since words can appear in the top R in many topics.

To gain an intuition for this regularizer, consider the toy example in Table 5.1. the

[2]Another possibility is to group the smallest set of words whose total probability given a topic amounts to P (e.g., 0.99). mass of a topic. Preliminary experiments found this not to work well.

Table 5.1: A toy example of $K = 4$ topics. The top $R = 5$ words in each topics are displayed. The LDA regularizer will construct four groups from these topics. The first group is $\langle w_{soccer}, w_{striker}, w_{midfielder}, w_{goal}, w_{defender} \rangle$, the second group is $\langle w_{injury}, w_{knee}, w_{ligament}, w_{shoulder}, w_{cruciate} \rangle$, etc. The third and fourth groups are constructed similarly. In this example, there are no words occurring in the top R of more than one topic, but that need not be the case in general.

$k = 1$	$k = 2$	$k = 3$	$k = 4$
soccer	injury	physics	monday
striker	knee	gravity	tuesday
midfielder	ligament	moon	april
goal	shoulder	sun	june
defender	cruciate	relativity	sunday

case where we have $K = 4$ topics and we select $R = 5$ top words from each topic. Supposed that we want to classify whether an article is a sports article or a science article. The regularizer might encourage the weights for the fourth topic's words toward zero, since they are less useful for the task. Additionally, the regularizer will penalize words in each of the other three groups collectively. Therefore, if (for example) *ligament* is deemed a useful feature for classifying an article to be about sports, then the other words in that topic will have a smaller effective penalty for getting nonzero weights—even weights of the opposite sign as $w_{ligament}$. It is important to distinguish this from unstructured regularizers such as the lasso, which penalize each word's weight on its own without regard for related word types.

Unlike the parse tree regularizer, the LDA regularizer is not tree structured. Since the lasso like penalty does not occur naturally in a non tree-structured regularizer, we add an additional lasso penalty for each word type (with hyperparameter λ_{las}) to also encourage weights of irrelevant words to go to zero. Our LDA regularizer is an instance of sparse group lasso (Friedman *et al.*, 2010).

5.2.4 Brown Cluster Regularizer

Brown clustering is a commonly used unsupervised method for grouping words into a hierarchy of hard clusters (Brown *et al.*, 1992). Because it uses local information, it tends to discover words that behave similar syntactically, though semantic groupings are often

evident, especially at the more fine-grained end of the hierarchy (see Figure 5.2 for an example).

We incorporate Brown clusters into a regularizer in a similar way to the topical word groups inferred using LDA in §5.2.3, but here we make use of the hierarchy. Specifically, we construct tree-structured groups, one per cluster (i.e., one per node in the hierarchy). Formally, the Brown cluster regularizer is:

$$\Omega_{brown}(\boldsymbol{w}) = \sum_{v=1}^{N} \lambda_v \|\boldsymbol{w}_v\|_2,$$

where v ranges over the N nodes in the Brown cluster tree. As a tree structured regularizer, this regularizer enforces constraints that a node v's group is given nonzero weights only if those nodes that dominate v (i.e., are on a path from v to the root) have their groups selected.

Consider a similar toy example to the LDA regularizer (sports vs. science) and the hierarchical clustering of words in Figure 5.2. In this case, the Brown cluster regularizer will create 17 groups, one for every node in the clustering tree. The regularizer for this tree (omitting the group coefficients and λ) is:

$$\Omega_{brown}(\boldsymbol{w}) = \sum_{i=0}^{7} \|\boldsymbol{w}_{v_i}\|_2 + |w_{goal}| + |w_{striker}| + |w_{midfielder}| + |w_{knee}| + |w_{injury}| \\ + |w_{gravity}| + |w_{moon}| + |w_{sun}|$$

The regularizer penalizes words in a cluster together, exploiting discovered syntactic relatedness, since Brown clustering tends to find syntactically similar words. Additionally, the regularizer can zero out weights of words corresponding to any of the internal nodes, such as v_7 if the words `monday` and `sunday` are deemed irrelevant to prediction.

Note that the regularizer already includes terms like the lasso naturally. Similar to the parse tree regularizer, for the lasso like penalty on each word, we tune one group weight for all word types on a development data with a hyperparameter λ_{las}.

A key difference between the Brown cluster regularizer and the parse tree regularizer is that there is only one tree for the Brown cluster regularizer, whereas the parse tree regularizer can have millions (one per sentence in the training data). The LDA and Brown cluster regularizers offer ways to incorporate unlabeled data, if we believe that

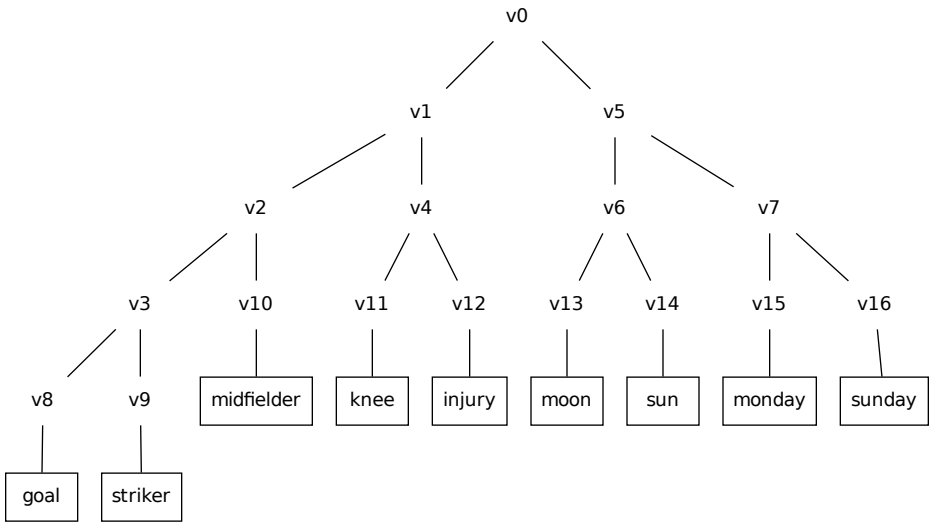


Figure 5.2: An toy example of Brown clusters for $N = 9$. The Brown cluster regularizer constructs 17 groups, one per node in for this tree, $v_0, v_1, \ldots, v_{16}$. v_0 contains 8 words, v_1 contains 5, etc. Note that the leaves, $v_8, v_9, \ldots, v_{16}$, each contain one word.

the unlabeled data can help us infer better topics or clusters. Note that the processes of learning topics or clusters, or parsing training data sentences, are a separate stage that precedes learning our predictive model.

5.3 Learning

Our approach to encoding lingustic knowledge in the form of structured regularizers introduces a new technical challenge. The nature of linguistic knowledge that we consider (e.g., sentences, constituents in parse trees, semantic concepts, etc.) can translate to thousands to millions of massively overlapping groups. As a result, learning can become very slow. In this section, we describe optimization methods that are suitable for general problems in this category.

5.3.1 ADMM for Sparse Group Lasso

There has been much work on optimization with overlapping group lasso penalty (Jacob *et al.*, 2009; Jenatton *et al.*, 2011b; Chen *et al.*, 2011; Qin and Goldfarb, 2012; Yuan *et al.*, 2013). Proximal methods (Beck and Teboulle, 2009; Duchi and Singer, 2009; Xiao, 2010; Bach *et al.*, 2011) offer one potential solution. For example, Martins *et al.* (2011a) introduced a proximal gradient algorithm for handling overlapping group lasso. We propose another optimization method that is more suitable for the case when we have massive numbers of overlapping groups (hundreds of thousands to millions of groups) based on the alternating directions method of multipliers (ADMM; Hestenes, 1969; Powell, 1969). Goldstein and Osher (2009) first proposed ADMM for sparse modeling. For a full review of ADMM, see Boyd *et al.* (2010).

