DOMAIN ADAPTATION OF NATURAL LANGUAGE PROCESSING SYSTEMS

John Blitzer

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2007

Acknowledgements

My first thanks must go to Fernando Pereira. He was a wonderful advisor, and every aspect of this thesis has benefitted from his insight. At times I was a difficult, even unruly graduate student, and Fernando had patience with all my ideas, whether good or bad. What I'll miss most, though, is the quick trip to Fernando's office, coming away with new insights on everything from numerical underflow to the state of the academic community in machine learning and NLP.

In addition to Fernando, this thesis was shaped by a great committee. Having Ben Taskar as committee chairman has given me the perfect excuse to interrupt his workday with new, ostensibly-thesis-related machine learning ideas. Mark Liberman and Mitch Marcus brought a much-needed linguistic perspective to a thesis on language, and many of the techniques described are based on work by Tong Zhang, who kindly served as my external committee member. Although he didn't directly serve on my committee, Shai Ben-David got me started on the theoretical aspects of this work, and chapter 4 grew out of work I co-authored with him.

I was also fortunate to have a great academic family. With brothers (and one sister!) like these, weekly "Pereira group" meetings were something I always looked forward to. Ryan McDonald and I academically "grew up" together, and since his graduation, I have often missed his Canadian cool presence. Herr Professor Doktor Yacov Shlomo Crammer, Ph.D. has been a mentor and friend for the past four years. Axel Bernal, as a fellow fan of the Maestro of zerg, I hope this is not the closing "GG" on our time together. Qian Liu 是我唯一的师妹, and not being able to shout at her across the cubicle divider has left a definite void in my life. Mark Dredze taught me how conducive kosher dairy meals are to coming up with good research ideas. I already miss the wry “Gancho Ganchev” humor of Kuzman Ganchev. João Graça is without a doubt my favorite ninja: May your sense of balance always be with you, my friend. Alex Kulesza deserves my sincerest thanks for his many counterexamples and Peeps®. He kept learning theory fun and sugary. And Partha Pratim Talukdar, may there never be a Lakshmana Rekha between us.

I was fortunate enough to spend time with Lawrence Saul and his students Fei Sha and Kilian Weinberger. My original interest in dimensionality reduction for language is the result of conversations and collaborations with them. Lawrence was like a second advisor to me: always helpful and always willing to listen patiently to my latest hare-brained scheme. Fei was and still is an endless source of advice on nearly every topic. Kilian, our attempts at combining psychoanalysis with machine learning have been some of the most fun times in graduate school for me.

Many other friends also buoyed my spirits throughout the past six years. Coffee breaks won’t be the same without my fellow (very) amateur ancient Mediterranean historian Nikhil Dinesh. Yuan Ding’s deep knowledge of ice age fauna helped me in many ways to take life less seriously. Liang Huang 3: How will I continue my study of “zan he ë wo” without you? I am also indebted to Anne Bracy, Bill Kandylas, Peng Li, Rene Liu, Yun Mao, Andrew McGregor, Ted Sandler, Jenn Wortman, and Bei Xiao for their great attitudes and good humor.

Finally, I thank my mother and father, and my sisters Eleanor Jane, Amy, Mary, and Grace. Their love and support have made my graduate school career, and all of my work, worthwhile. I dedicate this thesis to them.

ABSTRACT

DOMAIN ADAPTATION OF NATURAL LANGUAGE PROCESSING SYSTEMS

John Blitzer

Fernando Pereira

Statistical language processing models are being applied to an ever wider and more varied range of linguistic domains. Collecting and curating training sets for each different domain is prohibitively expensive, and at the same time differences in vocabulary and writing style across domains can cause state-of-the-art supervised models to dramatically increase in error.

The first part of this thesis describes structural correspondence learning (SCL), a method for adapting linear discriminative models from resource-rich *source* domains to resource-poor *target* domains. The key idea is the use of *pivot* features which occur frequently and behave similarly in both the source and target domains. SCL builds a shared representation by searching for a low-dimensional feature subspace that allows us to accurately predict the presence or absence of pivot features on unlabeled data. We demonstrate SCL on two text processing problems: sentiment classification of product reviews and part of speech tagging. For both tasks, SCL significantly improves over state of the art supervised models using only unlabeled target data.

In the second part of the thesis, we develop a formal framework for analyzing domain adaptation tasks. We first describe a measure of divergence, the $\mathcal{H}\Delta\mathcal{H}$-divergence, that depends on the hypothesis class $\mathcal{H}$ from which we estimate our supervised model. We then use this measure to state an upper bound on the true target error of a model trained to minimize a convex combination of empirical source and target errors. The bound characterizes the tradeoff inherent in training on both the large quantity of biased source data and the small quantity of unbiased target data, and we can compute it from finite labeled and unlabeled samples of the source and target distributions under relatively weak assumptions. Finally, we confirm experimentally that the bound corresponds well to empirical target error for the task of sentiment classification.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Statistical language processing tools are being applied to an ever wider and more varied range of linguistic data. Researchers and engineers are using statistical models to organize and understand financial news, legal documents, biomedical abstracts, and weblog entries, among many other domains. Many of these models are supervised at parameter estimation time – A human annotator must create a training set of examples for the relevant task. Because language varies so widely, collecting and curating training sets for each different domain is prohibitively expensive. At the same time, differences in vocabulary and writing style across domains can cause state-of-the-art supervised models to dramatically increase in error. Domain adaptation methods provide a way to alleviate the problem of creating training sets for different domains by generalizing models from a resource-rich *source* domain to a different, resource-poor *target* domain.

This thesis investigates both the empirical and theoretical aspects of domain adaptation. The first half describes structural correspondence learning (SCL), a method for domain adaptation which uses unlabeled data to induce correspondences among features from different domains. SCL first learns a mapping from the high-dimensional feature spaces commonly used for text processing to a low-dimensional real-valued space. Correspondences are encoded implicitly in the structure of this low-dimensional space. We demonstrate SCL for two text processing tasks: part of speech tagging and sentiment classification. For each of these tasks, SCL significantly reduces the error of a state-of-the-art discriminative model.

The latter half of this thesis focuses on characterizing theoretically the conditions under which domain adaptation algorithms can be expected to perform well. We give a formal definition of domain adaptation, and we use this definition to provide an upper bound on the generalization error of a source model in a new target domain. Our theory illustrates how the right feature representation is crucial for domain adaptation, and we show that for both part of speech tagging and sentiment classification, the feature space learned by SCL results in smaller values for the bound on error. Finally, we extend our theory to the case in which we have small amounts of labeled target training data. In this setting, we give a bound that explicitly models the tradeoff that arises from training on both a large but biased source sample and a small but unbiased target sample.

1.1 Supervised models

This thesis is about domain adaptation of statistical models for mapping an input text to an output label. These models are called supervised because we train them by giving them examples of input and output pairs. For example, we may consider a model which takes as input a review of a particular product and produces as output a rating indicating how much the reviewer liked the product. In this section we introduce linear discriminative models, a common paradigm in supervised learning that has been especially successful for text processing. Structural correspondence learning is designed to adapt linear models to new domains, and the SCL algorithm itself involves training multiple linear predictors on unlabeled data. Here we describe background for understanding SCL, including feature representations, loss functions, optimization techniques, and generalization theory.

