A Survey of Dimensionality Reduction Techniques for Natural Language

John Blitzer BLITZER@CIS.UPENN.EDU
Written Preliminary Examination 2
Department of Computer and Information Science
University of Pennsylvania

Contents

Abstract

Machine learning methods for natural language use features consisting of words or combinations of words to fit statistical models of linguistic phenomena. The discrete input spaces resulting from these features often have hundreds of thousands or millions of dimensions, and estimating reliable statistics of these features from limited amounts of training data is difficult. One technique for alleviating this data sparseness is to induce a low-dimensional representation of the original feature vector. Learning algorithms can then estimate statistics of the low-dimensional vectors more reliably than the original high-dimensional vectors.

We begin with a brief summary of two standard techniques for dimensionality reduction of language: latent semantic analysis (Deerwester et al., 1990) and probabilistic latent semantic analysis (Hofmann, 1999). We then give an overview of three more recent models for dimensionality reduction of language. Sufficient dimensionality reduction (Globerson and Tishby, 2003) is an information-theoretic dimensionality reduction technique which learns a low-rank factorization of the natural parameters of a joint multinomial distribution. The neural probabilistic language model (Bengio et al., 2003) learns a nonlinear mapping in order to give a low-dimensional representation for whole phrases simultaneously. Finally structural learning (Ando and Zhang, 2005) is a dimensionality reduction technique for semisupervised learning. Given a target prediction task with labeled and unlabeled data, structural learning finds a low dimensional feature space using the unlabeled data. This new feature space in turn yields a better target predictor.

We give a unified treatment of the five methods, focusing on their formulation as optimization problems. The three newer methods use more realistic loss functions and more flexible parameterizations than the two older ones, and as a result they are able to capture language phenomena more accurately. Finally, we discuss the disadvantages of these models and suggest directions for future research.

1. Introduction

Machine learning methods for natural language use features consisting of words or combinations of words to fit statistical models of linguistic phenomena. Generative models, which are the staple of language modeling (Chen and Goodman, 1996) and phrase-structure parsing (Collins, 1999) estimate a joint probability distribution over words, phrases, or sentences. For a sentence of length 10 and a simple language model that uses only word level features over a 50,000 word vocabulary, the resulting space of mean-parameterized multinomial distributions is a 10^{40}-dimensional simplex. Discriminative models for language explicitly represent words and word combinations as sparse binary feature vectors. These models regularly perform well for tagging (Ratnaparkhi, 1996) and chunking (Sha and Pereira, 2003) and have recently seen success in dependency parsing (McDonald, 2005) and language modeling (Roark et al., 2004). Parameter vectors for discriminative models are typically millions of dimensions in size.

Estimating so many parameters from limited data is a difficult task. In practice both generative and discriminative models employ the technique of backoff, which uses low-dimensional discrete features such as parts of speech, word prefix and suffix, or semantic role in place of or in addition to high-dimensional features like words or phrases. Since these low-dimensional features are specified by the designers of the system, this effectively amounts to a kind of manual dimensionality reduction. These features are often quite effective, and part of speech tags in particular are an essential part of most models. Unfortunately, such manual low-dimensional features are time-consuming to create. Furthermore, since they require a human to design, they aren't adaptable to new situations and domains.

One approach that is potentially less costly is to automatically induce low-dimensional representations from data. As with manual low-dimensional features, these features are less sparse, and their statistics are easier to estimate from limited data. They have the added advantage of requiring minimal human effort. The simplest such features are hard word clusters (Brown et al., 1992; Emami and Jelinek, 2005). In this framework a word is mapped to exactly one class. This makes cluster features easy to integrate with existing generative and discriminative models. Cluster features have several disadvantages, though. Hard clusters necessarily cut off words from part of their semantic neighborhoods (Schütze, 1993). Finding a hard clustering also requires a difficult combinatorial optimization, often resulting in models which require more computation time and perform worse.

In this survey we focus on models that pose dimensionality reduction as a continuous optimization problem. These methods treat dimensionality reduction similarly to other problems in machine learning for language. In particular they can incorporate features in a similar way, and they can be analyzed in terms of of the loss functions they minimize. We begin with the well-known methods of latent semantic analysis (LSA) (Deerwester et al., 1990) and probabilistic latent semantic analysis (PLSA) (Hofmann, 1999). LSA is an application of the singular value decomposition to co-occurrence data. PLSA rectifies some of the problems with LSA by creating a generative model for language as multinomial data.

The majority of the survey is devoted to three new models which offer different perspectives and improved performance when compared with LSA and PLSA. Sufficient dimensionality reduction (SDR) (Globerson and Tishby, 2003) learns the most informative real-valued low-dimensional features for describing co-occurrence data. The problem formulation they give leads to an exponential model with factored natural parameters, which ties SDR to the flexible and widely successful maximum entropy models for language. The neural probabilistic language model (Bengio et al., 2003) (NPLM) extracts continuous features similar to those of SDR but combines them nonlinearly in a neural network. This allows them to induce joint representations for unobserved histories in a language model, which in turn gives them significant improvements over the best methods on one of the most well-studied problems in natural language processing. Ando and Zhang (2005) use dimensionality reduction on unlabeled data to induce features for semisupervised learning. Given a target prediction problem with labeled and unlabeled data, Ando and Zhang (2005) first create many auxiliary problems which are related to the target problem but can be trained on unlabeled data. They then train auxiliary predictors for each of the auxiliary problems and learn a low-rank factorization of the parameter space for these predictors. This low-rank factorization yields a new, low-dimensional feature space for training a target predictor on the labeled data. Ando and Zhang (2005) report consistent improvements over state-of-the-art discriminative models using structural learning.

The remained of this paper is structured as follows: In the next section we introduce our notation and terminology. Section 3 discusses the LSA and PLSA models. The next three sections address the three new models. Finally section 7 analyzes these models and suggests further improvements.

2. Notation and Terminology

We choose to describe latent semantic analysis, probabilistic latent semantic analysis, and sufficient dimensionality reduction in the context of information retrieval (henceforth also IR). While these models are potentially more general, Deerwester et al. (1990), Hofmann (1999), and Globerson and Tishby (2003) all use information retrieval as their main task, so we focus on IR in this survey. The information retrieval task is characterized by the co-occurrence of documents and words. In this case we will denote the co-occurrence matrix as

$$\mathbf{X} \in \mathbb{R}^{V \times D} ,$$

where D is the number of documents and V is the vocabulary size. When formulating dimensionality reduction as a matrix factorization we will write the factors as

$$\mathbf{X} \approx \mathbf{\Phi}\mathbf{\Psi}' , \qquad \mathbf{\Phi} \in \mathbb{R}^{V \times k} \ , \ \mathbf{\Psi} \in \mathbb{R}^{D \times k} \ .$$

$\mathbf{\Psi}'$ indicates the transpose of the matrix $\mathbf{\Psi}$ and k is the rank of the low-rank factorization. For real-valued representations of documents and words, we will write $\phi(d)$ and $\psi(w)$, where $\phi_i(d)$ is the ith element in the vector representation for a document d.

The PLSA, SDR, and neural probabilistic language models all use explicit probability distributions to model linguistic data. For the information retrieval models, we will write $p(w, d)$ and $p(w|d)$ to indicate the joint and conditional probabilities of words given documents. The bag-of-words model, which LSA, PLSA, and SDR all assume can be written as

$$p(w_1, \ldots, w_m|d) = \prod_{i=1}^{m} p(w_i|d) \ .$$

$\tilde{p}$ indicates the empirical probability distribution induced by counting the discrete events in a training set. For language modeling we will write $p\left(w_t|w_{t-1}, \ldots, w_{t-n+1}\right)$ to indicate the conditional n-gram probability of word t given the previous $n-1$ words.

Finally, in structural learning we will need a notion of a weight vector for a linear classifier, which we will write in bold as $\mathbf{w}$, and the matrix whose columns are weight vectors we will write as $\mathbf{W}$. Other notation should be clear in the context in which it appears.

3. Information Retrieval Models: LSA and PLSA

Methods for dimensionality reduction appeared early in the field of information retrieval (Baker, 1962; Ossorio, 1966). The central task of information retrieval is to retrieve a small, relevant set of documents based on a (usually short) query. In order to perform this task well, it is essential to have an accurate measure of similarity between documents and queries. The most common model for document-query similarity is the vector space model. In this model, documents are points in a V-dimensional vector space. Each component corresponds to a single word, and the value of the component for a document is a function of the number of times the word has occurred in the document. Queries are treated as pseudo-documents in the same space, and similarity is measured with the inner product between the (normalized) document and query vectors. A detailed description of the field of information retrieval is well beyond the scope of this survey, but see Salton (1989) for a good introduction.

Retrieval errors occur either when an irrelevant document is mistakenly retrieved for a query or when a relevant document is not retrieved. Deerwester et al. (1990) motivate dimensionality reduction using the language phenomenon of synonymy. A document containing the word “automobile” may be relevant to a query containing the word “car”, even if the document itself does not contain the word “car”. Latent semantic analysis (Deerwester et al., 1990) derives its name from the assumption that a corpus has a small latent set of semantic topics from which words can be chosen. A document consists of words from some mixture of these topics. Deerwester et al. (1990) also assume that queries are intended to refer to topics. Now suppose that both “automobile” and “car” have the same topic. If we can create a good representation of the underlying topics of a corpus, then we can retrieve a document containing “automobile” from a query containg “car”.

What constitutes such a good representation? For information retrieval in the vector space model, we want similar documents to have a high scalar product in the new vector space. LSA and PLSA are linear factorization methods that project documents into a lower-dimensional space. In general it is impossible to guarantee that a linear matrix factorization can find a good representation. Under assumptions about the structure of the topics and the corpus, though, it is possible to show that linear factorizations can find good representations (Papadimitriou et al., 1998; Ando and Lee, 2001). Furthermore, as we shall see in the next two subsections, LSA and PLSA can achieve good factorizations in practice.