We consider the following convex minimization problem:

$$\min_{\boldsymbol{w}} \Omega(\boldsymbol{w}) + \mathcal{L}(\boldsymbol{w}),$$

where Ω is a regularization function or a combination of them and $\mathcal{L}$ is a convex loss function. We denote the dimensionality of $\boldsymbol{w}$ by V. Note that the loss function can depend on other variables such as features $\boldsymbol{x}$ and response variable y. We hide these dependencies for brevity.

The central idea in ADMM is to break the optimization problem down into subproblems, each depending on a subset of the dimensions of $\boldsymbol{w}$. Each subproblem g receives a "copy" of the subvector of $\boldsymbol{w}$ it depends on, denoted $\boldsymbol{v}_g$. We then encode constraints forcing each $\boldsymbol{v}_g$ to "agree" with the global solution $\boldsymbol{w}$.

ADMM for overlapping group lasso only produces weakly sparse solutions,[3] for reasons we explain below. To achieve strong sparsity in the solution, which is desirable for high-dimensional data such as text, we couple the group lasso regularizer with a classic lasso regularizer. Therefore, the objective function to be minimized is:

$$\min_{\boldsymbol{w}} \Omega_{glas}(\boldsymbol{w}) + \Omega_{las}(\boldsymbol{w}) + \mathcal{L}(\boldsymbol{w})$$

Note that Ω_{glas} can be instantiated as Ω_{sen}, Ω_{tree}, Ω_{lda}, or Ω_{brown}, depending on the kind

[3]Weakly sparse methods (e.g., ridge) do not drive the feature weights exactly to zero, whereas strongly sparse methods (e.g., lasso) result in exact zeroes.

of linguistic structures we want to incorporate into the model.

In ADMM, we rewrite this as a constrained optimization problem:

$$\min_{\boldsymbol{w},\boldsymbol{v}} \Omega_{glas}(\boldsymbol{v}) + \Omega_{las}(\boldsymbol{w}) + \sum_{d=1}^{D} \mathcal{L}(\boldsymbol{x}_d, \boldsymbol{w}, y_d) \tag{5.1}$$
$$\text{s.t. } \boldsymbol{v} = \boldsymbol{M}\boldsymbol{w},$$

where $\boldsymbol{v}$ consists of copies of the elements of $\boldsymbol{w}$. We have $\boldsymbol{v}$ of size $N = \sum_{g=1}^{G} \text{size}(g)$. $\boldsymbol{M} \in \{0,1\}^{N \times V}$ is a matrix whose 1s link elements of $\boldsymbol{w}$ to their copies, where $\boldsymbol{w}$ is a V-dimensional vector. Notice that we work directly on $\boldsymbol{w}$ instead of the copies for the lasso like penalty, since it does not have overlaps and has its own hyperparameter, which we denote by λ_{las}. We also have overloaded notation somewhat; the group lasso regularizer applied to $\boldsymbol{v}$ is given by:

$$\Omega_{glas}(\boldsymbol{v}) = \lambda_{glas} \sum_{g=1}^{G} \lambda_g \|\boldsymbol{v}_g\|_2.$$

Let $\boldsymbol{u}$ be the Lagrange variables. The augmented Lagrangian of Equation 5.1 is:

$$\min_{\boldsymbol{w},\boldsymbol{v}} \Omega_{las}(\boldsymbol{w}) + \Omega_{glas}(\boldsymbol{v}) + \mathcal{L}(\boldsymbol{w}) + \boldsymbol{u}^\top(\boldsymbol{v} - \boldsymbol{M}\boldsymbol{w}) + \frac{\rho}{2}\|\boldsymbol{v} - \boldsymbol{M}\boldsymbol{w}\|_2^2$$

Note the introduction of a quadratic penalty.

ADMM proceeds by updating each of $\boldsymbol{w}$, $\boldsymbol{v}$, and $\boldsymbol{u}$ by solving, in turn, the following problems:

$$\min_{\boldsymbol{w}} \Omega_{las}(\boldsymbol{w}) + \mathcal{L}(\boldsymbol{w}) - \boldsymbol{u}^\top \boldsymbol{M}\boldsymbol{w} + \frac{\rho}{2}\|\boldsymbol{v} - \boldsymbol{M}\boldsymbol{w}\|_2^2 \tag{5.2}$$
$$\cong \min_{\boldsymbol{w}} \Omega_{las}(\boldsymbol{w}) + \mathcal{L}(\boldsymbol{w}) + \frac{\rho}{2}\left\|\boldsymbol{M}\boldsymbol{w} - \left(\boldsymbol{v} + \frac{\boldsymbol{u}}{\rho}\right)\right\|_2^2$$
$$\min_{\boldsymbol{v}} \Omega_{glas}(\boldsymbol{v}) + \boldsymbol{u}^\top \boldsymbol{v} + \frac{\rho}{2}\|\boldsymbol{v} - \boldsymbol{M}\boldsymbol{w}\|_2^2 \tag{5.3}$$
$$\cong \min_{\boldsymbol{v}} \Omega_{sen}(\boldsymbol{v}) + \frac{\rho}{2}\left\|\boldsymbol{v} - \left(\boldsymbol{M}\boldsymbol{w} - \frac{\boldsymbol{u}}{\rho}\right)\right\|_2^2$$
$$\boldsymbol{u} = \boldsymbol{u} + \rho(\boldsymbol{v} - \boldsymbol{M}\boldsymbol{w}) \tag{5.4}$$

We consider each in turn.

Update for $\boldsymbol{w}$. In Equation 5.2, we fix $\boldsymbol{v}$ and $\boldsymbol{u}$ and update $\boldsymbol{w}$. We denote the element of $\boldsymbol{v}$ corresponding to the nth copy by v_n, for $n \in \{1, \ldots, N\}$. We denote the number of copies of feature i in the corpus by N_i. Let $v_{i,n}$ denote the element of $\boldsymbol{v}$ corresponding to the nth copy of feature i for $n \in \{1, \ldots, N_i\}$. We index $\boldsymbol{u}$ similarly.

Note that the quadratic term in Equation 5.2 can be rewritten as

$$
\begin{aligned}
&\sum_{n=1}^{N} \left(w_{v_n} - (v_n + \tfrac{u_n}{\rho}) \right)^2 \\
&= \sum_{n=1}^{N} w_{v_n}^2 - 2w_{v_n}(v_n + \tfrac{u_n}{\rho}) + (v_n + \tfrac{u_n}{\rho})^2 \\
&= \sum_{i=1}^{V} \left(N_i w_i^2 - 2w_i \sum_{n=1}^{N_i} (v_{i,n} + \tfrac{u_{i,n}}{\rho}) + \sum_{n=1}^{N_i} (v_{i,n} + \tfrac{u_{i,n}}{\rho})^2 \right) \\
&= \sum_{i=1}^{V} N_i \left(w_i - \tfrac{1}{N_i} \sum_{n=1}^{N_i} (v_{i,n} + \tfrac{u_{i,n}}{\rho}) \right)^2 + \text{constant}(\boldsymbol{w}) \\
&= \sum_{i=1}^{V} N_i \left(w_i - \mu_i \right)^2 \\
&\text{where } \mu_i = \frac{1}{N_i} \sum_{n=1}^{N_i} (v_{i,n} + \tfrac{u_{i,n}}{\rho}).
\end{aligned}
$$