Many state-of-the art text processing systems use some form of linear discriminative modeling, and it has been a staple of speech and language processing for several

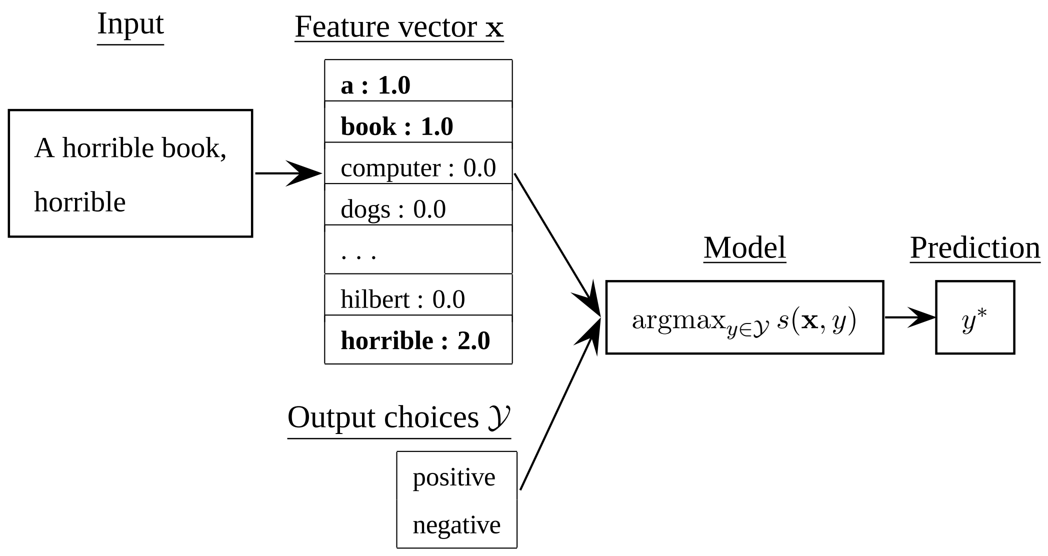

Input
A horrible book, horrible
Feature vector $\mathbf{x}$
a : 1.0
book : 1.0
computer : 0.0
dogs : 0.0
. . .
hilbert : 0.0
horrible : 2.0
Output choices $\mathcal{Y}$
positive
negative
Model
$\text{argmax}_{y\in\mathcal{Y}}\, s(\mathbf{x}, y)$
Prediction
y^*

Figure 1.1: Schematic of a supervised model for classifying product reviews. Given an input review of a book (far left), we first represent it as a feature vector. A supervised model scores combinations of feature vectors and outputs (either positive or negative) and returns the top-scoring label for this input (right).

decades. A complete discussion is well beyond the scope of this thesis, but we refer the reader to Hastie et al. [34] for an introduction to supervised learning and to Manning and Schütze [45] and Jelinek [38] for overviews of its use in natural language processing. Shawe-Taylor and Cristianini [58] is a good reference for support vector machines, a particularly popular method for training and representing linear models.

1.1.1 Feature representations for text

The first step in building a supervised model is to design features of the input which we believe will be helpful in predicting the output. Let us return to our previous example. What features of a product review are useful for predicting a potential rating? It turns out that for this problem, the presence or absence of particular words in the text are excellent features. For instance, the presence of the word “horrible” is good indication that the document expresses negative sentiment and should get a low rating.

Once we have decided on the features to use, we represent each input instance as a vector, where each dimension in the vector corresponds to a particular feature (figure 1.1). Such vectors are typically high-dimensional and sparse. If we use words as our feature representation, then the dimensionality of an input feature vector is the size of the vocabulary (in the tens or hundreds of thousands). For a particular document, however, only a few words will be present. In figure 1.1, for example, only the dimensions corresponding to “a”, “book” and “horrible” have non-zero value.

1.1.2 Linear discriminative models

A supervised model chooses a feature vector $\mathbf{x}$ and potential output y using a scoring function $s(\mathbf{x}, y)$. Finding the best output amounts to choosing the output y^* which has the highest score (right-hand side of figure 1.1). Linear models compute the score $s(\mathbf{x}, y)$ of an input and output pair using as a linear function of a weight vector $\mathbf{w}$. The simplest such models that we will address here are binary linear classification models. In this case, $\mathcal{Y} = \{-1, 1\}$ and

$$s_{\mathbf{w}}(\mathbf{x}, y) = y(\mathbf{w}'\mathbf{x}) \ ,$$

where $\mathbf{w}$ is a weight vector and $\mathbf{w}'\mathbf{x}$ is the inner product of $\mathbf{w}$ with $\mathbf{x}$. Since $y \in \{-1, 1\}$, the top-scoring label y is the same as the sign of the inner product

$$y^* = \underset{y}{\operatorname{argmax}}\, y(\mathbf{w}'\mathbf{x}) = \operatorname{sgn}(\mathbf{w}'\mathbf{x}) \ .$$

Aside from providing an important foundation on which more sophisticated linear models are built, binary linear predictors play an especially important role in this thesis. As we shall see in chapter 2, the SCL algorithm itself involves training hundreds or thousands of binary predictors.

Multiclass and structured linear modeling

For many learning problems, the set of possible labels $\mathcal{Y}$ is much larger than two. For example, we apply our domain adaptation techniques to the task of part of speech tagging (chapter 3), where the goal is to assign a sequence of part of speech tags to the words in a sentence. In this case, the number of possible sequences grows exponentially with the length of the sentence. For problems with more than two labels, we use a fixed mapping $\boldsymbol{\zeta}(\mathbf{x}, y)$ to create a sparse vector representation of input-output pairs[1]. As before, we compute the score using an inner product

$$s_{\mathbf{w}}(\mathbf{x}, y) = \mathbf{w}'\boldsymbol{\zeta}(\mathbf{x}, y) \; .$$

The model then outputs the top-scoring label

$$y^* = \operatorname*{argmax}_y \mathbf{w}'\boldsymbol{\zeta}(\mathbf{x}, y) \; .$$

For a fixed small set of labels, as in standard multiclass classification, we can create $\boldsymbol{\zeta}(\mathbf{x}, y)$ by concatenating the appropriate label with each feature. This gives us a new $\boldsymbol{\zeta}$ vector of dimension $|\mathcal{Y}|d$, where d is the dimensionality of the original feature vector $\mathbf{x}$, defined as follows:

$$\boldsymbol{\zeta}(\mathbf{x}, y)_i = \begin{cases} 1, & i = dy + j, \;\; \text{for some } j < d \;\; \text{and} \;\; \mathbf{x}_j = 1 \\ 0, & \text{otherwise} \end{cases} \; .$$

For learning problems such as part of speech tagging, natural language parsing, and machine translation, the number of labels is large enough that treating labels as atomic units is both statistically and computationally infeasible. In these tasks the labels themselves have internal structure that we can take advantage of in designing the mapping $\boldsymbol{\zeta}(\mathbf{x}, y)$. Because of this, models which solve these tasks are often referred to as structured predictors [42, 57, 60]. Methods for structured prediction must factor problems so as to be able to perform computationally efficient inference and to be able to make accurate predictions. When we investigate adapting part of speech taggers, we show how to integrate

[1]When we need to distinguish this vector from the input feature representation of the previous section, we will refer to this vector as the $\boldsymbol{\zeta}$ vector.

SCL with a structured linear predictor (chapter 3), but inference and learning algorithms for structured prediction are not central to the ideas of this thesis. For a good introduction to structured prediction problems and algorithms, we refer to Taskar [60].

1.1.3 Parameter estimation techniques

In section 1.1.2 we showed how to choose the best output for a particular input, given that we already have a linear model in hand. In this section we briefly review techniques for finding a linear model (parameterized by the weight vector $\mathbf{w}$) given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. We will refer to these techniques as parameter esimtation or training procedures. Let the error of a linear model $\mathbf{w}$ for a particular instance $(\mathbf{x}, y)$ be the 0-1 indicator variable

$$\left[\operatorname*{argmax}_{y\in\mathcal{Y}} s_{\mathbf{w}}(\mathbf{x}, y) \neq y\right] = \begin{cases} 0, & \operatorname{argmax}_{y\in\mathcal{Y}} s_{\mathbf{w}}(\mathbf{x}, y) = y \\ 1, & \operatorname{argmax}_{y\in\mathcal{Y}} s_{\mathbf{w}}(\mathbf{x}, y) \neq y \end{cases} .$$

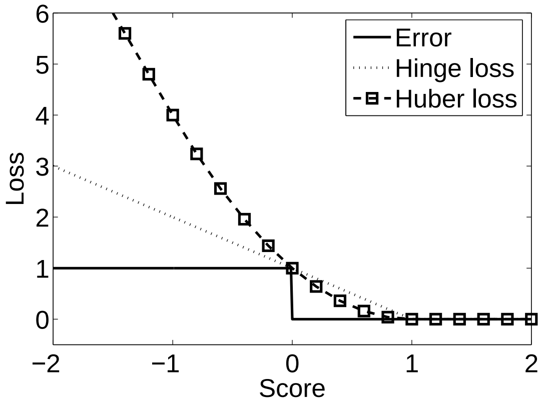
A natural criterion for $\mathbf{w}$ is to choose the model $\mathbf{w}^*$ which has minimum training error

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \left[\operatorname*{argmax}_{y\in\mathcal{Y}} s_{\mathbf{w}}(\mathbf{x}_i, y) \neq y_i\right] .$$