3.1 Latent Semantic Analysis

Deerwester et al. (1990) observe that if we represent documents using the vector space model, we can reduce the dimensionality of the document space by computing the singular value decomposition of the matrix whose columns are document vectors and truncating all but the top singular values. We will refer to this “term-by-document” matrix as $\mathbf{X}$. Then the latent semantic analysis is $[\mathbf{\Phi} \;\; \mathbf{S} \;\; \mathbf{\Psi}'] = \text{SVD}(\mathbf{X}, k)$, where

$$\underset{V \times D}{\left[\begin{array}{c} \\ \mathbf{X} \\ \\ \end{array} \right]} = \underset{V \times k}{\left[\begin{array}{c} \\ \mathbf{\Phi} \\ \\ \end{array} \right]} \underset{k \times k}{[\,\mathbf{S}\,]} \underset{k \times D}{\left[\quad \mathbf{\Psi}' \quad \right]}$$

That is, we discard all but the top k singular values of $\mathbf{X}$. $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are orthonormal matrices, so that the columns of $\mathbf{\Phi}$ span a latent semantic subspace. Finally, consider the the example from Deerwester et al. (1990), depicted in figure 1. In this example there are two topics: human computer interfaces and graph theory, and projecting the documents onto the top two singular vectors reveals the underlying latent structure.

Once we have performed LSA, we can compute similarities between queries and documents by first projecting them onto the semantic subspace and then computing the scalar products. That is, the similarity between a query $\mathbf{q}$ and the ith document in the corpus $\mathbf{X}_{[:,i]}$ can be written as

$$\text{sim}(\mathbf{X}_{[:,i]}, \mathbf{q}) = (\mathbf{S}^{1/2}\mathbf{\Phi}'\mathbf{q})'(\mathbf{S}^{1/2}\mathbf{\Phi}'\mathbf{X}_{[:,i]})$$

(a) Term-by-document matrix X

Terms	**Documents**					
	u1	u2	u3	g1	g2	g3
human	1	0	0	0	0	0
interface	1	0	1	0	0	0
computer	1	1	0	0	0	0
user	0	1	1	0	0	0
system	0	1	1	0	0	0
response	0	1	0	0	0	0
time	0	1	0	0	0	0
EPS	0	0	1	0	0	0
survey	0	1	0	0	1	0
trees	0	0	0	1	0	1
graph	0	0	0	0	1	1
minors	0	0	0	0	1	1

(b) Scatter plot of u1-u3 (red circles) and g1-g3 (blue triangles) projected onto the top two latent semantic basis vectors.

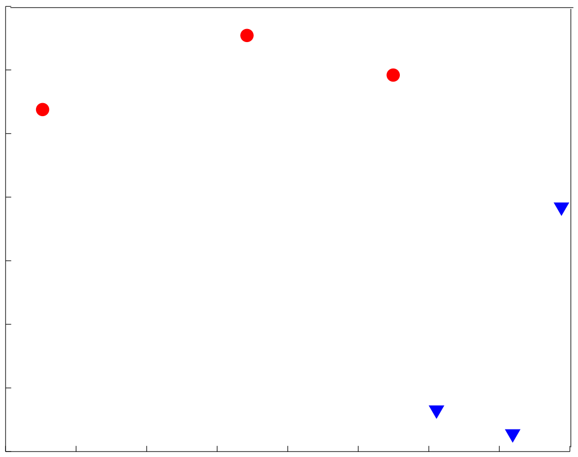

Figure 1: An example of a 2-dimensional latent semantic space from Deerwester et al. (1990). Three of the documents discuss user interfaces (u1-u3), and three discuss graph theory (g1-g3).

3.1.1 Latent Semantic Analysis as an Optimization Problem

The singular value decomposition can also be formulated as an optimization problem. We want to find a low-rank approximation to $\mathbf{X}$, which minimizes the squared loss, or equivalently the Frobenius distance. We follow Srebro (2004) by absorbing the diagonal matrix $\mathbf{S}$ into $\mathbf{\Phi}$. We wish to minmize the cost function

$$J(\mathbf{\Phi}, \mathbf{\Psi}) = \left|\left|\mathbf{X} - \mathbf{\Phi}\mathbf{\Psi}'\right|\right|_F^2 \qquad \text{s.t.} \;\; \mathbf{\Phi} \in \mathbb{R}^{V\times k},\;\; \mathbf{\Psi} \in \mathbb{R}^{D\times k},\;\; \mathbf{\Phi}'\mathbf{\Phi} = \mathbf{\Lambda},\;\; \mathbf{\Psi}'\mathbf{\Psi} = I$$

with $\mathbf{\Lambda}$ diagonal. Computing partial derivates of J, we have

$$\begin{aligned} \frac{\delta J}{\delta \mathbf{\Phi}} &= 2(\mathbf{X} - \mathbf{\Phi}\mathbf{\Psi}')\mathbf{\Psi} \\ \frac{\delta J}{\delta \mathbf{\Psi}} &= 2(\mathbf{X}' - \mathbf{\Psi}\mathbf{\Phi}')\mathbf{\Phi} \end{aligned}$$

Now substituting for $\mathbf{\Phi}$ in the partial with respect to $\mathbf{\Psi}$, we see that solutions are of the form

$$\mathbf{\Psi}\mathbf{\Lambda} = \mathbf{X}'\mathbf{X}\mathbf{\Psi} \; .$$

That is, the solutions for $\mathbf{\Psi}$ have the eigenvectors of $\mathbf{X}'\mathbf{X}$ as columns. Substituting back into the partial with respect to $\mathbf{\Phi}$ we have that the solutions have the corresponding eigenvectors of $\mathbf{X}\mathbf{X}'$ as columns, weighted by their eigenvalues. It remains to show that the global minimum of $\mathbf{\Phi}, \mathbf{\Psi}$ have only the top eigenvectors as columns. We know that we can write

$\mathbf{X}$ as the product of factors $\hat{\mathbf{\Phi}}\hat{\mathbf{\Psi}}'$, where the columns of $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Psi}}$ are the eigenvectors of the covariance matrix $\mathbf{XX}'$ and gram matrix $\mathbf{X}'\mathbf{X}$ respectively. Assume that[1] $D = V = m$. Finally, without loss of generality, let the first k columns of $\hat{\mathbf{\Phi}}$ equal $\mathbf{\Phi}$, $\hat{\mathbf{\Phi}}_{[:,1:k]} = \mathbf{\Phi}$, and similarly for $\hat{\mathbf{\Psi}}$.

$$\begin{aligned}
\sum_{d,w} \left|\left|\mathbf{X} - \mathbf{\Phi}\mathbf{\Psi}'\right|\right|_F^2 &= \sum_{d,w} \left(\sum_{\alpha=1}^{m} \hat{\mathbf{\Phi}}_{d,\alpha} \hat{\mathbf{\Psi}}_{w,\alpha} - \sum_{\alpha=1}^{k} \mathbf{\Phi}_{d,\alpha} \mathbf{\Psi}_{w,\alpha} \right)^2 \\
&= \left|\left| \hat{\mathbf{\Phi}}_{[:,(k+1):m]} \hat{\mathbf{\Psi}}'_{[(k+1):m,:]} \right|\right|_F^2 \\
&= \mathrm{tr}\left(\hat{\mathbf{\Phi}}_{[:,(k+1):m]} \hat{\mathbf{\Psi}}'_{[(k+1):m,:]} \hat{\mathbf{\Psi}}_{[:,(k+1):m]} \hat{\mathbf{\Phi}}'_{[(k+1):m,:]} \right) \\
&= \mathrm{tr}\left(\hat{\mathbf{\Phi}}_{[:,(k+1):m]} \hat{\mathbf{\Phi}}'_{[(k+1):m,:]} \right) \\
&= \sum_{\alpha=k+1}^{m} \lambda_\alpha
\end{aligned}$$

That is, the error is the sum of the eigenvalues whose eigenvectors are not in the solution. So we see that the minimum must occur where the columns of $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are the top eigenvectors of the covariance and gram matrices, respectively.

3.1.2 Problems with the Squared Loss

The squared loss can be shown to be equivalent to maximizing the likelihood of the data under a Gaussian with a low-rank covariance matrix. Several authors have observed that for discrete data such as text, this Gaussian assumption may be inappropriate (Hofmann, 1999; Collins et al., 2002). Probabilistic latent semantic analysis and sufficient dimensionality reduction address this by computing low rank factorizations that maximize the likelihood of the observed data under a multinomial model.

3.2 Probabilistic Latent Semantic Indexing

Probabilistic latent semantic indexing is an application of the aspect model (Hofmann and Puzicha, 1998) to modeling terms and documents (Hofmann, 1999). The technique is "probabilistic" since it is motivated by modeling the conditional multinomial distribution $p(w|d)$ of words given documents. The PLSA model is a mixture model with the following form:

$$p(w|d) = \sum_{z=1}^{k} \mathbf{\Phi}_{d,z} \mathbf{\Psi}_{w,z}, \quad \text{subject to } \forall d \sum_{z} \mathbf{\Phi}_{d,z} = 1, \ \ \forall z \sum_{w} \mathbf{\Psi}_{w,z} = 1 \quad .$$

Figure 2 gives a graphical model representation for the aspect model. The distributions $p(z|d)$ are parameterized by the columns of the matrix $\mathbf{\Phi}$, and the distributions $p(w|z)$ are parameterized by the columns of the matrix $\mathbf{\Psi}$. Just as LSA minimizes the Frobenius distance between the term-by-document matrix $\mathbf{X}$ and the factorization $\mathbf{\Phi}\mathbf{\Psi}'$, PLSA minimizes the Kullback-Leibler divergence between $\mathbf{X}$ and a factored matrix $\mathbf{UV}'$, defined as

1. The result holds for rectangular matrices as well, but requires a bit more algebra

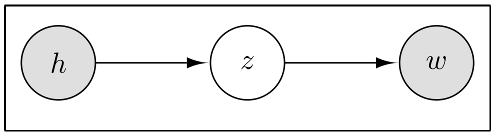

Figure 2: PLSA graphical model

$$\begin{aligned} KL(\mathbf{X}, \mathbf{\Phi}\mathbf{\Psi}') &= \sum_{d,w} \mathbf{X}_{d,w} \log \frac{\mathbf{X}_{d,w}}{(\mathbf{\Phi}\mathbf{\Psi}')_{d,w}} \\ &= \sum_{d,w} \mathbf{X}_{d,w} \left[\log \mathbf{X}_{d,w} - \log \left(\mathbf{\Phi}\mathbf{\Psi}' \right)_{d,w} \right] \end{aligned}$$

Notice that the first term in the summation above does not depend on the factors $\mathbf{\Phi}, \mathbf{\Psi}$. Thus we see that minimizing the KL divergence is the same as maximizing the data likelihood

$$\min_{\mathbf{\Phi},\mathbf{\Psi}} KL(\mathbf{X}, \mathbf{\Phi}\mathbf{\Psi}') = \max_{\mathbf{\Phi},\mathbf{\Psi}} \sum_{d,w} \mathbf{X}_{d,w} \log \sum_{z} \mathbf{\Phi}_{d,z} \mathbf{\Psi}_{w,z} \quad .$$

Like the SVD minimization problem, the rank constraint makes this minimization problem non-convex (Srebro, 2004). Unlike the minimizing the Frobenius distance, however, this is not an eigenvalue problem. Thus we must content ourselves with numerically finding a local maximum of the log-likelihood. Because of its mixture form, we can give an expectation maximization algorithm for determining a locally optimal set of parameters $\mathbf{\Phi}, \mathbf{\Psi}$ (Saul and Pereira, 1997; Hofmann, 1999). The M-step is closed form, and the optimization typically converges quickly, but see (Salakhutdinov et al., 2003) for more direct gradient-based optimizers for this model.