Intuitively, each variable is regularized towards a value near the mean of its corresponding copy variables. This is similar to some extent to ridge regularization. It now becomes clear why $\boldsymbol{w}$ will only be sparse in the limit (weakly sparse in practice) unless we add $\Omega_{las}(\boldsymbol{w})$ to the penalty, since the effective penalty is quadratic, as in ridge regression. This is the main reason to use sparse group lasso, if strong sparsity is required. (The reader may notice that when $\boldsymbol{\mu} = \mathbf{0}$ and $N_i = C$, this update is essentially equivalent to elastic net regression; Zou and Hastie, 2005, which penalizes $\boldsymbol{w}$ with a linear combination of Ω_{ridge} and Ω_{las}.) For this update, we apply a proximal gradient method (Bach *et al.*, 2011), since $\mathcal{L}(\boldsymbol{w}) + \frac{\rho}{2} \sum_{i=1}^{V} N_i (w_i - \mu_i)^2$ is convex and continuously differentiable, and $\Omega_{las}(\boldsymbol{w})$ is a convex function whose proximal operator (Moreau, 1963) can be evaluated efficiently. The proximal operator for $\Omega_{las}(\boldsymbol{w})$ is the soft-thresholding operator

Algorithm 2 ADMM for overlapping group lasso

Input: augmented Lagrangian variable ρ, regularization strengths λ_{glas} and λ_{las}
while stopping criterion not met **do**
  $\boldsymbol{w} = \arg\min_{\boldsymbol{w}} \Omega_{las}(\boldsymbol{w}) + \mathcal{L}(\boldsymbol{w}) + \frac{\rho}{2}\sum_{i=1}^{V} N_i(w_i - \mu_i)^2$
  for $g = 1$ **to** G **do**
    $\boldsymbol{v}_g = \text{prox}_{\Omega_{glas}, \frac{\lambda_g}{\rho}}(\boldsymbol{z}_g)$
  end for
  $\boldsymbol{u} = \boldsymbol{u} + \rho(\boldsymbol{v} - \boldsymbol{M}\boldsymbol{w})$
end while

(Donoho *et al.*, 2006):

$$[\text{prox}_{\Omega_{las}, \lambda_{las}}(\boldsymbol{w})]_j = \begin{cases} w_j - \lambda_{las} & \text{if } w_j > \lambda_{las} \\ 0 & \text{if } |w_j| \leq \lambda_{las} \\ w_j + \lambda_{las} & \text{if } w_j < -\lambda_{las} \end{cases}$$

Update for $\boldsymbol{v}$. Eq. 5.3 is the proximal operator of $\frac{1}{\rho}\Omega_{glas}$ applied to $\boldsymbol{M}\boldsymbol{w} - \frac{\boldsymbol{u}}{\rho}$. As such, it depends on the form of $\boldsymbol{M}$. Note that when applied to the collection of "copies" of the parameters, $\boldsymbol{v}$, Ω_{glas} no longer has overlapping groups. Define $\boldsymbol{M}_g$ as the rows of $\boldsymbol{M}$ corresponding to weight copies assigned to group g. Let $\boldsymbol{z}_g \triangleq \boldsymbol{M}_g\boldsymbol{w} - \frac{\boldsymbol{u}_g}{\rho}$. Denote $\lambda_g = \lambda_{glas}\sqrt{\text{size}(g)}$. The problem can be solved by applying the proximal operator used in non-overlapping group lasso to each subvector:

$$\begin{aligned} \boldsymbol{v}_g &= \text{prox}_{\Omega_{glas}, \frac{\lambda_g}{\rho}}(\boldsymbol{z}_g) \\ &= \begin{cases} \boldsymbol{0} & \text{if } \|\boldsymbol{z}_g\|_2 \leq \frac{\lambda_g}{\rho} \\ \dfrac{\|\boldsymbol{z}_g\|_2 - \frac{\lambda_g}{\rho}}{\|\boldsymbol{z}_g\|_2}\boldsymbol{z}_g & \text{otherwise.} \end{cases} \end{aligned}$$

For a tree structured regularizer, we can get speedups by working from the root node towards the leaf nodes when applying the proximal operator in the second step. If g is a node in a tree which is driven to zero, all of its children h that has $\lambda_h \leq \lambda_g$ will also be driven to zero.

Update for $\boldsymbol{u}$. Equation 5.4 is a simple update of the dual variable $\boldsymbol{u}$.

Algorithm 2 shows our ADMM algorithm for sparse overlapping group lasso.

5.3.2 Space and Time Efficiency

The learning algorithm is effective for large numbers of groups because each group operation and the $\boldsymbol{u}$ update (the second and third ADMM steps) can be done in *parallel*. The most expensive step is the minimization of $\boldsymbol{w}$. This is roughly as expensive as lasso or ridge methods since we can precompute $\boldsymbol{\mu}$, although we need to do the $\boldsymbol{w}$ minimization for every ADMM iteration.[4] Our model requires storing of two parameter vectors during learning: $\boldsymbol{w}$ and $\boldsymbol{v}$. Although the size of $\boldsymbol{v}$ is N, $\boldsymbol{v}$ is a sparse vector since most of the elements of $\boldsymbol{v}$ are driven to zero in the second ADMM step. Furthermore, $\boldsymbol{w}$ is also a sparse vector due to the Ω_{las} regularizer. The actual number of nonzero elements requiring storage will, of course, depend on λ_{glas}, λ_{las}, ρ, and the dataset.

5.3.3 Convergence and Stopping Criteria

We can show that Algorithm 2 is guaranteed to converge by simply noting that both $\mathcal{L}(\boldsymbol{w}) + \Omega_{las}(\boldsymbol{w})$ and $\Omega_{glas}(\boldsymbol{v})$ are closed, proper, and convex functions of $\boldsymbol{w}$ and $\boldsymbol{v}$ respectively;[5] and the function $\Omega_{glas}(\boldsymbol{v}) + \Omega_{las}(\boldsymbol{w}) + \mathcal{L}(\boldsymbol{w}) + \boldsymbol{u}^\top(\boldsymbol{v} - \boldsymbol{M}\boldsymbol{w})$ has a saddle point. As a result, our problem satisfies the two assumptions required for ADMM convergence (Boyd *et al.*, 2010). We can use the proof in Boyd *et al.* (2010) to show that Algorithm 2 has residual, objective, and dual variable convergence.

As noted there, ADMM is often slow to converge in practice, although tens of iterations are usually enough to obtain reasonably good solutions. In our experiments, we use relative changes in the ℓ_2 norm of the parameter vector $\boldsymbol{w}$ as our convergence criterion, and set the maximum number of iterations to 100. Other criteria such as primal and dual residual convergence and performance on development data can also be used to determine convergence of Algorithm 2 in practice.

[4] We minimize $\boldsymbol{w}$ to a relative convergence tolerance of 10^{-5}. The $\boldsymbol{w}$ minimization step need not be carried out to convergence at every iteration. Inexact ADMM (Boyd *et al.*, 2010), as this method is known, might provide speedups.

[5] Notice that λ_{sen} and λ_{las} translate into bounds on the norms of $\boldsymbol{v}$ and $\boldsymbol{w}$ since there is a one-to-one correspondence between a regularization constant and the parameter-vector norm due to the primal and dual representation of the objective function.

5.4 Experiments

5.4.1 Datasets

We use publicly available datasets to evaluate our model described in more detail below.