Unfortunately, finding the minimum error linear model is computationally intractable, even to approximate [11]. Instead we choose to minimize a convex upper bound on the error, also called a loss function. For binary classification, a loss function maps the score that a model gives to an instance $y\mathbf{w}'\mathbf{x}$ to a number indicating the penalty we assign to the model for scoring an instance this way. The error function itself is a kind of loss, where we assign a penalty of one to an instance whose score is less than zero and a penalty of zero to an instance whose score is greater than or equal to zero. In this thesis, we will minimize a regularized training loss

$$\operatorname*{argmin}_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} L(y_i \mathbf{w}'\mathbf{x}) + \lambda ||\mathbf{w}||_2^2 . \tag{1.1}$$

(a) Graph of loss functions versus score of a linear model for binary prediction



(b) Stochastic gradient descent algorithm for parameter estimation of a linear model

```
Input: labeled data $\{(\mathbf{x}_t, y_t)_{t=1}^{T}\}$
Output: A weight vector $\mathbf{w}^T$
for $t = 0 \dots T-1$
    $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^{t} \;-$
        $\eta \left(\frac{\delta}{\delta \mathbf{w}} L(y_{t+1}\mathbf{w}'\mathbf{x}_{t+1})|_{\mathbf{w}^t} + 2\lambda \mathbf{w}^t\right)$
end
```

Figure 1.2: **(a)** The error function together with two loss functions. The hinge loss is a continuous upper bound on error, and the Huber loss is a differentiable upper bound. **(b)** The stochastic gradient descent algorithm [69] for solving equation 1.1

This minimization problem captures in general form many common paradigms for training linear classifiers. For example, using the hinge loss

$$L(u) = \begin{cases} -(u-1), & u < 1 \\ 0, & u \geq 1 \end{cases}$$

yields a 2-norm support vector machine [58, 69]. The hinge loss is continuous but not differentiable. In order to directly apply unconstrained gradient minimization methods, we minimize a differentiable version, known as the Huber loss[2].

$$L(u) = \begin{cases} -4u, & u < -1 \\ (-u+1)^2, & -1 \leq u < 1 \\ 0, & u \geq 1 \end{cases}$$

Figure 1.2(a) depicts the error, the hinge loss, and the Huber loss as a function of the model score.

[2]Not to be confused with the Huber loss for regression, which we do not use in this thesis.

There are many choices of algorithm for finding a $\mathbf{w}^*$ which solves equation 1.1, and a thorough discussion of optimization techniques in machine learning is once again well beyond the score of this thesis. We follow Zhang [69] and use stochastic gradient descent. Stochastic gradient descent is an online algorithm parameterized by a learning rate η. Figure 1.2(b) is a description of the algorithm.

Parameter estimation for multiclass and structured models with MIRA

Parameter estimation for multiclass and structured linear predictors is more complex than for binary classification, but many of the basic aspects are similar. In this thesis, when we solve structured prediction problems, we use the margin infused relaxed algorithm (MIRA) [22]. MIRA is an online algorithm which updates the parameter vector each instance to give the minimum change to the weight vector (as measured by the L_2 norm) to separate the correct instance from the top-scoring incorrect instance by a margin. Crammer et al. [22] give a more complete description of the MIRA algorithm. The application of MIRA to structured prediction is strongly influenced by the work of Collins [19], who described an application of the perceptron to structured prediction. For a more general discussion and comparison of optimization techniques for structured predictors, we again refer to Taskar [60].

1.1.4 Generalization

In section 1.1.3, we suggested to estimate the parameters of a linear model by finding a weight vector $\mathbf{w}^*$ which minimizes a convex upper bound on the training error. We are not really interested in the training set error, however. We want to find a model which generalizes well to a new unseen test set. This section introduces elements of statistical learning theory which will allow us to give upper bounds on the expected test set error of a model in terms of its training set error. The concepts we discuss here provide an important basis for chapter 4, where we give theoretical results for generalization to new domains. Once again, we focus on binary classification and are necessarily brief, but we refer to

Kearns and Vazirani [41] and Anthony and Bartlett [6] for excellent introductions to the concepts of learning theory.

As before, we denote by $\mathbf{x} \in \mathcal{X}$ a feature vector in feature space. Formally, suppose that instances are drawn from a probability distribution $(\mathbf{x}, y) \sim \mathcal{D}$. For a particular hypothesis $h : \mathcal{X} \to \mathcal{Y}$, we define the binary indicator variable

$$[h(\mathbf{x}) \neq y] = \begin{cases} 0, & h(\mathbf{x}) = y \\ 1, & h(\mathbf{x}) \neq y \end{cases} .$$

The generalization error of a model is the expected error rate under distribution $\mathcal{D}$

$$\epsilon_{\mathcal{D}}(h) = \mathrm{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[h(\mathbf{x}) \neq y \right] .$$

Suppose that we choose a hypothesis from a class of finite cardinality $\mathcal{H}$. In this case, we may relate training and generalization error via Hoeffding's inequality [35] and the union bound. For a training sample $\{\mathbf{x}_i, y_i\}_{i=1}^N$ drawn from $\mathcal{D}$, with probability $1 - \delta$, for every $h \in \mathcal{H}$,

$$\epsilon_{\mathcal{D}}(h) \leq \frac{1}{n} \sum_{i=1}^{n} \left[h(\mathbf{x}_i) \neq y_i \right] + \sqrt{\frac{2 \log(2|\mathcal{H}|) - \log \delta}{n}} . \tag{1.2}$$

This result is a slight modification of theorem 2.3 in Anthony and Bartlett [6]. It is an example of what is known as a uniform convergence bound, since it shows that the training set error converges (as N grows large) to the generalizatoin error uniformly for every $h \in \mathcal{H}$. Note that the size of the hypothesis class $|\mathcal{H}|$ governs the rate at which the bound on training error converges to the generalization error.

A uniform convergence bound for linear models

Our linear models are parameterized by weight vectors $\mathbf{w} \in \mathbb{R}^d$. The number of possible weight vectors is clearly not finite, so the bound from equation 1.2 does not apply. We may still state a uniform convergence result, however, through a measure of hypothesis class complexity known as the Vapnik-Chervonenkis (VC) dimension [65]. For a given training sample $\mathcal{S}$ of size N, the number of possible unique partitions of the points into two classes is 2^N. But note that for a given dimension d and number of points N, not all partitions can be modeled using a linear predictor. To see this, note that for three points in two dimensions, we can model each of the eight possible partitions with linear classifiers[3], but for four points in two dimensions, we cannot model each of the 16 possible partitions with binary linear classifiers. When we can model all possible partitions of a set of points $\mathcal{S}$ with a hypothesis class $\mathcal{H}$, we will say that $\mathcal{H}$ shatters $\mathcal{S}$.

For a hypothesis class $\mathcal{H}$, the Vapnik-Chervonenkis dimension is the size of the largest subset $\mathcal{S}$ that can be shattered by $\mathcal{H}$. For linear classifiers parameterized by $\mathbf{w} \in \mathbb{R}^d$, the VC dimension is $d + 1$ [65, 6]. The VC dimension is a measure of the complexity of a hypothesis class. With it in hand, we may prove the following uniform convergence bound for linear classifiers:

Let $\mathbb{R}^d$ denote the space of parameter vectors for linear classifiers. For a training sample $\{\mathbf{x}_i, y_i\}_{i=1}^N$ drawn from $\mathcal{D}$, with probability $1 - \delta$, for every $\mathbf{w} \in \mathbb{R}^d$,

$$\epsilon_{\mathcal{D}}(\mathbf{w}) \leq \frac{1}{n} \sum_{i=1}^{n} [\mathrm{sgn}(\mathbf{w}'\mathbf{x}_i) \neq y_i] + O\left(\sqrt{\frac{d \log(N/d) - \log(\delta)}{N}}\right) . \tag{1.3}$$

Note that $d \log(N/d)$ takes the place of $\log(|\mathcal{H}|)$ in the earlier uniform convergence bound. In this bound, the larger the dimension d of the weight vector, the slower the training error converges to the generalization error.

As a final comment, we note that other measures of complexity can lead to significantly tighter bounds than VC dimension, but the precise measure of hypothesis class complexity is not essential to the theory in this thesis. We chose the VC dimension for its ease of exposition. For margin-based measures of complexity we refer to Anthony and Bartlett [6]. Shawe-Taylor and Cristianini [58] give a good introduction to data-dependent bounds and the Rademacher measure of complexity [8].