3.2.1 The PLSA model in practice

Using the PLSA model for information retrieval turns out to be somewhat intricate. This is because the PLSA model is not a complete generative model for the space of documents and words. When we receive a query $\mathbf{q}$, we ideally would compute $p(w|\mathbf{q})$ and compare this vector to the vectors for each document $p(w|d)$. But we don't know the paramters for $p(z|\mathbf{q})$, so we can't compute the model probabilities $p(w|\mathbf{q})$. Equivalently, there is no row in our matrix $\mathbf{\Phi}$ corresponding to the query $\mathbf{q}$. Hofmann (1999) suggests a technique he calls "folding in", which re-estimates just the vector $p(w|\mathbf{q})$ to maximize the likelihood of the observed query words. This entails a short, "one-sided" EM update for every query. "Folding in" is perhaps the largest drawback to PLSA.

4. Sufficient Dimensionality Reduction

Sufficient dimensionality reduction (Globerson and Tishby, 2003) is a method for finding a low-rank approximation to the sufficient statistics of an exponential family distribution.

Globerson and Tishby (2003) give many interpretations of sufficient dimensionality reduction (SDR), but in this report we will focus on two. The first is that for a joint distribution $p(d, w)$, SDR finds the real-valued features of a document $\phi(d)$ that are most informative about its words w. The second is that SDR minimizes the Kullback-Leibler divergence between an empirical co-occurrence matrix $\mathbf{X}$ and a factorization $\frac{\exp(\mathbf{\Phi}\mathbf{\Psi}')}{Z}$, where the matrix exponential function is element-by-element and Z is a normalization constant. Globerson and Tishby (2003) also discuss a method for addressing generalization error and finite sample effects by using Cramer-Rao bounds (Cover and Thomas, 1991) on the Fisher information. We discuss this is subsection 4.2. Finally in subsection 4.3 we give some applications of SDR and briefly address its relationship to PLSA and LSA.

4.1 Motivation and Interpretation

The goal of sufficient dimensionality reduction is to model co-occurrence data $p(d, w)$, just as in PLSA and LSA. For each document d there are several features $\phi(d)$ which may be relevant to co-occurring words w. These features might represent topic, style, and so on. Given an empirical sample $\mathbf{X}$ on which we can measure the expected value of these features, one natural question is "What is the 'most probable' distribution that has the same expected values for each of the features as the empirical distribution?"

Globerson and Tishby (2003) suggest a minimum mutual information distribution based on the following observation: The information about the random variable w captured by the expected values of the measured features $\phi(d)$ cannot be larger than the mutual information of *any* joint distribution $p(d, w)$ consistent with these expected values, since every such joint distribution can produce those observations. This leads to the following definition:

Definition 1 *The information in the measurement of the expected values of $\phi(d)$ on $\tilde{p}(d, w)$ is*

$$\begin{aligned} I_M(\phi(d), \tilde{p}) \quad &\equiv \quad \min_{p(d,w)} I\left(p(x, y)\right) \\ & \qquad s.t. \quad \langle \phi(d) \rangle_{p(x|y)} = \langle \phi(d) \rangle_{\tilde{p}(x|y)}\,, \quad p(d) = \tilde{p}(d), \quad p(w) = \tilde{p}(w) \end{aligned}$$

$\tilde{p}$ indicates the empirical distribution from our data matrix $\mathbf{X}$. The constraints on the marginal distributions are easy to estimate from small samples, and as we shall see, they will lead us to a convenient and familiar form for optimization.

As we said before, the task of SDR is to choose the features $\hat{\phi}(d)$ which are most informative. This leads us to the variational optimization problem

$$\hat{\phi}(d) = \operatorname*{argmax}_{\phi(d)} I_M\left(\phi(d), \tilde{p}(d, w)\right) \quad .$$

One key insight when deriving the solution to this optimization problem is to relate the minimum mutual information distribution to the well-known principle of maximum entropy (James, 1957). Globerson and Tishby (2003) observe that the distribution with minimum mutual information subject to marginal constraints also maximizes the joint entropy subject to the same marginal constraints.

$$I\left(p(d, w)\right) \quad = \quad \sum_{d,w} p(d, w) \log \frac{p(d, w)}{p(d)p(w)}$$

$$
\begin{aligned}
&= \sum_{d,w} p(d,w)\log p(d,w) - \sum_{d} p(d)\log p(d) - \sum_{d} p(w)\log p(w) \\
&= \sum_{d,w} p(d,w)\log p(d,w) - \sum_{d} \tilde{p}(d)\log \tilde{p}(d) - \sum_{d} \tilde{p}(w)\log \tilde{p}(w) \quad \text{(from constraints)}
\end{aligned}
$$

Observe that in the final equation, the first summation is the negative entropy. The second and third summations are constants which don't depend on p. Now we can proceed to derive the form of the maximum entropy distribution, subject to the constraints on ϕ and the marginals (Cover and Thomas, 1991). We begin by forming the Lagrangian of the maximum entropy optimization problem

$$
\begin{aligned}
L(p,\psi(w),A(x),B(y)) \quad &= \quad -\sum_{d,w} p(d,w)\log p(d,w) - \\
&\sum_{w}\psi(w)\left(\sum_{d} p(d,w)\phi(d) - \sum_{d}\tilde{p}(d,w)\phi(d)\right) + \lambda_0\left(\sum_{d,w} p(d,w) - 1\right) + \\
&\sum_{d} A(d)\left(\sum_{w} p(d,w) - \sum_{w}\tilde{p}(d,w)\right) + \sum_{w} B(w)\left(\sum_{w} p(d,w) - \sum_{w}\tilde{p}(d,w)\right) \ .
\end{aligned}
$$

The maximum entropy optimization problem is concave, so we can write write its unconstrained dual (Boyd and Vandenberghe, 2004) as

$$
\min_{\psi(w),A(x),B(y)} \max_{p} L(p,\psi(w),A(x),B(y)) \ .
$$

Taking the functional derviative with respect to $p(d,w)$, we can solve analytically to see that the optimal $\hat{p}(d,w)$ has the following form

$$
\hat{p}(d,w) = \frac{1}{Z}\exp\left(\sum_{i}\phi_i(d)\psi_i(w) + A(d) + B(w)\right) \ ,
$$

where Z is a normalizing constant[2]. With this in hand we can substitute for $p(d,w)$ in L, yielding the dual minimization problem

$$
\begin{aligned}
&\min_{\psi(w),A(d),B(w)} L(\psi(w),A(d),B(w)) \quad = \\
&\min_{\psi(w),A(d),B(w)} -\sum_{d,w}\hat{p}(d,w)\left(\sum_{i}\phi_i(d)\psi_i(w) + A(d) + B(w)\right) + \sum_{d,w}\hat{p}(d,w)\log Z + \\
&\sum_{d,w}\hat{p}(d,w)\left(\sum_{i}\phi_i(d)\psi_i(w) + A(x) + B(y)\right) - \sum_{d,w}\tilde{p}(d,w)\left(\sum_{i}\phi_i(d)\psi_i(w) + A(x) + B(y)\right) \ .
\end{aligned}
$$

Canceling and combining terms yields

$$
\min_{\psi(w),A(x),B(y)} -\sum_{d,w}\tilde{p}(d,w)\log\frac{1}{Z}\exp\left(\sum_{i}\phi_i(d)\psi_i(w) + A(x) + B(y)\right) \ .
$$

2. In fact we can absorb the normalization into the marginal constraints, but the formulation chosen by Globerson and Tishby (2003) yields the more familiar exponential form.

Finally note that the parameters which solve the dual problem above maximize the log-likelihood. Combining this observation with the original maximization problem over $\phi(d)$ gives us the final optimization problem from Globerson and Tishby (2003):

$$\underset{\phi(d),\psi(w),A(d),B(w)}{\text{argmax}} \sum_{d,w} \tilde{p}(d,w) \log \frac{1}{Z} \exp\left(\sum_i \phi_i(d)\psi_i(w) + A(d) + B(w)\right) .$$

Globerson and Tishby (2003) propose an alternating projection algorithm for finding the likelihood-maximizing parameters $\phi(d)$ and $\psi(w)$, but we will not address this algorithm here. Standard gradient-based techniques can perform 10-15 times faster than the algorithm outlined in the paper(Globerson, 2006).