Topic classification. We consider four binary categorization tasks from the 20 Newsgroups dataset.[6] Each task involves categorizing a document according to two related categories: `comp.sys`: `ibm.pc.hardware` vs. `mac.hardware` and `rec.sport`: `baseball` vs. `hockey` and `sci`: `med` vs. `space` and `alt.atheism` vs. `soc.religion.christian`.

Sentiment analysis. One task in sentiment analysis is predicting the polarity of a piece of text, i.e., whether the author is favorably inclined toward a (usually known) subject of discussion or proposition (Pang and Lee, 2008). Sentiment analysis, even at the coarse level of polarity we consider here, can be confused by negation, stylistic use of irony, and other linguistic phenomena. Our sentiment analysis datasets consist of movie reviews from the Stanford sentiment treebank (Socher *et al.*, 2013),[7] and floor speeches by U.S. Congressmen alongside "yea"/"nay" votes on the bill under discussion (Thomas *et al.*, 2006).[8] For the Stanford sentiment treebank, we only predict binary classifications (positive or negative) and exclude neutral reviews.

Text-driven forecasting. Forecasting from text requires identifying textual correlates of a response variable revealed in the future, most of which will be weak and many of which will be spurious (Kogan *et al.*, 2009). We consider two such problems. The first one is predicting whether a scientific paper will be cited or not within three years of its publication (Yogatama *et al.*, 2011); the dataset comes from the ACL Anthology and consists of research papers from the Association for Computational Linguistics and citation data (Radev *et al.*, 2009). The second task is predicting whether a legislative bill will be recommended by a Congressional committee (Yano *et al.*, 2012).[9]

Table 5.2 summarizes statistics about the datasets used in our experiments. In total, we evaluate our method on eight binary classification tasks.

[6] `http://qwone.com/~jason/20Newsgroups`
[7] urlhttp://nlp.stanford.edu/sentiment/
[8] `http://www.cs.cornell.edu/~ainur/data.html`
[9] `http://www.ark.cs.cmu.edu/bills`

Table 5.2: Descriptive statistics about the datasets (number of documents and vocabulary size).

	Dataset	D	# Dev.	# Test	V
20Newsgroups	science	952	235	790	30,154
	sports	958	239	796	20,832
	religion	870	209	717	24,528
	computer	929	239	777	20,868
Sentiment analysis	movie	6,920	872	1,821	17,576
	vote	1,175	257	860	24,508
Forecasting	science	3,207	280	539	42,702
	bill	37,850	7,341	6,571	10,001

5.4.2 Setup

In all our experiments, we use counts of unigrams as our features, plus an additional bias term which is not regularized. We compare our new regularizers with state-of-the-art methods for document classification: lasso, ridge, and elastic net regularization.

Hyperparameters are tuned on a separate development dataset, using accuracy as the evaluation criterion. For lasso and ridge models, we choose λ from $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$. For elastic net, we perform grid search on the same set of values as ridge and lasso experiments for λ_{rid} and λ_{las}. For the sentence, Brown cluster, and LDA regularizers, we perform grid search on the same set of values as ridge and lasso experiments for $\rho, \lambda_{glas}, \lambda_{las}$. For the parse tree regularizer, because there are many more groups than other regularizers, we choose λ_{glas} from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10\}$, ρ and λ_{las} from the same set of values as ridge and lasso experiments. If there is a tie on development data we choose the model with the smallest number of nonzero weights.

When explicit sentence boundaries are not given, we use MxTerminator (Reynar and Ratnaparkhi, 1997)[10] to segment documents into sentences. We parsed all corpora using the Berkeley parser (Petrov and Klein, 2007).[11] For the LDA regularizers, we ran LDA[12] on training documents with $K = 1,000$ and $R = 10$. For the Brown cluster regularizers, we ran Brown clustering[13] on training documents with $5,000$ clusters for

[10] `ftp://ftp.cis.upenn.edu/pub/adwait/jmx`
[11] `https://code.google.com/p/berkeleyparser/`
[12] `http://www.cs.princeton.edu/~blei/lda-c/`
[13] `https://github.com/percyliang/brown-cluster`

the topic classification and sentiment analysis datasets, and 1,000 for the larger text forecasting datasets (since they are bigger datasets that took more time).

Table 5.3: Classification accuracies on various datasets. "m.f.c." is the most frequent class baseline. Boldface indicates better accuracy than the best unstructured regularizer model.

Task	**Dataset**	**Accuracy (%)**							
		m.f.c.	**lasso**	**ridge**	**elastic**	**sentence**	**parse tree**	**Brown**	**LDA**
20N	science	50.13	90.63	91.90	91.65	**96.20**	**92.66**	**93.04**	**93.67**
	sports	50.13	91.08	93.34	93.71	**95.10**	**93.09**	93.71	**94.97**
	religion	55.51	90.52	92.47	92.47	**92.75**	**94.98**	**92.89**	**93.03**
	computer	50.45	85.84	86.74	87.13	**90.86**	**88.93**	86.36	**89.45**
Sentiment	movie	50.08	78.03	80.45	80.40	**80.72**	**81.55**	80.34	78.36
	vote	58.37	73.14	72.79	72.79	**73.95**	**73.72**	66.86	73.14
Forecasting	science	50.28	64.00	66.79	66.23	**67.71**	66.42	**69.02**	**69.39**
	bill	87.40	88.36	87.70	**88.48**	88.11	87.98	88.20	88.27

Table 5.4: Model sizes (percentages of nonzero features in the resulting models) on various datasets.

Task	**Dataset**	**Model size (%)**							
		m.f.c.	**lasso**	**ridge**	**elastic**	**sentence**	**parse tree**	**Brown**	**LDA**
20N	science	-	1	100	34	12	2	42	9
	sports	-	2	100	15	3	3	16	9
	religion	-	.3	100	48	94	72	41	15
	computer	-	2	100	24	10	5	24	8
Sentiment	movie	-	10	100	54	83	87	59	12
	vote	-	2	100	44	6	2	30	4
Forecasting	science	-	31	100	43	99	9	50	90
	bill	-	7	100	7	8	37	7	7

Table 5.5: An article from News 20 dataset categorized under `comp.sys.mac.hardware`. Each line is a sentence identified by the sentence segmenter. There are twelve sentences in this article. Selected sentences in the learner's copy variables are highlighted in **blue and bold**. We also display the color-coded log-odds scores, as discussed in the text (**sentence**, **elastic**, **ridge**, **lasso**) based on removing each sentence for each competing model. We only display scores that are greater than 10^{-3} in absolute values.