(a) Book Review

Running with Scissors: A Memoir

Title: Horrible book, horrible.

This book was horrible. I **read half** of it, suffering from **a headache** the entire time, and eventually i lit it on fire. One **less copy** in the world...don't waste your money. I wish i had the time spent reading this book back so i could use it for better purposes. This book wasted my life

(b) Kitchen Appliance Review

Avante Deep Fryer, Chrome & Black

Title: lid does **not work** well...

I love the way the Tefal deep fryer cooks, however, I am **returning** my second one due to a **defective** lid closure. The lid may close initially, but after a few uses it no longer stays closed. I will not be purchasing this one again.

Figure 1.3: Sentiment classification example: reviews of books (source) and kitchen appliances (target) from Amazon. Bold words and bigrams are at least five times more frequent in one domain than in the other.

1.2 Adapting supervised models to new domains

In the previous section, our training methods for linear models minimized a convex upper bound on the error. We motivated these methods by appealing to uniform convergence theory: For large training samples, the empirical error is a reasonable proxy for the generalization error. In many realistic applications of supervised techniques, however, the training data is drawn from a *source* distribution and the testing data is drawn from a different *target* distribution. Because of this, we cannot expect a large sample of our source data to allow us to build a good target model. In fact, as we shall see in chapter 3, realistic differences in domains can cause discriminative linear classifiers to more than double in error.

Figures 1.3 and 1.4 illustrate the differences that can appear across domains for sentiment classification and part of speech tagging, respectively. We chose these examples in particular because for each one, a linear predictor trained the source domain mis-labels

[3]as long as the three points are not co-linear

(a) Wall Street Journal

DT	JJ	VBZ	DT	NN	IN	DT	JJ	NN
The	clash	is	a	sign	of	a	new	**toughness**
CC	NN	IN	NNP	POS	JJ	JJ	NN	.
and	**divisiveness**	in	Japan	’s	**once-cozy**	**financial**	circles	.

(b) MEDLINE

DT	JJ	VBN	NNS	IN	DT	NN	NNS
The	**oncogenic**	**mutated**	forms	of	the	**ras**	proteins
VBP	RB	JJ	CC	VBP	IN	JJ	NN
are	**constitutively**	active	and	interfere	with	normal	**signal**
NN	.						
transduction	.						

Figure 1.4: Part of speech tagging example: sentences from the Wall Street Journal (source) and MEDLINE (target). Bold words are at least five times more frequent in one domain than in the other.

the target instances. In both cases, the source and target domains have very different vocabularies, and each new target vocabulary item corresponds to a new feature which is unobserved in the source domain. In the sentiment example, a classifier trained on books incorrectly identifies the “tefal deep fryer” review as positive, in part because it has never observed the negatively-skewed features `defective`, `not work`, and `returning`. In the part of speech example, a tagger trained on the Wall Street Journal (WSJ) mis-tags `signal` as an adjective, rather than correctly as a noun. The word `signal` is a frequent noun in the biomedical domain, even though it is relatively rare in the Wall Street Journal. Furthermore, `transduction` is even more rare in the Wall Street Journal, making this instance particularly hard to disambiguate.

(a) Sentiment Classification

domain	**book reviews**	**kitchen reviews**
positive correspondences	fascinating, engaging must_read, reader	are_perfect, years_now excellent_product, a_breeze
negative correspondences	plot, #_pages, predictable, reading_this	the_plastic, poorly_designed awkward_to, leaking

(b) Part of Speech Tagging

domain	**WSJ**	**MEDLINE**
adjectival correspondences	political, short-term pretty, your	metastatic, neuronal functional, transient
nominal correspondences	company, transaction investors, officials	receptors, assays mutation, lesions

Figure 1.5: Corresponding features from different domains for both tasks. These features fulfill similar roles across domains. For instance, “predictable” and “leaking” both express negative sentiment.

1.2.1 Learning feature correspondences

One recurring theme in text processing is the redundancy of features. The bold features in both figure 1.3 and figure 1.4 vary significantly in frequency across domains, but for both tasks, many unique target features have corresponding counterpart source features (figure 1.5). Suppose that we were given these correspondences. Intuitively, we should be able to use them to convert an effective source model into an effective target model by representing the weight for each target feature as the appropriate combination of source feature weights.

Pivot features and unlabeled data

Structural correspondence learning (SCL) is a method for learning these correspondences automatically from unlabeled data. The key concept behind SCL is the notion of *pivot*

(a) Examples of pivots for sentiment classification. In the target domain (kitchen appliances) all pivots co-occur with the feature `''defective''`.

book reviews	**kitchen reviews**
The book is so repetitive that I found myself yelling. . . . I will definitely **not_buy** another	Do **not_buy** the Shark portable steamer. . . . The trigger mechanism is defective
A **disappointment**. . . . Ender was talked about for #_pages altogether	the very nice lady assured me that that they must have been a defective set. . . . What a **disappointment**!
It's **unclear**. . . . It's repetitive and boring	Maybe mine was defective. . . . The directions were **unclear**

(b) Examples of pivots for part of speech tagging. In the target domain (MEDLINE) all pivots co-occur with the feature `''current word is <signal>''`.

WSJ	**MEDLINE**
of investment **required**	deliver the signal **required**
of buy-outs **from** buyers	stimulatory signal **from**
go to jail **for** violating	essential signal **for**

Table 1.1: Examples of pivots in both domains, together with the contexts in which they occur

features. Pivot features are features which occur frequently and behave similarly in both the source and target domains. The right column of table 1.1(a) shows pivot features for kitchen appliances that occur together with the word "defective". The pivot features "not buy", "disappointment", "unclear" are good indicators of negative sentiment, regardless of domain. Similarly, the right column of table 1.1(b) shows examples of PoS-tagging pivot features for WSJ and MEDLINE that occur together with the word "signal". In this case our pivot features are all of type `<the token on the right>`. Note that "signal" is unambiguously a noun in these contexts. Adjectives rarely precede past tense verbs such as "required" or prepositions such as "from" and "for".

We now search for occurrences of the pivot features in the source domains (book reviews and WSJ). The left column of tables 1.1(a) and (b) show some words that occur together with the pivot features in the source domains. In the case of sentiment classification, “repetitive”, “#_pages”, and “boring”. are all common ways of expressing negative sentiment about books. In the case of part of speech tagging, “investment”, “buy-outs”, and “jail” are all common nouns in the WSJ.

For each task structural correspondence learning uses co-occurrence between pivot and non-pivot features to learn a representation under which features from different domains are aligned. Note that we do not require labels to estimate these co-occurrences. This step only requires *unlabeled* source and target samples. Once we have this representation, we can use it to train a classifier from source data that is effective in the target domain as well. Chapter 2 describes in detail methods for choosing pivot features, learning the underlying representation, and using that representation in discriminative models.

1.2.2 Generalization to new domains

In section 1.1.4, we showed how a model can generalize from a training sample drawn from a distribution $\mathcal{D}$ to another unseen test sample from $\mathcal{D}$. For domain adaptation, however, this assumption breaks down. Our source training data is drawn from one distribution $\mathcal{D}_S$, but we need our model to generalize to target data drawn from a different distribution $\mathcal{D}_T$. Because of this we cannot use standard generalization theory to prove a bound analogous to equation 1.3.

Under what conditions on $\mathcal{D}_S$ and $\mathcal{D}_T$ can we expect to be able to train on a sample from $\mathcal{D}_S$ and perform well on a sample from $\mathcal{D}_T$? If $\mathcal{D}_S$ and $\mathcal{D}_T$ are arbitrary probability distributions on $(\mathbf{x}, y)$, then we cannot expect to learn an effective model without any labeled target data. To see this note that for binary classification, we may choose $\mathcal{D}_S$ and $\mathcal{D}_T$ to have the same marginal distribution on instances but exactly opposite distributions on labels. That is,

$$\begin{aligned}\Pr_{\mathcal{D}_T}[\mathbf{x}] &= \Pr_{\mathcal{D}_S}[\mathbf{x}] \\ \Pr_{\mathcal{D}_T}[y|\mathbf{x}] &= 1 - \Pr_{\mathcal{D}_S}[y|\mathbf{x}]\end{aligned}$$

where $\Pr_{\mathcal{D}}[\cdot]$ is the density (mass) function for distribution $\mathcal{D}$. Now the model we choose by minimizing source error converges in the limit to the worst possible target model.