4.2 Cramer-Rao Bounds for ϕ and ψ

One of the goals for dimensionality reduction of language is to provide a compact representation from a finite sample, which generalizes well to new, unseen samples from the same distribution. The Cramer-Rao inequality (Cover and Thomas, 1991) provides a lower bound on the variance of an estimator for the parameter vector θ of a parametric distribution. Let $\tilde{\theta}(x^n)$ be the estimator of θ on the n-element i.i.d. sample x^n. Then the Cramer-Rao bound for $\tilde{\theta}(x^n)$ is

$$Var(\tilde{\theta}_i(x^n)) \geq \frac{1}{J_{i,i}(\theta)} , \qquad J_{i,j}(\theta) = \left\langle -\frac{\delta^2 \log p(x|\theta)}{\delta\theta_i \delta\theta_j} \right\rangle .$$

$J(\theta)$ is the Fisher information matrix for the parameter θ. Computing the Fisher information matrix for the SDR parameters $\phi_i(d)$ and $\psi_i(w)$ reveals the relationships

$$J_{i,j}(\psi) = Cov(\tilde{\phi}(d^n)) , \qquad J_{i,j}(\phi) = Cov(\tilde{\psi}(w^n)) .$$

In particular along the diagonal, we have the following Cramer-Rao bounds:

$$Var(\tilde{\phi}(x^n)) \geq \frac{1}{Var(\psi)n} , \qquad Var(\tilde{\psi}(y^n)) \geq \frac{1}{Var(\phi)n} .$$

That is, we can lower-bound the variance of $\tilde{\phi}(x^n)$, by the sample variance of ψ and vice versa. For exponential families, we know that the Cramer-Rao bound is tight, and there are ϕ and ψ that achieve the Cramer-Rao bounds (Muller-Funk et al., 1989). Of course, real-world data is not guaranteed to come from an exponential-family distribution.

The Cramer-Rao bounds given by Globerson and Tishby (2003) provide *lower* bounds on the variance (and thus the error) of the estimators for ϕ and ψ. If the lower bound is high, then we can reasonably despair of finding a good estimator. A lower bound is useful only when it is pessimistic, though. If the Fisher information is high, then we can tell nothing about the goodness of our own estimators for ϕ and ψ.

We might wish to *upper* bound the error of our estimate of the distribution $\hat{p}(d,w)$ with respect to the true distribution $p_{\text{true}}(d,w)$. The most natural notion of error is the L_1 error

$$\sum_{d,w} |\hat{p}(d,w) - p_{\text{true}}(d,w)| .$$

Unfortunately, finding a distribution which is close in the L_1 sense for an arbitrary distribution can require samples polynomial in the number of possible states of the underlying discrete space (Batu et al., 2000). Even for pairwise co-occurrence data, this quantity is polynomial in DV, which could be quite large.

4.3 Applications of SDR

Globerson and Tishby (2003) report improved information retrieval performance using SDR, when compared with locally linear embedding (Roweis and Saul, 2000) and LSA. They also show that using only a few SDR features can achieve almost the same performance as using the whole feature set for document classification.

4.4 Relationship to PLSA and LSA

Like PLSA and LSA, SDR can be viewed as a matrix factorization technique. Globerson and Tishby (2003) observe that we can write the optimization as

$$\underset{\mathbf{\Phi},\mathbf{\Psi}}{\operatorname{argmin}}\, KL\left(\mathbf{X}, \frac{\exp(\mathbf{\Phi}\mathbf{\Psi}')}{Z}\right) \qquad \mathbf{\Phi} \in \mathbb{R}^{D\times k}, \quad \mathbf{\Psi} \in \mathbb{R}^{m\times k}$$

where KL is defined as before and the function exp is element-by-element. Unlike the PLSA optimization, the matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are unconstrained, but the partition function effectively couples the elements of the matrix together. Ultimately, though, the most important difference between PLSA and SDR, both of which model joint multinomial distributions is the parameterization. Sufficient dimensionality reduction chooses to factorize the natural parameters of the multinomial, and as we will see in the next section, this affords a flexibility that can be used to model linguistic relationships beyond pairs of words.

5. The Neural Probabilistic Language Model

PLSA, LSA, and SDR are all models for reducing the dimensionality of pairwise co-occurrence data. While this can be effective in bag-of-words information retrieval, it ignores an important part of natural language semantics. Linguistic meaning is built up from many smaller pieces, and when modeling language whole phrases are important. The neural probabilistic language model (NPLM) (Bengio et al., 2003) learns a reduced-dimensional representation for whole phrases simultaneously. We first introduce n-gram language modeling in section 5.1. Then we discuss the neural probabilistic language model architecture in section 5.2. Finally, we address the effects of nonlinearity, speedups, and disadvantages to the NPLM in section 5.4.

5.1 Language Modeling

A statistical language model predicts the probability that a sentence will occur. For many natural language processing systems which have text as output, such as machine translation (Brown et al., 1990) and automatic speech recognition (Jelinek, 1997), statistical language modeling is a central component. More recently techniques from language modeling have been shown to help in natural language parsing (Collins, 1999) and information retrieval Zhai and Lafferty (2004).

A basic first step in many statistical language models is to decompose the probability of a sentence using the product rule:

$$p(w_1, \ldots, w_m) = p(w_1| < \text{start} >) \prod_{t=2}^{m} p(w_t|w_1, \ldots, w_{t-1}) \ .$$

Here <start> is a special symbol for the start of a sentence. Given a large corpus of text, one can estimate the maximum likelihood paramters for each of these conditional probabilities using relative frequencies:

$$p_{\text{RF}}(w_t|w_1, \ldots, w_{t-1}) = \frac{c(w_1, \ldots, w_t)}{c(w_1, \ldots, w_{t-1})} \ ,$$

where $c(\cdot)$ is the count of a joint event.

Once we write the joint probability in this form, we can observe the central problem of language modeling: data sparseness. As the number of preceding words t becomes large, the number of possible phrases grows exponentially. For a vocabulary of 100,000 words, there are 10^{25} possible five word phrases. With a limited amount of data, even grammatical phrases are unlikely to have been observed before. A common approximation to help alleviate this problem is the n-gram approximation. Instead of counting phrases of length t, we count phrases only up to some fixed number n. For example, a trigram ($n = 3$) language model gives the probability of a sentence as

$$p(w_1, \ldots, w_t) = p(w_1| < \text{start} >) \prod_{t=2}^{m} p_{\text{RF}}(w_t|w_{t-2}, w_{t-1}) \ .$$

While it is easier to collect statistics for lower-order n-gram models, longer contexts give more precise estimates of the probability of the next word. To deal with this, one often combines higher and lower-order n-grams with interpolation or backoff (Jelinek, 1997). The simplest interpolated trigram estimates the probability of the next word as

$$p_{\text{LI}}(w_t|w_{t-2}, w_{t-1}) = \lambda_1 p_{\text{RF}}(w_t|w_{t-2}, w_{t-1}) + \lambda_2 p_{\text{RF}}(w_t|w_{t-1}) + \lambda_3 p_{\text{RF}}(w_t) \ , \qquad \lambda_1 + \lambda_2 + \lambda_3 = 1 \ .$$

n-gram language model smoothing is a very well-studied area, and there is no way to possibly cover every aspect of it in this report, but Chen and Goodman (1996) give an excellent overview.

Even with the most elaborate backoff schemes, though, n-gram language models do not capture an important generalization aspect of natural language. That is that certain words and even whole phrases are reasonable substitutions for one another. For instance, if we observe the phrase "`The brilliant written preliminary examination`", we should also be able to assign high probability to the phrase "`A solid oral qualification test`", despite the fact that these two phrases share no common words. In the next section we shall see how the neural probabilistic language model accomplishes this.

5.2 Neural Network Architecture

The neural probabilistic language model models an n-gram conditional probability $p(w_t|w_{t-n+1}, \ldots, w_{t-1})$ in three steps:

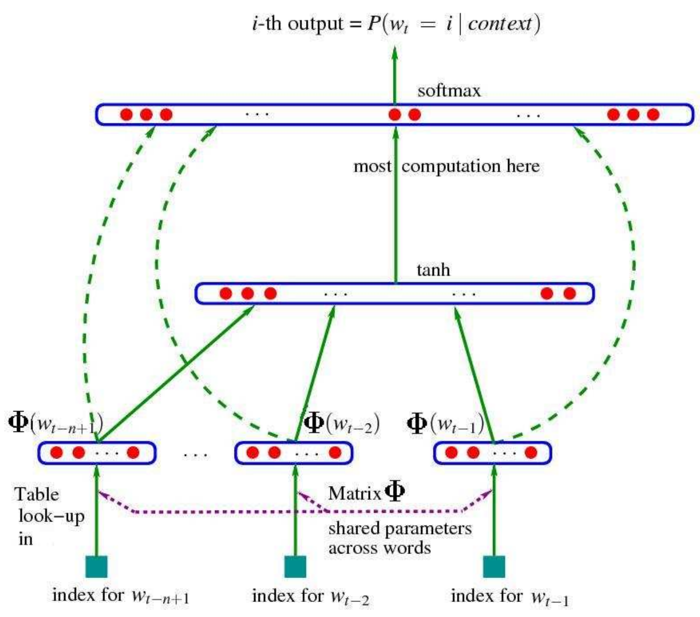

Figure 3: The NPLM architecture (Bengio et al., 2003)

1. **Input layer:** For each word w_{t-i} in the history, find an embedding $\phi(w_{t-i}) \in \mathbb{R}^k$.
2. **Hidden layer:** Compute a combined representation using a sigmoid nonlinearity.
3. **Output layer:** Choose a probability distribution over next words w conditioned on the hidden layer.

The NPLM learns the input representations $\phi(w)$, as well as the weights of the neural network simultaneously to maximize likelihood. This is depicted in figure 3.

Beginning with the input representation, we can write the vector-valued function ϕ as a concatenated vector

$$\phi(w_{t-1}, \ldots, w_{t-n+1}) = [\phi(w_{t-1}), \ldots, \phi(w_{t-n+1})] \ .$$

The hidden layer has the form

$$\mathbf{h}(w_{t-1}, \ldots, w_{t-n+1}) = \tanh(C\phi(w_{t-1}, \ldots, w_{t-n+1})) \ , \qquad C \in \mathbb{R}^{k(n-1)\times h}$$

where C is a weight matrix and tanh is element-by-element. h is the number of units in the hidden layer. Finally the output layer has the log-linear form

$$p(w_t | w_{t-n+1}, \ldots, w_{t-1}) = \frac{1}{Z(w_{t-1}, \ldots, w_{t-n+1})} \exp(\psi(w_t) \cdot \mathbf{h}) \ ,$$

where $Z(w_{t-1}, \ldots, w_{t-n+1})$ is a normalizing constant for the n-gram history. Bengio et al. (2003) use stochastic gradient ascent on the log-likelihood to find the (locally) maximizing parameters of the NPLM.