Sentence	Negative Positive
from : *anonymized*	
subject : accelerating the macplus ... ;)	(0.05)
lines : 15 we ' re about ready to take a bold step into the 90s around here by accelerating our rather large collection of stock macplus computers .	(0.07) (0.03) (0.02) (0.02)
yes indeed , difficult to comprehend why anyone would want to accelerate a macplus, but that's another story .	(0.06) (0.02) (0.02) (0.04)
suffuce it to say , we can get accelerators easier than new machines .	(0.01)
hey , i don ' t make the rules ...	(0.01)
anyway , on to the purpose of this post: i ' m looking for info on macplus acelerators .	(0.04) (0.01)
so far , i ' ve found some lit on the novy accelerator and the micrmac multispeed accelartor .	(0.02) (0.02) (0.02) (0.04)
both look acceptable , but i would like to hear from anyone who has tried these .	(–0.01)
also , if someone would recommend another accelerator for the macplus , i ' d like to hear about it .	(0.06) (0.03) (0.02) (0.06)
thanks for any time and effort you expend on this !	(–0.01) (–0.01) (–0.01)
karl	

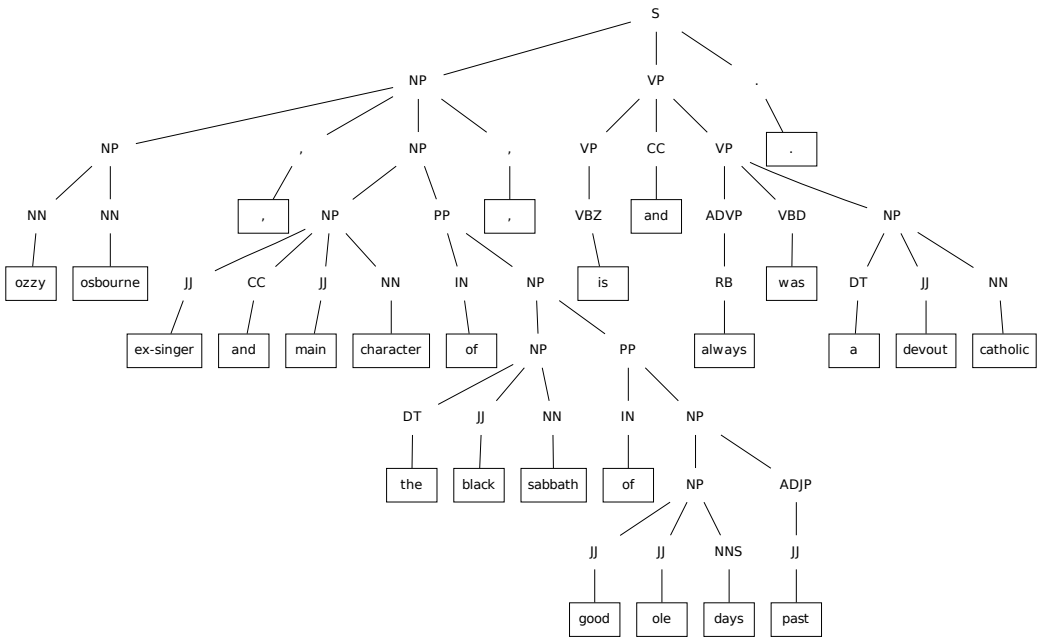


Figure 5.3: An example sentence from the 20N:religion dataset and its parse tree output. The NP node corresponding to the `ozzy osbourne` phrase was driven to zero in our model.

$= 0$	"acknowledgment": workshop arpa program session darpa research papers spoken technology systems "document metadata": university references proceedings abstract work introduction new been research both "equation": pr w h probability wi gram context z probabilities complete "translation": translation target source german english length alignment hypothesis translations position
$\neq 0$	"translation": **korean** translation english rules sentences parsing input evaluation machine verb "speech processing": speaker **identification** topic recognition recognizer models acoustic test vocabulary independent "parsing": parser parsing **probabilistic** prediction parse **pearl** edges chart phase theory "classification": documents learning accuracy **bayes classification** wt document **naive** method selection

Table 5.6: Examples of LDA regularizer-removed and -selected groups (in v) in the forecasting scientific articles dataset. Words with weights (in w) of magnitude greater than 10^{-3} are highlighted in **red** (not cited) and **blue** (cited).

$= 0$	underwater industrial spotted hit reaped rejuvenated destroyed stretched undertake shake run seeing developing tingles diminishing launching finding investigating receiving maintaining adds engage explains builds
$\neq 0$	failure reproductive **ignition** reproduction **cyanamid planetary nikola** fertility **astronomical** geophysical # **lunar** cometary **supplying** astronautical **magnetic** atmospheric **std** underwater **hpr** wordscan exclusively aneutronic industrial peoples **obsessive** congenital **rare simple bowel** hereditary breast

Table 5.7: Examples of Brown regularizer-removed and -selected groups (in v) in the 20N:science task. # denotes any numeral. Words with weights (in w) of magnitude greater than 10^{-3} are highlighted in **red** (space) and **blue** (medical).

5.4.3 Results

Table 5.3 shows the results of our experiments on the eight datasets. The results demonstrate the superiority of structured regularizers. One of them achieved the best result on all but one dataset.[14] It is also worth noting that in most cases all variants of the structured regularizers outperformed lasso, ridge, and elastic net.

Recall that, during learning, we make a copy in $\boldsymbol{v}$ of each weight in $\boldsymbol{w}$ for each corresponding word token. $\boldsymbol{w}$ is used to make predictions; it seeks to be a consensus among all of the $\boldsymbol{v}_g$. In practice, with many overlapping groups, the constraint $\boldsymbol{v} = \boldsymbol{M}\boldsymbol{w}$ is rarely satisfied exactly. Inspecting which $\boldsymbol{v}_g$ are nonzero can give some insight into what is learned, by showing which groups are treated as "relevant" by the algorithm (i.e., "selected" vs. "removed"). For each of the proposed regularizers, we inspect the model for tasks in which it performed reasonably well.

Sentence regularizer We can see that the sentence regularizer performed well on the topic categorization tasks. Table 5.5 shows an example of selected sentences in a training instance from the 20N:computer task. We can see that in this particular case, the learner selected informative sentences and removed uninformative ones. We also show the log-odds scores for removing each sentence. The log-odds score is defined here as the log of the model probability of the class label for an instance (document) using all sentences minus the log of the probability of the class label using all sentences except one. Intuitively, the scores indicate how much the sentence affects the model's decision. From the log-odds scores, we can see our model tends to make its decision mostly based on the sentences it "selects." We observed that in some cases (e.g., religion, vote, etc.) the model selected most sentences, whereas in other cases (e.g., dvd, electronics, etc.) the model excluded many sentences. We believe that the flexibility of our model to include or exclude sentences through validation on development data contributes to the performance improvements.

Parse tree regularizer The parse tree regularizer performed the best for the movie review dataset. The task is to predict sentence-level sentiment, so each training example

[14] This "bill" dataset, where they offered no improvement, is the largest by far (37,850 documents), and therefore the one where regularizers should matter the least. Note that the differences are small across regularizers for this dataset.

is a sentence. Since constituent-level annotations are available for this dataset, we only constructed groups for neutral constituents (i.e., we drive neutral constituents to zero during training).It has been shown that syntactic information is helpful for sentence-level predictions (Socher *et al.*, 2013), so the parse tree regularizer is naturally suitable for this task. For the parse tree regularizer, we inspect the model for the 20N:religion task, in which it also performed well. We observed that the model included most of the sentences (root node groups), but in some cases removed phrases from the parse trees. For example, Fig. 5.3 shows an example sentence and parts that are driven to zero by our regularizer. For this sentence, the model decided the verb phrase `ozzy osbourne` is not helpful for the atheist vs. christian prediction, hence driven to zero.

LDA regularizer For the LDA regularizer, we inspect zero and nonzero groups (topics) in the forecasting scientific articles task. The task is to predict whether an article will be cited or not within three years after publication. Regularizers that exploit the knowledge of semantic relations (e.g., topical categories), such as the LDA regularizer, are therefore suitable for this type of prediction. In this task, we observed that 642 out of 1,000 topics are driven to zero by our model. Table 5.6 shows examples of zero and nonzero topics for the dev.-tuned hyperparameter values. We can see that in this particular case, the model kept meaningful topics such as machine translation and speech recognition, and discarded general topics that are not correlated with the content of the papers (e.g., acknowledgement, document metadata, equation, etc.). Interestingly, we also observed that there are translation topics that were driven to zero. Since the task is to predict whether an article will be cited or not, perhaps words in this translation topics are more general of machine translation papers, so they are not indicative of citations. Treating topics as groups contributes to the performance improvements since it informs the model about semantic relations among the words in the corpus. Note that when there are two words that are equally predictive, the lasso tends to pick one and exclude the other. Grouping features can help the model choose the "right" word that is better for generalization.