Suppose we constrain $\mathcal{D}_S$ and $\mathcal{D}_T$ such that there exists some classifier $h^* \in \mathcal{H}$ which has low error on both $\mathcal{D}_S$ and $\mathcal{D}_T$. More precisely, let κ be defined as

$$\kappa = \min_{h \in \mathcal{H}} \epsilon_{\mathcal{D}_S}(h) + \epsilon_{\mathcal{D}_T}(h) \ .$$

Then we constrain $\mathcal{D}_S$ and $\mathcal{D}_T$ such that κ is small. This captures intuitively the assumption that we make when building a model for domain adaptation. For example, on our product reviews task, we expect there exists a single model which can identify reviews of both books and kitchen appliances as being positive or negative, even if finding it using only source data is difficult. With this assumption in hand, we can prove a bound on target error of the form

$$\epsilon_{\mathcal{D}_T}(h) \leq \epsilon_{\mathcal{D}_S}(h) + \kappa + \text{div}(\mathcal{D}_S, \mathcal{D}_T) \ .$$

The term $\text{div}(\mathcal{D}_S, \mathcal{D}_T)$ denotes the divergence between the source and target marginal distributions $\Pr_{\mathcal{D}_S}[\mathbf{x}]$ and $\Pr_{\mathcal{D}_T}[\mathbf{x}]$ on instances. In chapter 4, we show how to exploit the structure of a hypothesis space to derive a divergence between distributions that is computable from finite samples of *unlabeled* data. We call this divergence the $\mathcal{H}\Delta\mathcal{H}$-divergence, and for linear models, it is closely tied to feature representations we use. We use this fact to prove that structural correspondence learning finds a feature representation under which source and target distributions are close. The $\mathcal{H}\Delta\mathcal{H}$-divergence also plays a key role in our theory of learning from labeled source data and small amounts of lableled target data.

1.3 Thesis overview

Structural correspondence learning. Chapter 2 describes in detail the structural correspondence learning algorithm. We first introduce the structural learning paradigm of Ando and Zhang [3] and show why it is well-suited for domain adaptation. Then we discuss the details of SCL, including methods for selecting pivot features, and the hyperparameters necessary for combining SCL features with standard text features. Finally, we relate structural learning and SCL to other methods for using unlabeled data. We show that structural learning and SCL are closely related to the statistical method of canonical correlations analysis (CCA) [36], which recent theoretical work [40] has shown can be effective for semi-supervised learning. Aside from CCA, we also examine several other methods for using unlabeled data, including graph regularization, bootstrapping, and instance weighting, and for each we briefly discuss their feasibility for domain adaptation.

Adapting linear discriminative models with SCL. Chapter 3 illustrates the use of SCL on our sentiment classification and part of speech tagging tasks. We first demonstrate that SCL consistently makes significant reductions in error, even without any target labeled data. Then we show how to use small amounts of target data together with SCL to achieve even greater reductions in error. Finally, we examine in detail the basis found by SCL.

Learning bounds for domain adaptation. Chapter 4 develops a formal framework for analyzing domain adaptation tasks. We first show how the divergence between two domains can be computed using finite samples of unlabeled data. We use this divergence to bound the target generalization error of a model trained in the source domain. This bound depends closely on the feature representation of our model, and in particular the representation learned by SCL gives much lower values for the bound than the standard representation. Finally, we give a uniform convergence learning bound on the target generalization error of a model trained to minimize a convex combination of empirical source and target errors.

Chapter 2

Structural correspondence learning

Structural correspondence learning (SCL) is an algorithm for domain adaptation. It learns a shared representation for both source and target domains from unlabeled source and target data. SCL is a variant of the structural learning paradigm of Ando and Zhang [3]. This semisupervised method uses unlabeled data to discover a predictive linear subspace of the original hypothesis space. Section 2.1 describes structural learning, and section 2.2 introduces the SCL algorithm itself. The key idea of SCL is to exploit pivot features which are common to both domains. We describe methods for choosing effective pivots, as well as hyperparamters used for combining SCL with standard supervised learning methods.

The latter part of this chapter is devoted to exploring the connection between SCL and other semi-supervised learning techniques. By dividing up the feature space into pivot and non-pivot features, structural learning and SCL are effectively exploiting “multiple views” of the input data. Recent theoretical work has examined canonical correlation analysis (CCA) [36] to learn a norm for semi-supervised multi-view regression [40]. We show that structural learning and SCL are closely related to CCA, and we discuss the implications of this for domain adaptation. Finally, we briefly review other methods for using unlabeled data and discuss their suitability for domain adaptation.

2.1 Structural learning

Supervised learning methods find a hypothesis which generalizes well from a labeled training set to new data. For the linear models explored in this thesis, the space of hypotheses consists of linear functions of the features present in an instance. Thus the complexity of the hypothesis space is directly related to the size of the feature space. Strucutral learning [3] aims to learn a new, reduced-complexity hypothesis space by exploiting regularities in feature space via unlabeled data. For text these regularities come in the form of lexical features that function similarly for prediction. For instance, prepositions such as “with”, “on”, and “in” all are likely to precede noun phrases. Recognizing this is helpful for part of speech tagging.

Structural learning characterizes feature space regularities through “auxiliary problems” which are carefully chosen using knowledge about the supervised task. Structural learning finds a linear subspace of the original hypothesis space, where all of the auxiliary problems can be solved well using predictors from this subspace. If this new hypothesis space is much smaller than the original, but contains an equally accurate best hypothesis, we may expect to achieve better accuracy for smaller amounts of data, as compared to the original hypothesis space.

2.1.1 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

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

For realistic problems, we do not have the true distribution available to us, but only a finite sample. One method for choosing f given $\mathcal{H}$ is the method of regularized empirical risk minimization. For a sample of size N, we solve the optimization problem

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} \sum_{i=1}^{N} \frac{1}{N} L\left(f(\mathbf{x}_i), y_i\right) + ||f||_2^2 \, .$$

In semisupervised learning, in addition to our labeled sample, we also are endowed with a large amount of unlabeled data. The basic idea behind structural learning is to learn a hypothesis class $\mathcal{H}_\Phi$ using the unlabeled data, where Φ parameterizes the space of hypothesis classes. Then we choose our predictor from $\mathcal{H}_\Phi$. If the hypothesis class we learn is good, then we expect to be able to choose a better function f from our labeled sample.

2.1.2 Shared structure via auxiliary problems

From now on we will refer to the classification problem for which we have labeled data as the *supervised* 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 supervised problem. For instance, all the auxiliary problems we discuss here will use the same feature set as the supervised problem.

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

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

For example, suppose our supervised 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 word binary classification problems, one for each of the one thousand most frequent left words. For more detail on auxiliary problems, we refer the reader to section 2.2.1 which discusses the pivot features we use for SCL.

Since auxiliary problems require only unlabeled data to create, we can train reliable auxiliary predictors from the unlabeled data. If the auxiliary problems are diverse, we can further 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 supervised problem, any common structure they have is likely to also be shared by a good supervised predictor. Thus if we can discover a good *predictor subspace* from our auxiliary predictors, this subspace can serve as our hypothesis space.

2.1.3 Joint empirical risk minimization

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 [3] suggest to choose the linear predictor subspace parameterized by the matrix $\Phi \in \mathbb{R}^{k \times V}$ which minimizes the regularized empirical risk of all the auxiliary problems. 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 Φ.

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

Ando and Zhang [3] 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 2.1.3, we discuss the the actual implementation that Ando and Zhang [3] use in their experiments in section 2.1.4.