Model	n	clusters	h	k	direct	mix	PPL
MLP8	3	–	50	30	yes	yes	270
MLP10	5	–	100	30	no	yes	**252**
Kneser-Ney	5	–	–	–	–	–	321
class-based KN	3	500	–	–	–	–	**312**

Table 1: Selected results from Bengio et al. (2003)

5.3 Results

Bengio et al. (2003) ran a large battery of tests with the NPLM, and in this section we focus on an illustrative subset. Language model performance is often measured using perplexity, the geometric mean of the negative log-likelihood:

$$\mathrm{PPL} = \exp\left[\sum_{w_{t-n+1},\ldots,w_t} \tilde{p}(w_{t-n+1},\ldots,w_t)\log p(w_t|w_1,\ldots,w_{t-n+1})\right] .$$

The name "perplexity" indicates that the number is supposed to indicate how perplexed a model is. This number can also be viewed as the size of a list from which the model must pick, if it was picking uniformly at random. The best perplexity possible is a perplexity of 1, and the worst is V.

Table 1 gives a small subset of the results from Bengio et al. (2003) The training corpus is the Brown corpus, a 1 million-word collection of different genres of English. The column n is the n-gram length. The column "clusters" is the number of clusters for the cluster-based n-gram baseline. Finally the columns h and k refer to the number of hidden units and dimensionality of ϕ, respectively. The "direct" column refers to whether or not there are direct links between the input layer ϕ and output layer ψ. These links model a log-linear relationship between the n-gram history and the predicted word. Indeed, a bigram model with *only* linear links has exactly the same parameterization as the SDR model, although it models a conditional distribution rather than a joint distribution.

The best-performing model is MLP10, which has 100 hidden units, no direct links, and is mixed with a Kneser-Ney trigram. This model improves over the Kneser-Ney 5-gram by 21.5%. Bengio et al. (2003) note that interpolating the neural-network models with the standard trigram models consistently improves perplexity. Furthermore, direct connections improve the performance of the neural network model alone, but actually decrease performance of the interpolated model. They speculate that the highly nonlinear "MLP10" neural network model learns a much different function than the interpolated trigram, and thus they can improve each other's performance.

5.4 Discussion

The neural probabilistic language model is the first large-scale attempt to represent whole phrases compactly and simultaneously, and to use this representation for a standard natural language task. It is still not completely understood how this model represent multi-word semantics or how applicable it could be in other areas of natural language processing.

The NPLM as discussed in the previous section is very slow to train, mostly because for each observed n-gram in training, one must recompute the normalization factor $Z(w_{-1}, \ldots, w_{-(n-1)})$. Bengio and Sncal (2003) give a 100-fold speedup using importance sampling, and Morin and Bengio (2005) show that by using a hierarchical decoding mechanism one can achieve an exponential speedup over the model discussed in this report.

Finally, it is important to note that language modeling, unlike many areas of natural language processing, has a potentially infinite amount of training data. The 1 million word corpus employed by Bengio et al. (2003) is paltry compared to typical n-gram language models used in large-scale speech and machine translation systems, which are typically trained on billions of words. With such an immense amount of data available, it is difficult to make progress in traditional statistical language modeling using new and interesting models such as the NPLM (Goodman, 2001)

One place which might benefit even more from the ideas of the NPLM is discriminative language modeling (Roark et al., 2004). Discriminative language models work by reranking the m-best lists of a first pass speech recognition or machine translation system. Discriminative language modeling has two aspects that make it a particularly appealing use of the NPLM. First, one needs only to score the best word sequence and does not need to normalize a probability distribution over words, resulting in a potentially very large speedup. Secondly, unlike traditional language modeling, there *is* a limited amount of training data in the form of translation pairs or speech transcriptions.

6. Structural Learning

Structural learning (Ando and Zhang, 2005) is different from the four previous dimensionality reduction techniques in that it is specifically design for semisupervised learning. The dimensionality reduction training criterion is quite different from the classification loss which is used to evaluate the representation. Indeed, the dimension reduction itself is a straightforward application of the singular value decomposition. The fundamental insight of structural learning is in the design of "auxiliary problems" to guide the choice of subspace. We first introduce the motivation for structural learning with shared hypothesis spaces. In section 6.3.1 we describe the alternating structure optimization algorithm. Section 6.4 gives several examples of auxiliary problems. Finally we give results and a brief discussion in sections 6.5 and 6.6.

6.1 Motivation: Finding good hypothesis spaces

The main task in supervised classification is to find a predictor mapping an input (here we assume vector) $\mathbf{x}$ to an output label y. In most formulations of this problem we select this predictor from a hypothesis space $\mathcal{H}$. Predictor goodness is evaluated using a loss function which measures the discrepancy between the output of a labeling predictor $f(\mathbf{x})$ and the associated correct label y. For a distribution $\mathcal{D}$ on pairs $(\mathbf{x}, y)$, the optimal predictor in the hypothesis class $\mathcal{H}$ is

$$\hat{f} = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \, E_{\mathcal{D}} \left(L \left(f(\mathbf{x}), y \right) \right) \ .$$

For realistic problems, we do not have the true distribution available to us, but only a finite sample, which we denote $\mathcal{S}$. One method for choosing f given $\mathcal{H}$ and $\mathcal{S}$ is the method of regularized empirical risk minimization:

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} E_{\mathcal{S}} \left(L \left(f(\mathbf{x}), y \right) \right) + R(f) \ ,$$

where $R(f)$ is a regularization term. As an example, we might choose as our loss function a hinge loss where $\mathcal{H}$ is the set of linear classifiers and $R(f) = ||f||^2$ is the squared norm of the weight vector.

In semisupervised learning, in addition to our labeled sample $\mathcal{S}$, we also are endowed with a large amount of unlabeled data. The basic idea behind structural learning is to learn a hypothesis class $\mathcal{H}_{\mathbf{\Phi}}$ using the unlabeled data, where $\mathbf{\Phi}$ parameterizes the space of hypothesis classes. Then we choose our predictor from $\mathcal{H}_{\mathbf{\Phi}}$. If the hypothesis class we learn is good, then we expect to be able to choose a better function f from our labeled sample $\mathcal{S}$. The idea to use the unlabeled data to learn a hypothesis space bears a strong resemblance to the techniques of graph manifold-based semisupervised learning (Belkin and Niyogi, 2004; Belkin et al., 2005; Zhu et al., 2003, 2005). These techniques construct a data manifold based on a neighborhood graph of the unlabeled data and use this manifold to constrain supervised predictors. In practice structural learning most closely resembles the Laplace regularization work of Belkin et al. (2005). As we shall see, though, structural learning is formulated differently than any of the data manifold methods. We will return to this relationship again in section 6.6.

6.2 Shared structure via auxiliary problems

From now on we will refer to the classification problem for which we have labeled data as the *target* problem. The key idea in structural learning is the design of *auxiliary* problems which meet the following three criteria:

1. Auxiliary problems are closely related to the target problem. For instance, all the auxiliary problems we discuss here will use the same *feature* set as the target problem.

2. Auxiliary problems are as different as possible from one another.

3. Auxiliary problems do not require target labeled data to train.

For example, suppose our target problem is part of speech tagging, where each instance consists of features over word triples, and the task is to give the part of speech tag of the middle word. The label for "`the insightful paper`" is "`adjective`", the part of speech tag for "`insightful`". One set of appropriate auxiliary problems would be to predict the identity of the left word from features on the middle and right words. For each instance we can create one thousand left binary classification problems, word problems, one for each of the one thousand most frequent left words.

Since auxiliary problems require only unlabeled data to create, we can train highly reliable auxiliary predictors from the unlabeled data. Since we required the auxiliary problems to be diverse, we can say that the auxiliary predictors span the space of *predictor functions* (This will be made more precise in the next section). Intuitively, since we designed the auxiliary problems to be similar to the target problem, any common structure they have is likely to also be shared by a good target predictor. Thus if we can discover a good *predictor subspace* from our auxiliary predictors, this subspace can serve as our hypothesis space.

6.3 Learning a good hypothesis space

Suppose we create m auxiliary problems, where the ℓth auxiliary problem has n_ℓ instances. Let $\mathbf{x}_\ell^i \in \mathbb{R}^V$ be the ith instance for the ℓth auxiliary problem. Ando and Zhang (2005) suggest to choose the linear predictor subspace parameterized by the matrix $\mathbf{\Phi} \in \mathbb{R}^{k \times V}$ which minimizes the regularized empirical risk of all the auxiliary problems simultaneously. Each auxiliary predictor is characterized by two weight vectors: $\mathbf{w}_\ell$ on the original feature space and $\mathbf{v}_\ell$ on the feature space that has been transformed via the mapping $\mathbf{\Phi}$.

$$\begin{aligned}\left[\{\hat{\mathbf{w}}_\ell, \hat{\mathbf{v}}_\ell\}, \hat{\mathbf{\Phi}}\right] &= \underset{\mathbf{w}_\ell, \mathbf{v}_\ell, \mathbf{\Phi}}{\operatorname{argmin}} \sum_{\ell=1}^{m} \left(\frac{1}{n_\ell} \sum_{i=1}^{n_\ell} L\left((\mathbf{w}_\ell + \mathbf{\Phi}'\mathbf{v}_\ell)'\mathbf{x}_i^\ell, y_i^\ell \right) + \lambda ||\mathbf{w}_\ell||^2 \right) \\ & \text{s.t.} \quad \mathbf{\Phi}\mathbf{\Phi}' = I_{k \times k} \ .\end{aligned}$$

Ando and Zhang (2005) call this optimization criterion joint empirical risk minimization. Note that $\mathbf{w}_\ell$ is regularized, but $\mathbf{v}_\ell$ is not. This will play an important role in the derivation of the the alternating structural optimization algorithm for minimizing the joint empirical risk. After the derivation of the basic algorithm in section 6.3.1, we discuss the the actual implementation that Ando and Zhang (2005) use in their experiments in section 6.3.2.