Brown regularizer For the Brown cluster regularizer, we inspect the model from the 20N:science task. 771 out of 5,775 groups were driven to zero for the best model tuned

on the development set. Examples of zero and nonzero groups are shown in Table 5.7. Similar to the LDA example, the groups that were driven to zero tend to contain generic words that are not relevant to the predictions. We can also see the tree structure effect in the regularizer. The group { `underwater`, `industrial` } was driven to zero, but not once it combined with other words such as `hpr`, `std`, `peoples`. We also observed that the groups often contain words that are not intuitive that they should belong to the same groups (the group structures are not as good as the LDA regularizer). This explains why the LDA regularizer tends to perform better on our collection of datasets. Note that we ran Brown clustering on the training documents; running it on a larger collection of (unlabeled) documents relevant to the prediction task is worth exploring in future work (i.e., semi-supervised learning).

Topic and cluster features. Another way to incorporate LDA topics and Brown clusters into a linear model is by adding them as additional features. For the 20N datasets, we also ran lasso, ridge, and elastic net with *additional* LDA topic and Brown cluster features.[15] Note that these new baselines use more features than our model. We can also add these additional features to our model and treat them as regular features (i.e., they do not belong to any groups and are regularized with standard regularizer such as the lasso penalty). The results in Table 5.8 show that for these datasets, models that incorporate this information through structured regularizers outperformed models that encode this information as additional features in 4 out 4 of cases (LDA) and 2 out of 4 cases (Brown). Sparse models with Brown clusters appear to overfit badly; recall that the clusters were learned on only the training data—clusters from a larger dataset would likely give stronger results. Of course, better performance might also be achieved by incorporating new features as well as using structured regularizers.

These results demonstrate that linguistic structure in the data can be used to improve bag-of-words models, through structured regularization. State-of-the-art approaches to some of these problems have used additional features and representations (Yessenalina *et al.*, 2010; Socher *et al.*, 2013). For example, for the vote sentiment analysis datasets, latent variable models of Yessenalina *et al.* (2010) achieved a superior result of 77.67%. To do so, they sacrificed convexity and had to rely on side information for initialization.

[15]For LDA, we took the top 10 words in a topic as a feature. For Brown clusters, we add a cluster as an additional feature if its size is less than 50.

Table 5.8: Classification accuracies on the 20N datasets for lasso, ridge, and elastic net models with additional LDA features (top) and Brown cluster features (bottom). The last column shows structured regularized models from Table 5.3.

Dataset	**+ LDA features**			**LDA**
	lasso	**ridge**	**elastic**	**reg.**
science	90.63	91.90	91.90	**93.67**
sports	91.33	93.47	93.84	**94.97**
religion	91.35	92.47	91.35	**93.03**
computer	85.20	86.87	86.35	**88.42**
Dataset	**+ Brown features**			**Brown**
	lasso	**ridge**	**elastic**	**reg.**
science	86.96	90.51	91.14	**93.04**
sports	82.66	88.94	85.43	**93.71**
religion	94.98	**96.93**	**96.93**	92.89
computer	55.72	**96.65**	67.57	86.36

Our experimental focus is on a controlled comparison between regularizers for a fixed model family (the simplest available, linear with bag-of-words features). However, the improvements offered by our regularization methods can be applied in future work to other model families with more carefully engineered features, metadata features (especially important in forecasting), latent variables, etc. In particular, note that the penalties for other kinds of weights (e.g., metadata) can be penalized conventionally, or incorporated into the structured regularization where it makes sense to do so (e.g., n-grams, as in Nelakanti et al., 2013).

Runtime In terms of running time (wall clock), our model is slightly slower than standard regularizers. For example, for the sports dataset, learning models with the best hyperparameter value(s) for lasso, ridge, and elastic net took 27, 18, and 10 seconds, respectively, on an Intel Xeon CPU E5645 2.40 GHz machine with 8 processors and 24 GB RAM. Our sentence regularizer model with the best hyperparameter values took 33 seconds to reach convergence. As mentioned previously, the major drawback is the need to do grid search for each of the hyperparameters: λ_{sen}, λ_{las}, and ρ, whereas lasso and ridge only have one hyperparameter and elastic net has two hyperparameters. However, note that this grid search can also be done in parallel, since they are not dependent on each other, so given enough processors our method is only marginally slower than

standard regularizers.

In future work, smaller or larger structures—as well as combinations of them (e.g., sentence+Brown+LDA)—might be used. One particularly interesting direction is defining each document as a group. It has an intuitive connection with support vectors in support vector machines (Cortes and Vapnik, 1995), although further investigation is required to determine its effectiveness.

5.5 Conclusion

In this chapter, we introduced a new sparse overlapping group lasso regularizer for text modeling inspired by the structure inherent in linguistic data. We also showed how to efficiently perform learning for sparse group lasso with thousands to millions of overlapping groups using the alternating direction method of multipliers. We empirically demonstrated that our model consistently outperformed competing models on various datasets for various real-world document categorization tasks.

Chapter 6

Future Work

In this thesis, we presented instantiations of sparse models for text categorization, word embeddings, and temporal models of text and empirically demonstrated benefits of sparse models in these problems. We also showed efficient optimization methods for the proposed instantiations of sparse models. There are many interesting directions that can be explored further in the future. We discuss four of them in the following.

Structured Sparse Coding with Overcomplete Representation In Chapter 4, we propose to use sparse coding to learn word representations and obtain promising results. However, our approach is still limited to low-dimensional embeddings of words due to computational complexity of the learning procedure. Sparse, overcomplete representations have been shown to increase separability and be more stable to noise in many applications (Olshausen and Field, 1997; Lewicki and Sejnowski, 2000; Donoho *et al.*, 2006), including in learning word representations in NLP (Faruqui *et al.*, 2015). However, due the high-dimensional nature of text data, overcomplete representations of word embeddings are created by first inducing low-dimensional embeddings of words in a two-step process. Future work can explore overcomplete representations for word embeddings directly from high-dimensional word representations. In order to do this, we need to develop a very efficient optimization method for structured sparse coding that can handle hundreds of thousands of words, hundreds of thousands of contexts, and hundreds of thousands (or possibly millions) of code space.

Structured Sparsity in the Output Space Another interesting direction of future work is the case when we can define structures over the output (label) space. For example, Kim and Xing (2008) consider the case when the output space can be represented as a tree in the multi-task setting. In statistical text analysis, there are many problems where we can define structures over the output space. For example, in named entity classification, the output space might form a tree structure. Consider the sentence `President Obama presented Dr. Maha Al-Muneef with an International Women of Courage award.` We assume that the named entities have been tagged, and there are three named entities in this sentence: `President Obama`, `Dr. Maha Al-Muneef`, and `International Women of Courage`. We want to classifty `President Obama` into our entity classes. For example, it can belong to three classes: `human`, `president`, `Barack Obama`. However, we know that an entity in class `Barack Obama` and `president` has to also be of class `human`, so `human` can be the root of this label tree. Note that in our case, the labels for a named entity do not necessarily correspond to a non-branching part of the label tree. For example, if there are classes `professor`, `male`, and `female`, we know that a president can be either male or female, but so does a professor. Designing a sparse model that can make use of this type of structure in the output space is useful to create a better named entity classifier. An important challenge is that the number of classes can be huge, so we also need a very efficient learning algorithm.