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 write $\mathbf{u}_\ell = \mathbf{w}_\ell - \Phi'\mathbf{v}_\ell$. Now we can rewrite the optimization problem as

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

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

1. Fix $(\Phi, \mathbf{v})$ and optimize with respect to $\mathbf{u}$.

2. Fix $\mathbf{u}$ and optimize with respect to $(\Phi, \mathbf{v})$.

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 [3] suggest stochastic gradient descent. We focus now on step 2, which for fixed $\mathbf{u}_\ell = \hat{\mathbf{u}}_\ell$ yields the optimization problem

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

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

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

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

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

```
Input:   labeled data $\{(\mathbf{x}_t, y_t)_{t=1}^T\}$,
         unlabeled data $\{\mathbf{x}_j\}$

Output:  predictor $f : X \to Y$

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. $W = [\hat{\mathbf{w}}_1 | \dots | \hat{\mathbf{w}}_m]$,   If $W_{i,\ell} < 0$, set $W_{i,\ell} = 0$.

4. $[U\ D\ V'] = \text{SVD}(W)$,   $\Phi = U'_{[1:k,:]}$

5. Return $f$, a predictor trained on $\left\{ \left( \begin{bmatrix} \mathbf{x}_t \\ \Phi\mathbf{x}_i \end{bmatrix}, y_t \right)_{t=1}^T \right\}$
```

Figure 2.1: ASO algorithm as it is implemented in practice

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

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

Let $W = [\mathbf{u}_1, \dots, \mathbf{u}_m]$ be the matrix whose columns are the weight vectors $\mathbf{u}$. The solution has the form

$$\Phi\Lambda = WW'\Phi \ ,$$

with Λ diagonal. Together with the orthogonality constraint, we know that the columns of Φ are eigenvectors of the covariance matrix WW'. Thus we can also solve the optimization problem above with a singular value decomposition.

2.1.4 The ASO algorithm in practice

One could run the alternating structural optimization as described in the previous section to find Φ, but in order to achieve the results that Ando and Zhang [3] report, we must make several changes to the form of the algorithm. In this section, we briefly describe these changes. The final, simpler algorithm is shown in figure 2.1.

One iteration of optimization

The first change from the ASO algorithm as described in section 2.1.3 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, Φ 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 $\Phi = \mathbf{0}^{k \times V},\ \mathbf{w}_\ell = \mathbf{0},\ \mathbf{v}_\ell = \mathbf{0}\ \forall \ell$, 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.

Only positive entries in W

The second important change to ASO is that when constructing the matrix W whose columns are the weight vectors $\mathbf{w}_\ell$, we set all the negative entries $W_{i,\ell} = 0$ 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, 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

An example weight matrix W, where columns are auxiliary predictor weight vectors and rows are features. The grayed block is the submatrix for feature type $\mathcal{T}_k$.

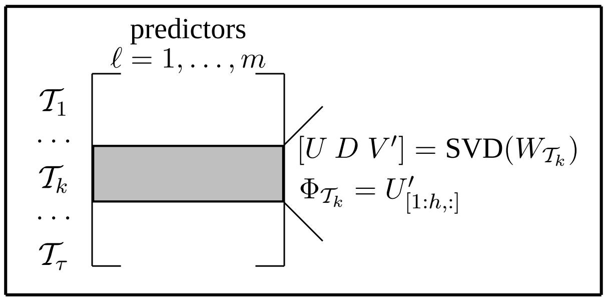

predictors
$\ell = 1, \ldots, m$
$\mathcal{T}_1$
$\mathcal{T}_k$
$\mathcal{T}_\tau$
$[U\ D\ V'] = \mathrm{SVD}(W_{\mathcal{T}_k})$
$\Phi_{\mathcal{T}_k} = U'_{[1:h,:]}$

Figure 2.2: An illustration of block SVD by type, from Ando and Zhang [3].

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.

Dimensionality reduction by feature type

Ando and Zhang [3] also suggest an extension that computes separate singular value decompositions for blocks of weights corresponding to what they 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 point out that these feature types are not homogenous and should not necessarily be represented with the same projection Φ. They suggest performing an SVD just on the submatrix corresponding to a specific feature type (figure 2.2). Then, during supervised training and testing, the matrices $\Phi_{\mathcal{T}}$ are applied to the appropriate types separately, and the features are concatenated into a single vector.

Training a supervised model using Φ

The final step of the algorithm (step 5 in figure 2.1) is to train a linear predictor on labeled data. Notice that instead of completely replacing the hypothesis space, we augment the original feature vector $\mathbf{x}$ with $\Phi\mathbf{x}$. In practice Ando and Zhang [3] suggest to train a single linear predictor by minimizing the combined loss

$$\operatorname*{argmin}_{\mathbf{w},\mathbf{v}} \left(\sum_j L(\mathbf{w}'\mathbf{x}_j + \mathbf{v}'\Phi\mathbf{x}_j, y_j) + \lambda ||\mathbf{w}||^2 \right) . \tag{2.1}$$

We regularize the weight vector $\mathbf{w}$ which multiplies the original feature vector, but not the weight vector $\mathbf{v}$ which multiplies the low-dimensional transformed feature vector. This assymetric regularization encourages the model to use the low-dimensional representation rather than the original high-dimensional representation, but it allows the original feature representation if necessary.

Applications of ASO

A complete discussion of the tasks to which ASO has been applied is beyond the scope of this thesis, but in addition to Ando and Zhang [3], we refer the reader to Ando [4] for a discussion of its application to word sense disambiguation. Ando et al. [5] applied ASO for information retrieval, and more recently Liu and Ng [44] applied ASO to the task of semantic role labeling.

2.2 The SCL algorithm

Domain adaptation methods encounter different problems from the semi-supervised learning setting which motivates structural learning. When adapting from one domain to another, we may have a large number of source domain training instances, but because the target distribution is different from the source, we still can't estimate good statistics for it. Structural correspondence learning (SCL) operates on labeled source training data and unlabeled source and target training data. The goal of SCL is to design a small number of features which are useful predictors in *both* the source and the target domains. The most important part of SCL is the selection of pivot features. Pivot features correspond to the auxiliary problems of structural learning, and they provide the mechanism for relating the two domains. Choosing good pivot features is thus essential for good performance. Section 2.2.1 gives examples of pivot features and discusses how to select them. As with ASO, the final step of SCL is a singular value decomposition of a pivot predictor matrix. Applying the final projection matrix Φ to an instance $\Phi\mathbf{x}$ results in a low-dimensional,

```
Input:    labeled data from source domain $\{(\mathbf{x}_t, y_t)_{t=1}^T\}$,
          unlabeled data from both domains $\{\mathbf{x}_j\}$

Output:   predictor $f : X \to Y$

1.  Choose $m$ pivot features (section 2.2.1).

2.  Create $m$ binary prediction problems, $p_\ell(\mathbf{x}),\ \ell = 1 \dots m$

3.  For $\ell = 1$ to $m$
        $\hat{\mathbf{w}}_\ell = \text{argmin}\,\mathbf{w}\Big( \sum_j L(\mathbf{w} \cdot \mathbf{x}_j, p_\ell(\mathbf{x}_j)) + \lambda ||\mathbf{w}||^2 \Big)$
    end

4.  $W = [\hat{\mathbf{w}}_1 | \dots | \hat{\mathbf{w}}_m],\quad [U\ D\ V'] = \text{SVD}(W) \quad \Phi = U'_{1:k,:}$

5.  Return $f$, a predictor trained on $\left\{ \left( \begin{bmatrix} \mathbf{x}_t \\ \Phi \mathbf{x}_i \end{bmatrix}, y_t \right)_{t=1}^T \right\}$
```

Figure 2.3: SCL algorithm

dense feature representation.

The SCL algorithm is given in Figure 2.3. Step 1 of the algorithm is the choice of pivot features. After this, the remainder of the algorithm is closely based on on ASO, and because the two have so much in common, we refer the reader to section 2.1 regarding details that are not present in this section.

2.2.1 Pivot features

Pivot features in SCL play the same role as auxiliary problems in ASO. They should occur frequently in the unlabeled data of both domains, since we must estimate their co-occurrence with non-pivot features accurately. At the same time, they must also be sufficiently predictive for the supervised task, since we will build a representation using them. Pivots that don't meet both of these criteria cannot help us learn a good representation for adaptation.

Pivot features for sentiment classification

Sentiment classification is the task of labeling each document with whether or not it expresses positive or negative sentiment. We use the standard bag-of-words representation, where the features are words and bigrams. Each word is weighted by its term frequency and the values are normalized sum to one for each document. Pivot features are features which occur in more than k documents in both source and target domains. Then, for each document, we create a pivot predictors for problems of the form "`Does the pivot feature <w> occur in this document?`". From table 1.1(a), we can create the pivot problem "`Does the bigram <not buy> occur in this document?`", for example. When predicting a particular pivot we remove this feature from the feature vector (or equivalently, always give it 0 weight).