6.3.1 Alternating structural optimization

In order to derive the alternating structural optimization (ASO) algorithm, we first introduce a change of variables. For each auxiliary problem we can write $\mathbf{u}_\ell = \mathbf{w}_\ell - \mathbf{\Phi}'\mathbf{v}_\ell$. We can rewrite the optimization problem as

$$\begin{aligned}\left[\{\hat{\mathbf{u}}_\ell, \hat{\mathbf{v}}_\ell\}, \hat{\mathbf{\Phi}}\right] &= \underset{\mathbf{u}_\ell, \mathbf{v}_\ell, \mathbf{\Phi}}{\operatorname{argmin}} \sum_{\ell=1}^{m} \left(\frac{1}{n_\ell} \sum_{i=1}^{n_\ell} L\left(\mathbf{u}_\ell'\mathbf{x}_i^\ell, y_i^\ell \right) + \lambda \left|\left|\mathbf{u}_\ell - \mathbf{\Phi}'\mathbf{v}_\ell\right|\right|^2 \right) \\ & \text{s.}t. \quad \mathbf{\Phi}\mathbf{\Phi}' = I_{k \times k},\end{aligned}$$

and at the optimal solution we can recover $\hat{\mathbf{u}}_\ell = \hat{\mathbf{w}}_\ell - \mathbf{\Phi}'\hat{\mathbf{v}}_\ell$. Now we come to the basic formulation of the ASO:

1. Fix $(\mathbf{\Phi}, \mathbf{v})$ and optimize with respect to $\mathbf{u}$.
2. Fix $\mathbf{u}$ and optimize with respect to $(\mathbf{\Phi}, \mathbf{v})$.
3. Iterate until convergence.

Note that in step 1, the optimizations for each auxiliary problem decouple, and we can solve each one separately. These are just standard empirical risk minimization problems, and if the loss function L is convex, then we can solve them with any minimization technique. Ando and Zhang (2005) suggest stochastic gradient descent. We focus now on step 2, which for fixed $\mathbf{u}_\ell = \hat{\mathbf{u}}_\ell$ yields the optimization problem

$$\left[\{\hat{\mathbf{v}}_\ell\}, \hat{\mathbf{\Phi}}\right] = \underset{\{\mathbf{v}_\ell\}, \mathbf{\Phi}}{\operatorname{argmin}} \sum_{\ell=1}^{m} \lambda \left|\left|\hat{\mathbf{u}}_\ell - \mathbf{\Phi}'\mathbf{v}_\ell\right|\right|^2 \qquad \text{s.}t. \quad \mathbf{\Phi}\mathbf{\Phi}' = I_{k \times k}.$$

```
Input:    labeled data $\{(\mathbf{x}_t, y_t)_{t=1}^T\}$,
          unlabeled data $\{\mathbf{x}_j\}$
Output:   predictor $f : X \rightarrow Y$
1.  Choose $m$ binary auxiliary problems, $p_\ell(\mathbf{x})$,  $\ell = 1 \dots m$
2.  For $\ell = 1$ to $m$
        $\hat{\mathbf{w}}_\ell = \text{argmin}_{\mathbf{w}} \left( \sum_j L(\mathbf{w} \cdot \mathbf{x}_j, p_\ell(\mathbf{x}_j)) + \lambda ||\mathbf{w}||^2 \right)$
    end
3.  $\mathbf{W} = [\hat{\mathbf{w}}_1 | \dots | \hat{\mathbf{w}}_m]$,    If $\mathbf{W}_{i,\ell} < 0$, set $\mathbf{W}_{i,\ell} = 0$.
4.  $[U\ D\ V'] = \text{SVD}(\mathbf{W})$,    $\mathbf{\Phi} = U'_{[1:k,:]}$
5.  Return $f$, a predictor trained on $\left\{ \left( \begin{bmatrix} \mathbf{x}_t \\ \mathbf{\Phi}\mathbf{x}_i \end{bmatrix}, y_t \right)_{t=1}^{T} \right\}$
```

Figure 4: ASO algorithm as it is implemented in practice

For fixed $\mathbf{\Phi}$, we have a least squares problem for $\mathbf{v}$

$$\min_{\mathbf{v}_\ell} \left|\left| \hat{\mathbf{u}}_\ell - \mathbf{\Phi}' \mathbf{v}_\ell \right|\right|^2 \ .$$

Differentiating with respect to $\mathbf{v}$ and setting to 0 reveals

$$0 = 2\mathbf{\Phi} \left(\hat{\mathbf{u}}_\ell - \mathbf{\Phi}' \mathbf{v}_\ell \right) \ .$$

Solving for $\mathbf{v}_\ell$ we arrive at the solution $\hat{\mathbf{v}}_\ell = \mathbf{\Phi}\hat{\mathbf{u}}$. Finally, we can substitute this back into the original minimization problem, yielding

$$\hat{\mathbf{\Phi}} = \underset{\mathbf{\Phi}}{\text{argmin}} \sum_{\ell=1}^{m} \lambda \left|\left| \hat{\mathbf{u}}_\ell - \mathbf{\Phi}'\mathbf{\Phi}\hat{\mathbf{u}}_\ell \right|\right|^2 \qquad \text{s.}t. \ \ \mathbf{\Phi}\mathbf{\Phi}' = I_{k \times k}.$$

Let $\mathbf{W} = [\mathbf{u}_1, \dots, \mathbf{u}_m]$ be the matrix whose columns are the weight vectors $\mathbf{u}$. Now by following a procedure similar to section 3.1.1, we can see that the solutions are at

$$\mathbf{\Phi}\mathbf{\Lambda} = \mathbf{W}\mathbf{W}'\mathbf{\Phi} \ ,$$

with $\mathbf{\Lambda}$ diagonal. Together with the orthogonality constraint, we know that the columns of $\mathbf{\Phi}$ are eigenvectors of the covariance matrix $\mathbf{W}\mathbf{W}'$. Finally, by a similar argument to that of section 3.1.1, we can derive that the minimizing solution occurs at the top eigenvectors. Thus we can also solve the optimization problem above with a singular value decomposition.

6.3.2 The ASO algorithm in practice

On could run the alternating structural optimization as described in the previous section to find $\mathbf{\Phi}$, but in order to achieve the results that Ando and Zhang (2005) report, we must make several changes to the form of the algorithm. The final, simpler algorithm is shown

An example weight matrix $\mathbf{W}$, where columns are auxiliary predictor weight vectors and rows are features. Rows are organized in blocks by feature type. The grayed block is th submatrix for feature type $\mathcal{T}_k$.

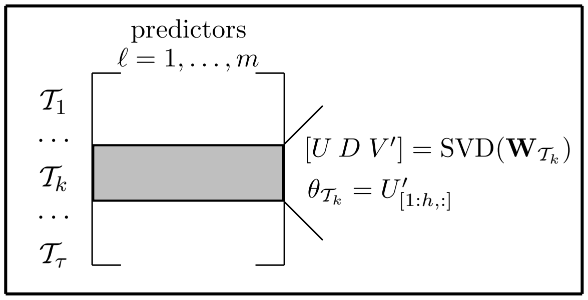

Figure 5: An illustration of block SVD by type, from Ando and Zhang (2005).

in figure 4. The first change from the ASO algorithm as described in section 6.3.1 is that there is no alternation. That is, we only need to run one iteration of (each step of) the optimization. In practice there are far fewer parameters from the weight vectors $\mathbf{v}_\ell$ on the transformed feature space than from the weight vectors $\mathbf{w}_\ell$ on the original space. Thus the $\mathbf{u}_\ell$ are unlikely to change significantly in the later iterations. Since the $\mathbf{u}_\ell$ do not change significantly, $\mathbf{\Phi}$ will not change significantly, either.

Running only one iteration allows us to simplify training the auxiliary predictors $\mathbf{u}$. Since we are only running one iteration, and since we initialize $\mathbf{\Phi} = \mathbf{0}^{k \times V}, \quad \mathbf{w}_\ell = \mathbf{0}, \quad \mathbf{v}_\ell = \mathbf{0}\ \forall \ell$, we know that $\mathbf{u}_\ell = \mathbf{w}_\ell$. Thus we can simply set the weight vectors by minimizing the empirical risk with a quadratic regularization.

The second important change to ASO is that when constructing the matrix $\mathbf{W}$ whose columns are the weight vectors $\mathbf{w}_\ell$, we set all the negative entries $\mathbf{W}_{i,\ell} = 0$ and compute the SVD of the resulting sparse matrix. This serves two purposes. First, it saves space and time. For a feature space of size 1 million and 3,000 auxiliary problems, $\mathbf{W}$ has 3 billion entries. Since, as we will see in the next section, most auxiliary problems are of the form "predict whether an adjacent word is `<w>`", they have many more negative instances than positive. Solving the sparse singular value decomposition that results from setting these entries to zero provides a significant speedup. Secondly, for many auxiliary problems we really care about positive instances, but not negative instances. For example, when predicting whether a word occurs, a positive instance gives us much more information. Thus we can consider discarding the negative entries as discarding the "noisy" entries of this matrix.

The last extension that makes an important difference is to split features based on what Ando and Zhang (2005) call "feature type". Suppose that for a tagging problem, we have three types of features: left words, middle words, and right words. Ando and Zhang (2005) point out that these feature types are not homogenous and should not necessarily be represented with the same projection $\mathbf{\Phi}$. They suggest performing an SVD just on the submatrix corresponding to a specific feature type (shown in figure 5). Then, during supervised training and testing, the matrices $\mathbf{\Phi}_\mathcal{T}$ are applied to the appropriate types separately, and the features are concatenated into a single feature vector.

6.4 Examples of auxiliary problems

The choice of auxiliary problems is essential to the performance of structural learning. Auxiliary problems that closely mimic the target problem can lead to a hypothesis space that gives significant gains over the original feature space, but auxiliary problems that are noisy or orthogonal to the target problem give no improvement. Ando and Zhang (2005) divide auxiliary problems into two basic types: unsupervised and partially supervised. Unsupervised auxiliary problems involve predicting the identity of one binary feature in an instance given the values of the other features in the instance. Partially supervised auxiliary problems are inspired by co-training (Blum and Mitchell, 1998). To create a partially supervised auxiliary problem, we first train a classifier on the labeled data. Then we *label* the unlabeled data with this classifier and predict the output of this classifier using subsets of the features. In this section we give examples of auxiliary problems for document classification and named entity recognition, and we briefly discuss other applications of structural learning.