Automatic Learning of Group Structures In this thesis, we show how we can define linguistically-motivated group structures for the group lasso to encode linguistic knowledge into statistical NLP models. One promising direction is to let a machine learning model discover what kind of grouping structure is useful for a given task. Bayesian optimization (Brochu *et al.*, 2010; Hutter *et al.*, 2011; Bergstra *et al.*, 2011; Snoek *et al.*, 2012; Yogatama and Smith, 2015) is one promising approach to automatically learn group structures. We can treat the problem of finding the best partition of the feature space as a hyperparameter optimization problem and apply Bayesian optimization techniques to efficiently search this space. Treating the problem of finding the best grouping structures as a hyperparameter problem opens up many possibilities, including transfer learning and multitask learning of grouping strucrures from many related text datasets or NLP tasks (Bardenet *et al.*, 2013; Swersky *et al.*, 2013; Yogatama and Mann, 2014).

Structured Sparse Regularization of Deep Models Deep learning methods have enjoyed considerable success in natural language processing (Li *et al.*, 2014; Kalchbrenner *et al.*, 2014; Johnson and Zhang, 2015; Dyer *et al.*, 2015). Deep models typically have a large number of parameters, owing to their hierarchical architectures, so regularization becomes a crucial component in succesfully training deep models. Dropout (Srivastava *et al.*, 2014) is a popular regularization technique in deep models. It operates by randomly dropping hidden units along with their connections from the neural networks during training. Recently, there has been an attempt to induce more structures in the hidden units using nested dropout (Rippel *et al.*, 2014), where the hidden units are grouped into nested sets before performing dropout. This technique produces representations similar to the structured penalty we introduced in Chapter 4. One drawback of dropout is that it slows down training procedure considerably, since we ignore some hidden units at each iteration. It is interesting to see empirical and theoretical comparisons between structured sparse penalties and nested dropout for learning deep models. Since the resulting representations exhibit similar sparsity patterns, it is possible that applying structured sparse regularizers provides a more efficient way to regularize hidden units of deep networks.

Bibliography

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proc. of NAACL-HLT*.

Amaldi, E. and Kann, V. (1998). On the approximation of minimizing non zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, **209**, 237–260.

Andrew, G. and Gao, J. (2007). Scalable training of l1-regularized log-linear models. In *Proc. of ICML*.

Angelosante, D. and Giannakis, G. B. (2009). RLS-weighted lasso for adaptive estimation of sparse signals. In *Proc. of ICASSP*.

Angelosante, D., Giannakis, G. B., and Grossi, E. (2009). Compressed sensing of time-varying signals. In *Proc. of ICDSP*.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2011). *Convex Optimization with Sparsity-Inducing Norms*. The MIT Press.

Bamman, D., Dyer, C., and Smith, N. A. (2014). Distributed representations of situated language. In *Proc. of ACL*.

Bardenet, R., Brendel, M., Kegl, B., and Sebag, M. (2013). Collaborative hyperparameter tuning. In *Proc. of ICML*.

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.

Belmonte, M. A. G., Koop, G., and Korobilis, D. (2012). Hierarchical shrinkage in time-varying parameter models. Working paper.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, **3**, 1137–1155.

Bergstra, J., Bardenet, R., Bengio, Y., and Kegl, B. (2011). Algorithms for hyper-parameter optimization. In *Proc. of NIPS*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, **3**, 993–1022.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (2008). *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, **3**(1), 1–122.

Brochu, E., Brochu, T., and de Freitas, N. (2010). A Bayesian interactive optimization approach to procedural animation design. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, **18**, 467–479.

Bruni, E., Boleda, G., Baroni, M., and Tran, N.-K. (2012). Distributional semantics in technicolor. In *Proc. of ACL*.

Caron, F., Bornn, L., and Doucet, A. (2012). Sparsity-promoting bayesian dynamic linear models. arXiv 1203.0106.

Charles, A. S. and Rozell, C. J. (2012). Re-weighted ℓ_1 dynamic filtering for time-varying sparse signal estimation. arXiv 1208.0325.

Chen, S. F. and Rosenfeld, R. (2000). A survey of smoothing techniques for me models. *IEEE Transactions on Speech and Audio Processing*, **8**(1), 37–50.

Chen, X., Lin, Q., Kim, S., Carbonell, J. G., and Xing, E. P. (2011). Smoothing proximal gradient method for general structured sparse learning. In *Proc. of UAI*.

Collins, A. M. and Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behaviour*, **8**, 240–247.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuska, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, **12**, 2461–2505.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**(3), 273–297.

Davis, G., Mallat, S., and Avellaneda, M. (1997). Greedy adaptive approximation. *Journal of Constructive Approximation*, **13**, 57–98.

Della Pietra, S., Della Pietra, V., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 380–393.

Donoho, D. L., Elad, M., and Temlyakov, V. N. (2006). Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, **52**(1).

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, **10**, 2899–2934.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**, 2121–2159.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Eisenstein, J., Smith, N. A., and Xing, E. P. (2011a). Discovering sociolinguistic associations with structured sparsity. In *Proc. of ACL*.

Eisenstein, J., Ahmed, A., and Xing, E. P. (2011b). Sparse additive generative models of text. In *Proc. of ICML*.

Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proc. of EACL*.

Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., and Smith, N. A. (2015). Sparse binary word vector representations. In *Proc. of ACL*.

Figueiredo, M. A. T. (2002). Adaptive sparseness using jeffreys' prior. In *Proc. of NIPS*.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, **20**(1), 116–131.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, **3**, 1289–1305.

Friedman, J., Hastie, T., and Tibshiran, R. (2010). A note on the group lasso and a sparse group lasso. Technical report, Stanford University.

Fyshe, A., Talukdar, P. P., Murphy, B., and Mitchell, T. M. (2014). Interpretable semantic vectors from a joint model of brain- and text- based meaning. In *Proc. of ACL*.

Goldstein, T. and Osher, S. (2009). The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, **2**(2), 323–343.

Gutmann, M. and Hyvarinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. of AISTATS*.

Halawi, G. and Dror, G. (2014). The word relatedness mturk-771 test collection.

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, **4**, 303–320.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, **12**(1), 55–67.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proc. of ACL*.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*.

Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proc. of ICML*.

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. (2011a). Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, **12**, 2297–2334.

Jenatton, R., Audibert, J.-Y., and Bach, F. (2011b). Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, **12**, 2777–2824.

Johnson, R. and Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In *Proc. of NAACL*.

Joshi, M., Das, D., Gimpel, K., and Smith, N. A. (2010). Movie reviews and revenues: An experiment in text regression. In *Proc. of NAACL*.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proc. of ACL*.

Kim, S. and Xing, E. P. (2008). Feature selection via block-regularized regression. In *Proc. of UAI*.

Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., and Smith, N. A. (2009). Predicting risk from financial reports with regression. In *Proc. of HLT-NAACL*.

Kong, L., Rush, A. M., and Smith, N. A. (2015). Transforming dependencies into phrase structures. In *Proc. of NAACL-HLT*.

Lebret, R. and Collobert, R. (2014). Word embeddings through hellinger PCA. In *Proc. of EACL*.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse coding algorithms. In *Proc. of NIPS*.

Lee, H., Raina, R., Teichman, A., and Ng, A. Y. (2009). Exponential family sparse coding with application to self-taught learning. In *Proc. of IJCAI*.