Pivot features for part of speech tagging

Part of speech tagging is a sequence labeling problem with many heterogeneous features. Given a sentence, the task of a part of speech tagger is to label each word with its grammatical function. The best part of speech taggers encode a sentence label as a chain-structured graph [53, 20, 62]. In this formulation, the part of speech label factors along the cliques of the graph. We will design pivot features for individual cliques and the input features associated with them. Consider the edge ending with the tag for "`signal`" in the phrase with "`normal signal transduction`" (The correct label for this edge is `JJ-NN`). We create pivots from left, middle, and right words that occur more than k times in both corpora. Then we create pivot predictors for problems of the form "`Is the left/middle/right word for this edge <w>?`". From table 1.1(b), we can create the pivot problem "`Is the right word for this edge <required>?`". When predicting a left word, we remove all features related to that word from the input feature vector.

Pivot predictors and correspondences

Each pivot predictor implicitly aligns features from separate domains. For example, if non-pivot features from both the source and target domains are both highly predictive for the binary problem "`Is the right word for this edge <required>?`", then after step 3 of SCL (figure 2.3), they will both have weights in the linear classifier which solves this problem. If two non-pivot features have high weights across many pivot prediction problems, then we have significant reason to believe they should correspond. In step 4, when we perform the SVD on the weight matrix, these features will have a similar representation in the low dimensional basis.

Choosing pivot features

We choose frequent features to function as pivots, and as we shall see, this often results in good representations for domain adaptation. But up to now, we have not chosen pivots by explicitly taking into account the supervised problem we'd like to perform. While we do not have target labeled data, we can potentially make use of a large amount of *source* labeled data to select pivots which are more targeted at the underlying supervised learning problem. A simple way to do this is to score a feature $\mathbf{x}^i$ based on the conditional entropy of y given $\mathbf{x}^i$:

$$H(Y|\mathbf{x}^i) = -\left(\log \frac{c(\mathbf{x}^i, +1)}{c(\mathbf{x}^i)} + \log \frac{c(\mathbf{x}^i, -1)}{c(\mathbf{x}_i)}\right) .$$

Here $c(\mathbf{x}^i)$ indicates the empirical count of feature $\mathbf{x}^i$, and $c(\mathbf{x}^i, \cdot)$ indicates the joint empirical count of that feature with a particular label. Section 3.1.3 explores results using SCL with pivots selected using conditional entropy.

2.2.2 Feature normalization and scaling

Section 2.1.4 describes the version of ASO that Ando and Zhang [3] apply in practice. In this section, we list and further address two practical issues beyond those mentioned in Ando and Zhang [3]. When combining dense features (from the projection under Φ) and standard sparse features, we need to normalize and scale the dense features for use with standard optimizers.

Centering and normalization. When we receive a new instance $\mathbf{x}$, we perform the projection $\Phi\mathbf{x}$, yielding k new real-valued features. For each of these features, we center it by subtracting out the mean on the training data. Then we normalize each feature to have unit variance on the training data. This is especially important for supervised tasks with multiple feature types. Since the projections are orthonormal, projections for feature types such as bigrams tend to have much smaller values (and correspondingly small variance), whereas projections for feature types such as prefixes and suffixes have much larger values. These differences in feature values make it difficult to use online and stochastic optimization techniques effectively.

Scaling. The sparse features that we use for standard text processing have an intrinsic scale. For bag-of-words representations of documents, our representation requires that the feature values for each instance sum to 1. For the binary representations we use in part of speech tagging, the feature values for each instance sum to a number much larger than 1. The real-valued features, however, all have unit variance after normalization. Because of this, we scale the values of the real-valued features using a single scaling factor $\alpha\Phi\mathbf{x}$, which we set on heldout data.

2.3 ASO and SCL as multiple-view learning algorithms

Structural learning and SCL model the inherent redundancy in text features. For sentiment analysis, this redundancy comes in the form of multiple words used to express positive or negative sentiment. For part of speech tagging, this comes in the form of orthographic versus contextual sources of information. Many times, the orthography of a word provides important information about its part of speech. Similarly, the context in which a word appears also can provide information.

One way to analyze this feature redundancy is to split the feature space into multiple "views" [17]. Learning in the two views model proceeds by training separate classifiers for each view and requiring that they "agree" on the unlabeled data [26, 1, 2, 29, 55]. In this section, we show how to relate ASO and SCL to new theoretical work on using canonical correlation analysis for multiple view learning [36, 40]. We show that a variant of the ASO optimization problem is equivalent to the optimization solved by CCA. Kakade and Foster [40] give simple conditions under which the CCA-learned feature space allows for faster convergence to optimal linear regression parameters than the original feature space. While these conditions are insufficient to give a complete theory of domain adaptation with SCL, they provide intuitions about when SCL can succeed.

2.3.1 Canonical correlation analysis

Let $\mathbf{X} \sim \mathcal{D}$ be a random variable which is divided into two views. We will write $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ to denote the portions of $\mathbf{X}$ that are specific to each view. Note that $\mathcal{D}$ is a joint distribution on both views. Canonical correlation analysis finds two sets of basis vectors such that for all k, the projections of $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ onto the first k bases are maximally correlated [36, 33]. Let C be the joint covariance matrix for $\left(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}\right)$

$$C = \mathrm{E}_{\mathbf{x} \sim \mathcal{D}}\left[\mathbf{x}\mathbf{x}'\right] \ .$$

We may write C in the block form

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \ ,$$

where $C_{11} = \mathrm{E}_{\mathbf{x}^{(1)} \sim \mathcal{D}}\left[\mathbf{x}^{(1)}\mathbf{x}^{(1)'}\right]$ and likewise for the other blocks. Finally, we abuse notation and write C for a covariance matrix approximated from a finite sample. The appropriate interpretation should be clear from context.

Let $\mathbf{a}_i^*$ and $\mathbf{b}_i^*$ be the ith canonical basis vectors found by CCA for views one and two, respectively. $\mathbf{a}_i^*$ and $\mathbf{b}_i^*$ are the solutions to the following optimization problem:

$$\begin{aligned}
&\mathrm{argmax}_{\mathbf{a}_i,\mathbf{b}_i}\ \mathbf{a}_i' C_{12} \mathbf{b}_i' \\
&\text{s.t.} \quad && \mathbf{a}_i' C_{11} \mathbf{a}_i = 1 \\
& && \mathbf{b}_i' C_{22} \mathbf{b}_i = 1 \\
& && \mathbf{b}_i' C_{22} \mathbf{b}_i = 1 \\
&\forall j \leq i && \mathbf{a}_i' \mathbf{a}_j = 0 \\
&\forall j \leq i && \mathbf{b}_i' \mathbf{b}_j = 0 \\
&\forall j \leq i && \mathbf{a}_i' \mathbf{b}_j = 0
\end{aligned}$$

A complete derivation of the solution to this optimization is beyond the scope of this thesis, but the optimization may be solved efficiently as an eigenvalue problem [33]. Let $C_{11} = R_{11} R_{11}'$ be the Cholesky factorization of C_{11}, where R_{11} is lower triangular. Then the columns of A are the solutions to

$$R_{11}^{-1} C_{12} C_{22}^{-1} C_{21} R_{11}^{-1'} \mathbf{a}_j = \rho^2 \mathbf{a}_j \ . \tag{2.2}$$

2.3.2 Multiple-view regression with CCA

Suppose we partition our feature space into two views $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. Intuitively, CCA captures the maximum “agreement” that may be captured by linear transformations of the two views. Kakade and Foster [40] show that for linear regression under the squared loss, CCA provides a new norm for regularization. This norm can in turn lead to significantly decreased complexity when compared with the original norm. This section is a brief review of the Kakade and Foster result.