Document classification. Document classification is the task of labeling a document with its category. In the 20 newsgroups data set, for instance, each document is a newsgroup posting and the task is to predict the identity of the newsgroup from the text of the document. Ando and Zhang (2005) use as features each word weighted by its term frequency, where the feature values sum to one for each document. Auxiliary problems are created as follows: Discard all stop words from a standard stop word list. Then randomly divide the words into two groups. For each document, choose the most frequent word in that document from each list. Then rank these words by frequency and choose the top 1000 from each group. For each document, we create auxiliary problem instances of the form "`Is the word <w> the most frequent from group 1 in this document?`" and similarly for group 2. Now when predicting words from group 1, we *only* use words from group 2 as features. There are a total of 2000 unsupervised binary auxiliary problems. The partially supervised task is to predict the top k categories output by a supervised classifier. For a classification task with C labels, there are a total of $\left(\begin{array}{c} C \\ k \end{array} \right)$ binary auxiliary problems.

Named entity recognition. Named entity recognition is a labeling and segmentation problem with many heterogeneous features. Given a document, the task of a named entity recognizer is to tag each word as `beginning a name`-B, `inside a name`-I, or `outside a name`-O. In addition, we may be tagging different types of names simultaneously, such as people, locations, and organizations, giving us B-Per, I-Per (and similar labels for the other types of named entities). Instances for named entity recognition usually represent the labels for individual words or adjacent pairs of words (Florian et al., 2003). These individual instances usually consist of windows centered around a current word or pair and will serve as instances for auxiliary problem training on the unlabeled data, as well.

For unsupervised auxiliary problems, Ando and Zhang (2005) predict the occurence of the 1000 most frequent words in the left, middle and right positions. This creates 3000 auxiliary problems of the form "`Does the word <w> occur as the left (middle/right) word of this instance?`". They mask all features derived from a word when predicting that word. For instance, when predicting the left word, they never use features derived from the left word. For partially supervised auxiliary problems, Ando and Zhang (2005) predict the top 2 labels of a supervised classifier.

Ando and Zhang (2005) break the features for named entity recognition into the following types: current words, left words, right words, bag-of-words in the current, left, and right syntactic chunks, current, left, and right part of speech tags, and prefix and suffix features. Each of these types is treated separately for dimension reduction, and they learn a separate $\mathbf{\Phi}$ projection for each feature type.

Other tasks. Ando and Zhang (2005) also apply structural learning to handwritten digit recognition and show that by designing features and auxiliary problems based on pixel configurations, they can achieve a low error rate in other tasks besides language. Ando (2006) recently applied structural learning to the task of word sense disambiguation. She did not use unsupervised auxiliary problems, but she showed improvement from using partially supervised auxiliary problems as well as multitask training only on the labeled data.

6.5 Results

Ando and Zhang (2005) report a plethora of results on many different problems. In this section we briefly summarize them. For document classification, they compare with both co-training and the manifold semisupervised method of Belkin and Niyogi (2004). The free parameter for co-training is the number of iterations, and the free parameter for Belkin and Niyogi (2004) is the number of eigenvectors of the graph Laplacian to use. On the standard 20 newsgroups and Reuters RCV-1 corpora, they show that structural learning consistently outperforms both of these methods at the *best* setting of their free parameters. For handwritten digit recognition, they again compare with Belkin and Niyogi (2004). With the smallest amount of labeled data, the manifold semisupervised learning outperforms structural learning, but structural learning outperforms the manifold learning method at all other numbers of data.

The most impressive result, however, is performance for the 2003 conference on natural language learning shared task on named entity recognition. Winners of these competitions usually employ a great deal of feature engineering in building the best systems. Ando and Zhang (2005) reports results using a large amount of training data which outperform the best systems in this competition for both English and German named entity recognition.

6.6 Perspectives on structural learning

Structural learning can be viewed from a variety of perspectives. Ando and Zhang (2005) focus on multitask learning of a shared hypothesis space. In this section we address a few other, not necessarily contradictory, views of structural learning.

Structural learning as regularization. In section 6.1, we noted that structural learning is similar in spirit to the Laplace regularization method of Belkin et al. (2005). In that work linear predictors are regularized based on their smoothness along the data manifold. This manifold is approximated using a neighborhood graph on the unlabeled data. Similar to this work, structural learning can be seen as preferring functions which are smooth in the reduced "predictor space" derived from the auxiliary predictors. Ando and Zhang (2005)

Model	Parameterization	Loss	Optimization	Applications
LSA	Gaussian	squared L_2	eigenvalue	information retrieval
PLSA	multinomial mean	KL	EM	information retrieval
SDR	multinomial natural	KL	iterative projection	information retrieval
NPLM	log-linear neural network	KL	stochastic gradient descent	language modeling
ASO	classifier weight space	joint ERM	stochastic gradient descent + SVD	classification

Table 2: Tabular summary of the five dimensionality reduction methods.

point out that measuring smoothness along the data manifold may not be an appropriate way to constrain functions for classification, and they suggest that careful design of auxiliary problems provides a better tailored way to do this.

Two-step dimensionality reduction. It is worthwhile to note that by choosing significantly fewer auxiliary problems than the size of the original features $O(10^3)$ vs $O(10^6)$, Ando and Zhang (2005) are already significantly reducing the dimensionality of the hypothesis space. In this respect, structural learning is reducing dimensionality in two steps. The first step may make the second cleaner or more reliable.

Unsupervised Auxiliary problems model feature covariance. Alternating structural optimization is motivated from the perspective of joint empirical risk minimization. As such, the algorithm trains thousands of auxiliary predictors on a large amount of unlabeled data. But it may be that these predictors are superfluous and that just the co-occurrence counts alone are sufficient to find a good mapping $\mathbf{\Phi}$. Schutze (1998) also used the SVD directly on co-occurrence data successfully to model word sense disambiguation. Ando (2004) performed experiments with named entity recognition that used normalized co-occurrence counts in place of the positive-valued weight vectors from ASO. It would be interesting to see a comparison of ASO and an algorithm which uses the (normalized) co-occurrence counts directly.

Theoretical analysis. Ando and Zhang (2005) give a PAC-style theoretical analysis of structural learning showing that in the limit as the number of auxiliary problems $m \to \infty$, the learned hypothesis space will converge uniformly to the optimal hypothesis space . While this is an interesting analysis in its own right, we omit it here since it sheds limited light on the question of semisupervised learning. That is, the theory has nothing to say about the error of the *target* problem, which is what we are really interested in.

7. Discussion and Future Work

In this section we highlight some similarities and differences among the models in this report. We also suggest some concrete directions for future work in section 7.1 and more speculative ideas in section 7.2.

Table 2 summarizes some of the aspects of each of the models in this report. The older methods of PLSA and LSA have somewhat unflexible parameterizations. In the case of LSA, the Gaussian parameterization can lead to decreases in information retrieval performance (Collins et al., 2002; Hofmann, 1999; Globerson and Tishby, 2003). For PLSA, parameterizing a joint multinomial by factoring the mean parameters, rather than the natural parameters, forces a "folding in" optimization to be run at every query.

Suffcient dimensionality reduction parameterizes a joint multinomial using a factorization of the natural parameters. Bengio et al. (2003) takes this idea a step further by learning a nonlinear mapping of separate real-valued embeddings for each discrete element. Finally, Ando and Zhang (2005) shows that by reducing the dimension of a real vector "classifier space", rather than a discrete "feature space" can lead to increased performance for supervised classifiers.

Despite the advantages the more recent models have over LSA, it is important to recognize the virtues of the singular value decomposition for co-occurrence data. The top eigenvectors of the covariance matrix are the globally minimal solution for the squared loss, and there exist prepackaged, fast eigenvalue solvers for large sparse matrices. Although Globerson and Tishby (2003) and Bengio et al. (2003) both show that the SVD does not perform as well as a properly motivated multinomial model, we note that their datasets are *much* smaller than the data sets that could easily be handled by standard SVD solvers. Ando and Zhang (2005) do test their method on large data, but they don't comparre with their own previous method (Ando, 2004), which performed SVD directly on the co-occurrence data. It is certainly worthwhile to consider performing SVD on the count data as a first, approximate technique.

7.1 Future Work on Dimensionality Reduction for Language

Dimensionality reduction of language data has traditionally been applied in the fields of language modeling and information retrieval. While it has achieved some limited success on small data sets, large-scale success has been more difficult. This is primarily because the large amount of available data for those tasks makes sophisticated statistical modeling both less effective and more time-consuming. Ando and Zhang (2005) showed how dimensionality reduction for semisupervised learning could be used to improve state-of-the-art discriminative models on a wide variety of natural language processing tasks. The algorithm they suggest, alternating structural optimization, is not completely understood, though. Thus it seems that many immediate successes in dimension reduction for language could come in the area of semisupervised learning. In this section we briefly outline three new ideas.

New applications for structural learning. Applying structural learning to document classfication and sequence labeling involves reusing some of the same ideas from dimensionality reduction for language modeling and information retrieval. It would be interesting to see how structural learning applies to problems like parsing, where instances involve attachment decisions, or machine translation, where the notion of a single "instance" is less clear.

Better co-occurrence representations for structural learning. In section 6.6, we speculated that structural learning is modeling co-occurrence data and that at least the unsupervised auxiliary predictors were superfluous. If this is true, it is certainly worth investigating more appropriate methods for modeling the feature-feature co-occurrences. One easy and directly method to try is to apply sufficient dimensionality reduction directly to the feature-feature co-occurrence data.

Supervised dimensionality reduction for structural learning. Several authors have recently applied dimension reduction techniques to learning linear projections directly from labeled data. Goldberger et al. (2005) and Weinberger et al. (2006) learn a linear mapping to minimize a k-nearest-neighbor-inspired loss functions. As we mentioned in section 6.6, structural learning can be thought of as a two-step dimension reduction process. For the alternating structural optimization algorithm, the second step is a singular value decomposition. But we could learn the $\mathbf{\Phi}$ to directly minimize the target problems loss on the labeled training data. This seems to be a fruitful direction for future research.