Levy, O. and Goldberg, Y. (2014). Neural word embeddings as implicit matrix factorization. In *Proc. of NIPS*.

Lewicki, M. and Sejnowski, T. (2000). Learning overcomplete representations. *Neural computation*, **12**(2), 337–365.

Li, J., Li, R., and Hovy, E. (2014). Recursive deep models for discourse parsing. In *Proc. of EMNLP*.

Lin, T.-H. and Kung, H. (2014). Stable and efficient representation learning with non-negativity constraints. In *Proc. of ICML*.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming B*, **45**(3), 503–528.

Liu, J. and Ye, J. (2010). Moreau-yosida regularization for grouped tree structure learning. In *Proc. of NIPS*.

Luong, M.-T., Socher, R., and Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. In *Proc. of CONLL*.

Lv, X., Bi, G., and Wan, C. (2011). The group lasso for stable recovery of block-sparse signal representations. *IEEE Transactions on Signal Processing*, **59**(4), 1371–1382.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, **11**, 19–60.

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011a). Online learning of structured predictors with multiple kernels. In *Proc. of AISTATS*.

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011b). Structured sparsity in structured prediction. In *Proc. of EMNLP*.

Martins, A. F. T., Yogatama, D., Smith, N. A., and Figueiredo, M. A. T. (2014). Structured sparsity in natural language processing: Models, algorithms, and applications. In *Tutorial at EACL*.

Mikolov, T., Martin, K., Burget, L., Cernocky, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proc. of Interspeech*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013b). Efficient estimation of word representations in vector space. In *Proc. of ICLR Workshop*.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, **38**(11), 39–41.

Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, **6**(1), 1–28.

Mnih, A. and Hinton, G. (2008). A scalable hierarchical distributed language model. In *Proc. of NIPS*.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *Proc. of ICML*.

Moreau, J. J. (1963). Fonctions convexes duales et points proximaux dans un espace hilbertien. *CR Acad. Sci. Paris Sńer. A Math*, **255**, 2897–2899.

Murphy, B., Talukdar, P., and Mitchell, T. (2012). Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proc. of COLING*.

Nakajima, J. and West, M. (2012). Bayesian analysis of latent threshold dynamic models. *Journal of Business and Economic Statistics*.

Nelakanti, A., Archambeau, C., Mairal, J., Bach, F., and Bouchard, G. (2013). Structured penalties for log-linear language models. In *Proc. of EMNLP*.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning*, **39**(2-3), 103–134.

Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, **37**(23), 3311 – 3325.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, **2**(1–2), 1–135.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Proc. of HLT-NAACL*.

Petrov, S. and Klein, D. (2008). Sparse multi-scale grammars for discriminative latent variable parsing. In *Proc. of EMNLP*.

Powell, M. J. D. (1969). A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press.

Qin, Z. T. and Goldfarb, D. (2012). Structured sparsity via alternating direction methods. *Journal of Machine Learning Research*, **13**, 1435–1468.

Radev, D. R., Muthukrishnan, P., and Qazvinian, V. (2009). The ACL anthology network corpus. In *Proc. of ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*.

Radinsky, K., Agichtein, E., Gabrilovich, E., and Markovitch, S. (2011). A word at a time: Computing word relatedness using temporal semantic analysis. In *Proc. of WWW*.

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In *Proc. of NIPS*.

Raposo, A., Mendes, M., and Marques, J. F. (2012). The hierarchical organization of semantic memory: Executive function in the processing of superordinate concepts. *NeuroImage*, **59**, 1870–1878.

Ratnaparkhi, A., Roukos, S., and Ward, R. T. (1994). A maximum entropy model for parsing. In *Proc. of ICSLP*.

Reynar, J. C. and Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proc. of the Fifth Conference on Applied Natural Language Processing*.

Rippel, O., Gilbert, M. A., and Adams, R. P. (2014). Learning ordered representations with nested dropout. In *Proc. of ICML*.

Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Communications of the ACM*, **8**(10), 627–633.

Schutze, H. (1998). Automatic word sense discrimination. *Computational Linguistics - Special issue on word sense disambiguation*, **24**(1), 97–123.

Snoek, J., Larrochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proc. of NIPS*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, **15**, 72–101.

Sra, S. (2012). Scalable nonconvex inexact proximal splitting. In *Proc. of NIPS*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**, 1929–1958.

Stojnic, M., Parvaresh, F., and Hassibi, B. (2009). On the reconstruction of block-sparse signals with an optimal number of measurements. *Signal Processing, IEEE Transactions on*, **57**(8), 3075–3085.

Su, J., Sayyad-Shirabad, J., and Matwin, S. (2011). Large scale text classiøcation using semi-supervised multinomial naive bayes. In *Proc. of ICML*.

Swersky, K., Snoek, J., and Adams, R. P. (2013). Multi-task bayesian optimization. In *Proc. of NIPS*.

Tackstrom, O. and McDonald, R. (2011). Discovering fine-grained sentiment with latent variable structured prediction models. In *Proc. of ECIR*.

Thomas, M., Pang, B., and Lee, L. (2006). Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proc. of EMNLP*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society B*, **58**(1), 267–288.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society B*, **67**(1), 91–108.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL*.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, **37**, 141–188.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, **9**, 2579–2605.

Volpi, L. (2003). Eigenvalues and eigenvectors of tridiagonal uniform matrices.

Xiao, L. (2010). Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, **11**, 2543–2596.

Yang, D. and Powers, D. M. W. (2006). Verb similarity on the taxonomy of wordnet. In *Proc. of GWC*.

Yano, T., Smith, N. A., and Wilkerson, J. D. (2012). Textual predictors of bill survival in congressional committees. In *Proc. of NAACL*.

Yessenalina, A., Yue, Y., and Cardie, C. (2010). Multi-level structured models for document sentiment classification. In *Proc. of EMNLP*.

Yogatama, D. and Mann, G. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *Proc. of AISTATS*.

Yogatama, D. and Smith, N. A. (2014a). Linguistic structured sparsity in text categorization. In *Proc. of ACL*.

Yogatama, D. and Smith, N. A. (2014b). Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In *Proc. of ICML*.

Yogatama, D. and Smith, N. A. (2015). Bayesian optimization of text representations. In *Proc. of EMNLP*.

Yogatama, D., Heilman, M., O'Connor, B., Dyer, C., Routledge, B. R., and Smith, N. A. (2011). Predicting a scientific community's response to an article. In *Proc. of EMNLP*.

Yogatama, D., Routledge, B. R., and Smith, N. A. (2013). A sparse and adaptive prior for time-dependent model parameters. arXiv 1310.2627.

Yogatama, D., Faruqui, M., Dyer, C., and Smith, N. A. (2015). Learning word representations with hierarchical sparse coding. In *Proc. of ICML*.

Yuan, L., Liu, J., and Ye, J. (2013). Efficient methods for overlapping group lasso. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(9), 2104–2116.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, **68**(1), 49–67.

Zhang, Y. and Yeung, D.-Y. (2010). A convex formulation for learning task relationships in multi-task learning. In *Proc. of UAI*.

Zhao, P., Rocha, G., and Yu, B. (2009). The composite and absolute penalties for grouped and hierarchical variable selection. *The Annals of Statistics*, **37**(6A), 3468–3497.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, **67**, 301–320.

Zweig, G. and Burges, C. J. C. (2011). The microsoft research sentence completion challenge. Technical report, Microsoft Research Technical Report MSR-TR-2011-129.