7.2 Dimension Reduction and Natural Language Semantics

A holy grail of natural language understanding is to accurately model natural language semantics. One question that has been asked since the first works on dimensionality reduction of natural language is whether or not dimensionality reduction captures linguistic meaning. Indeed, the name "latent semantic analysis" was coined exactly to allude to this connection. All of the models in this report do seem to capture some notion of *lexical* semantics. By using dimensionality reduction for word sense disambiguation, Schutze (1998) and Ando (2006) both indicate that dimensionality reduction does capture some notion of natural language semantics.

Perhaps the most interesting long-term future goal is to use low-dimensional representations to model *compositional* semantics (Heim and Kratzer, 1998). Compositional semantics describes the way words and phrases combine to form meaning for larger phrases. The neural probabilistic language model does seem to at least partially model some of this behavior. Table 1 shows that a nonlinear hidden layer in the neural network can improve language model performance. But it is difficult to pinpoint what exactly the hidden layer captures.

Other than lower perplexity scores, which are not very enlightening, how might we evaluate real-valued semantic representations? It seems likely that combining representations hierarchically, perhaps in conjunction with syntactic parse trees, may give some answer to this question (Zettlemoyer and Collins, 2005). But there is some speculation that true natural language semantics cannot be learned without real-world grounding (Roy, 2005; Bengio, 2006). While truly representing natural language semantics does seem to be a distant goal, it is the heart of dimensionality reduction for language. We should not forget it when designing new models and methods.

8. Conclusion

Data sparseness is a central problem in natural language processing. In information retrieval, we need to match a query string with relevant documents. Both queries and documents are high-dimensional discrete objects, and measuring distances between them can be unreliable. In language modeling, the task is to estimate the probability of a sentence. Even short sequences (n-grams) of words can be very high-dimensional, and reliably estimating the joint n-gram probability without any low-dimensional structure can be very difficult. In discriminative learning for parsing and tagging problems, state-of-the-art models typically use millions of features, but often have only thousands of labeled training instances.

In this survey we reviewed methods that alleviate data sparseness by reducing the dimensionality of linguistic entities. We first reviewed the document-word co-occurence models of latent semantic analysis and probabilistic latent semantic analysis. LSA computes a low-rank factorization of the co-occurrence matrix by minimizing the squared loss of the counts. This leads to a fast and easy-to-solve eigenvalue problem, but the squared loss is equivalent to assuming a Gaussian generating distribution for the co-occurrence matrix. PLSA attempts to define a more appropriate statistical model by assuming the co-occurrence data is generated by a multinomial model and learning a factorization of the mean parameters of the multinomial to minimize the Kullback-Leibler divergence to the empirical distribution. Hofmann (1999) reports improvements in retrieval performance using PLSA instead of LSA, but PLSA also has drawbacks. Unlike LSA, PLSA requires an optimization to "fold in" new query vectors as it processes them.

The main focus of this survey is on three new models for dimensionality reduction of linguistic co-occurrence data. Sufficient dimensionality reduction, like PLSA, models empirical co-occurrence data as multinomial. Unlike PLSA, though, SDR computes a low rank factorization of the *natural* parameters of the multinomial. This exponential family formulation yields a flexible real-valued embedding of words and documents, allowing Globerson and Tishby (2003) to avoid "folding in" new query objects and yielding an unconstrained optimization problem.

The neural probabilistic language model uses real-valued embeddings to model multiple co-occurrences simultaneously. Bengio et al. (2003) compute the probability of an n-gram by first embedding each of the history words as a real vector. These real vectors are fed as input to a neural network whose ouput is normalized using a softmax to create a probability distribution over next words. The resulting embeddings and network parameters are trained together to minimize the Kullback-Leibler divergence to the empirical distribution. Bengio et al. (2003) report large gains over traditional n-gram smoothing techniques by using the NPLM on a small corpus.

Structural learning (Ando and Zhang, 2005) is a dimensionality reduction technique for semisupervised learning. Given a high-dimensional feature space, as well as labeled and unlabeled data, structural learning uses the unlabeled data to learn a low-dimensional projection of the feature space which generalizes well for the target classification task. The alternating structural optimization algorithm finds a projection of the feature space which minimizes the joint empirical risk of many auxiliary problems, trained on the unlabeled. The reduced-dimension feature space obtained by applying this projection is then used to train a target predictor using the labeled data. Ando and Zhang (2005) report results on several standard natural language processing tasks demonstrating impressive improvements over state-of-the-art discriminative classifiers.

The final part of the survey suggests areas for future work in dimensionality reduction of linguistic co-occurrence data, including combining some of the models in this survey, as well as applying new work in supervised dimensionality reduction (Goldberger et al., 2005; Weinberger et al., 2006) to language. We believe dimensionality reduction techniques for natural language is one of the most important areas for future research, and many of the ideas we suggested in section 7.1 are ideas we hope to actively pursue in the near future.

References

R. Ando. Exploiting unannotated corpora for tagging and chunking. In *ACL. Short paper*, 2004.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.

Rie Kubota Ando. Applying alternating structural optimization to word sense disambiguation. In *Conference on Natural Language Learning CoNLL*, 2006.

Rie Kubota Ando and Lillian Lee. Iterative residual rescaling: An analysis and generalization of LSI. In *Proceedings of the 24th Annual ACM Conference on Research and Development in Information Retrieval*, 2001.

Frank Baker. Information retireval based on latent class analysis. *Journal of the ACM*, 9 (4):512–521, 1962.

T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White. Testing that distributions are close. In *FOCS*, volume 41, pages 259–269, 2000.

Mikhail Belkin and Partha Niyogi. Semisupervised learning on riemannian manifolds. *Machine Learning Special Issue on Clustering*, 56:209–239, 2004.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. On manifold regularization. In *10th International Workshop on Artificial Intelligence and Statistics*, 2005.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003. ISSN 1533-7928.

Yoshua Bengio. Personal communication, 2006.

Yoshua Bengio and Jean-Sbastien Sncal. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2003.

David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

John Blitzer, Amir Globerson, and Fernando Pereira. Distributed latent variable models of lexical co-occurrences. In *10th International Workshop on Artificial Intelligence and Statistics*, 2005a.

John Blitzer, Kilian Weinberger, Lawrence Saul, and Fernando Pereira. Hierarchical distributed representations for statistical language modeling. In *Advances in Neural Information Processing Systems 18*, 2005b.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Workshop on Computational Learning Theory*, 1998. URL `citeseer.ist.psu.edu/blum98combining.html`.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossina. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85, 1990.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *CL Journal*, 18(4):467–479, 1992. URL `citeseer.ist.psu.edu/brown90classbased.html`.

S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL 1996*, pages 310–318, 1996.

M. Collins. *Head-driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

Michael Collins, Sanjoy Dasgupta, and Robert Schapire. A generalization of principal component analysis to the exponential family. In *Advances in Neural Information Processing Systems 14*, 2002.

Thomas Cover and Joy Thomas. *Elements of Information Theory.* John Wiley and Sons, 1991.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990. URL `citeseer.ist.psu.edu/deerwester90indexing.html`.

Ahmed Emami and Fred Jelinek. Random clusterings for language modeling. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2005.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Conference on Natural Language Learning CoNLL*, 2003.

Amir Globerson. Personal communication, 2006.

Amir Globerson and Naftali Tishby. Sufficient dimensionality reduction. *Journal of Machine Learning Research*, 3:1307–1321, 2003.

Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighborhood components analysis. In *Advances in Neural Information Processing Systems 18*, 2005.

J. Goodman. A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research, 2001.

Irene Heim and Angelika Kratzer. *Semantics in Generative Grammar.* Blackwell, Oxford, 1998.

Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, August 1999.

Thomas Hofmann and Jan Puzicha. Statistical models for co-occurrence data. Technical Report 1625, MIT Artificial Intelligence Laboratory, 1998.

Edwin James. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

F. Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, 1997.

Ryan McDonald. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, 2005.

F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

Ulrich Muller-Funk, Friedrich Pukelsheim, and Hermann Witting. On the attainment of the cramer-rao bound in $\mathbb{L}_r$-differentiable families of distributions. *The Annals of Statistics*, 17:1742–1748, 1989.

Peter Ossorio. Classification space: A multivariate procedure for automatic document indexing and retrieval. *Multivariate Behavioral Research*, pages 479–524, October 1966.

C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the Symposium on Principles of Database Systems*, 1998.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey, 1996. URL `citeseer.ist.psu.edu/ratnaparkhi96maximum.html`.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of ACL 2004*, 2004.

Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

Deb Roy. Grounding words in perception and action: Insights from computational models. *Trends in Cognitive Science*, 9:389–396, 2005.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. Optimization with em and expectation-conjugate-gradient. In *International Conference on Machine Learning*, 2003.

Gerald Salton. *Automatic Text Processing: the transfomation, analysis, and retrieval of information by computer.* Addison Wesley Publishion Company, Inc., 1989.

Lawrence Saul and Fernando Pereira. Aggregate and mixed-order markov models for statistical language modeling. In *Empirical Methods in Natural Language Processing*, 1997.

Hinrich Schütze. Word space. In *Proceedings of NIPS 7*, volume 7, 1993.

Hinrich Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1): 97–123, 1998. URL `citeseer.ist.psu.edu/schutze98automatic.html`.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL 2003*, Edmonton, Canada, 2003.

Nathan Srebro. *Learning with Matrix Factorizations.* PhD thesis, Massachusetts Institute of Technology, Boston, MA, 2004.

S. Wang, S. Wang, R. Greiner, D. Schuurmans, and L. Cheng. Exploiting syntactic, semantic, and lexical regularities in language modeling via directed markov random fields. In *Internation Conference on Machine Learning*, 2005.

Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 19*, 2006.

Luke Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of Uncertainty in Artificial Intelligence*, 2005.

C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Twentieth International Conference on Machine Learning*, 2003.

Xiaojin Zhu, Jaz Kandola, Zoubin Ghahramani, and John Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 18*, 2005.