In the following derivation, let $\mathcal{D}$ be a joint distribution on pairs $(\mathbf{x}, y)$. We wish to find the solution to the minimization problem

$$\mathbf{w}^* = \underset{w}{\mathrm{argmin}}\, \mathrm{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[(\mathbf{w}'\mathbf{x} - y)^2 \right] \ .$$

For each view $\nu \in \{1, 2\}$, we may also examine the minimum error regression using just the features in that view:

$$\mathbf{w}^{(\nu)*} = \operatorname*{argmin}_{\mathbf{w}^{(\nu)}} \mathrm{E}_{\left(\mathbf{x}^{(\nu)},y\right)} \left[\left(\mathbf{w}^{(\nu)'} \mathbf{x}^{(\nu)} - y \right)^2 \right] .$$

The assumption that Kakade and Foster make on the distribution $\mathcal{D}$ is that the optimal regression from each view has low regret with respect to the optimal joint regression:

$$\mathrm{E}_{\mathbf{x}^{(\nu)},y} \left[\left(\mathbf{w}^{(\nu)*'} \mathbf{x}^{(\nu)} - y \right)^2 \right] - \mathrm{E}_{\mathbf{x},y} \left[\left(\mathbf{w}^{*'} \mathbf{x} - y \right)^2 \right] \leq \epsilon . \tag{2.3}$$

This assumption does not involve a notion of independence between the two views. In fact, the resulting bound holds even for completely dependent views. In contrast to previous theories where the views were assumed to satisfy some notion of independence [17, 26, 1], Kakade and Foster incorporate the amount of *correlation* between the views as part of the resulting bound.

With this assumption in hand, Kakade and Foster suggest the following procedure for finding a good vector $\mathbf{w}$: Compute the canonical basis vectors $\mathbf{a}_i$ and $\mathbf{b}_i$, the solutions to $CCA(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$. We give the regression procedure for view 1 (the procedures for the two views are identical). Let ρ_i indicate the correlation of the canonical basis projections $\mathbf{a}_i'\mathbf{x}_i^{(1)}$ and $\mathbf{b}_i'\mathbf{x}_i^{(2)}$. If we denote by A the matrix whose columns are the canonical basis vectors for view 1, then define

$$\begin{aligned} \hat{\mathbf{x}}^{(1)} &= \rho A' \mathbf{x}^{(1)} \\ ||w^{(1)}||^2_{\text{CCA}} &= \textstyle\sum_i \frac{1-\rho_i}{\rho_i} \left(\mathbf{w}_i^{(1)} \right)^2 . \end{aligned}$$

Now solve the following ridge regression problem:

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}^{(1)}} \mathrm{E}_{\mathbf{x}^{(1)},y} \left[\left(\mathbf{w}^{(1)'} \hat{\mathbf{x}}^{(1)} - y \right)^2 + ||\mathbf{w}^{(1)}||^2_{\text{CCA}} \right] .$$

The main theorem of the paper bounds the bias and variance of $\hat{\mathbf{w}^{(1)}}$.

Theorem 1 *Suppose that the assumption from equation 2.3 holds and* $\mathrm{E}\left[y^2|\mathbf{x}\right] \leq 1$. *Let* T *be a training set consisting of* n *pairs* $(\mathbf{x}, y)$ *drawn from* $\mathcal{D}$. *Then*

$$\mathrm{E}_T \left[\left(\hat{\mathbf{w}}^{(1)'} \mathbf{x}^{(1)} - y \right)^2 \right] \leq \mathrm{E}_{(\mathbf{x},y)} \left[\left(\mathbf{w}^{*'} \mathbf{x} - y \right)^2 \right] + 5\epsilon + \frac{\sum_i \rho_i^2}{n} .$$

Let us briefly examine this bound. The bias term 5ϵ depends on the assumption from equation 2.3. The variance term depends on the amount of correlation between the two views. If the two views are highly correlated, then this term is high, and we cannot expect to learn from few examples. If, on the other hand, the views are conditionally independent given y, then there will only be one non-zero correlation coefficient, corresponding to the optimal regression.

2.3.3 Relating CCA and structural learning

Structural learning is not identical to CCA, but we may analyze a variant of structural learning that solves an optimization problem identical to that of CCA, subject to certain constraints on the generating distribution. We briefly outline the algorithmic changes to ASO (figure 2.1) here.

Multiple views, auxiliary problems, and pivot features

We first divide our feature space into multiple views. For some tasks, this is natural. For instance, when using SCL to learn representations for part of speech taggers, we already partition pivots into “context” (left and right words) and “content” (middle words). For sentiment, we can randomly partition the words in the vocabulary into two views [3]. Now we treat every feature as an auxiliary problem. For each auxiliary problem generated from a feature in view 1, we only use features from view 2 when training a predictor for it. For those in view 2, we only use features from view 1 when training predictors. For domain adaptation, it will be useful divide our views into a view consisting of only pivot features and one consisting of only non-pivot features.

Squared loss

The third step of SCL and structural learning is to train predictors for each view. For this discussion, we set the loss to be the squared loss $L(\mathbf{w} \cdot \mathbf{x}_j, y) = (\mathbf{w} \cdot \mathbf{x}_j - y)^2$.

Within-view whitening

Before we train the predictors, we whiten each view individually by pre-multiplying by $R_{\nu\nu}^{-1}$, where $R_{\nu\nu}R'_{\nu\nu} = C_{\nu\nu}$, as in section 2.3.1. This has creates identity within-view covariance matrices $C_{11} = C_{22} = I$.

Block SVD by view

Now let us examine the matrix W. The columns of W are linear predictors, the solution to the optimization problems from step 3 in algorithm 2.1. Since we don't use features from the same view to make predictions within that view, we note that W has a block off-diagonal form:

$$W = \begin{bmatrix} 0 & W^{(1)} \\ W^{(2)} & 0 \end{bmatrix} .$$

Following Ando and Zhang's suggested extension (section 2.1.4), we focus here on separate SVDs for each off-diagonal block, $W^{(\nu)}$ for $\nu \in \{1, 2\}$.

Equivalence of ASO and CCA

We begin by partitioning the features into views and whitening. Let us first focus on $W^{(1)}$. Our modified ASO algorithm finds $W^{(1)}$ by solving the multiple least-squares optimization problem

$$W^{(1)} = \operatorname*{argmin}_V ||V'R_{11}^{-1}X^{(1)} - R_{22}^{-1}X^{(2)}||_F^2 .$$

The solution to this problem is given by

$$W^{(1)} = (R_{11}^{-1}X^{(1)}X^{(1)'}R_{11}^{-1'})^{-1}R_{11}^{-1}X^{(1)}X^{(2)'}R_{22}^{-1'} .$$

By definition, the first term in the product simplifies to the identity matrix. We are left with

$$W^{(1)} = R_{11}^{-1}X^{(1)}X^{(2)'}R_{22}^{-1'} .$$

The columns of $\Phi^{(1)}$ are the left singular vectors of $W^{(1)}$. Equivalently, they are the eigenvectors of $W^{(1)}W^{(1)'}$, which gives us the following form for $\boldsymbol{\phi}_j$:

$$\begin{aligned} R_{11}^{-1}X^{(1)}X^{(2)'}R_{22}^{-1'}R_{22}^{-1}X^{(2)}X^{(1)'}R_{11}^{-1'}\boldsymbol{\phi}_j &= \rho^2\boldsymbol{\phi}_j \\ R_{11}^{-1}C_{12}C_{22}^{-1}C_{21}R_{11}^{-1'}\boldsymbol{\phi}_j &= \rho^2\boldsymbol{\phi}_j \ . \end{aligned}$$

This is exactly the CCA optimization from equation 2.2. Thus our modifications allow us to view the work of Kakade and Foster [40] as a theoretical treatment of ASO as well as CCA. When we run SCL in practice, we don't whiten the data or use squared loss, however, and it is unclear whether these changes are essential to the performance of structural learning and SCL.

2.3.4 Implications for domain adaptation

Structural correspondence learning works by training linear predictors for each pivot. These predictors are linear mappings from non-pivot features to binary labels indicating the presence or absence of each pivot. If we divide our feature space into two views, one each for pivot and non-pivot features, the relationship to CCA allows us to interpret SCL as finding a low-dimensional basis under which linear predictors trained using non-pivot features are highly correlated with linear predictors trained using pivot predictors.

For semisupervised learning, the theory of Kakade and Foster [40] leads us directly to a bound on the error of a predictor. Unfortunately the same is not true for domain adaptation. The main assumption of Kakade and Foster is that a good predictor can be trained from each view in isolation. In the case of SCL, we may interpret this as indicating that a good predictor may be trained from the pivot features alone. Empirically, however, we have found that this is not true. Non-pivot features *are* necessary for prediction in the target domain (see chapter 3). An important goal for future work is to develop assumptions under which SCL can be directly shown to perform well.

2.4 Other methods for exploiting unlabeled data

SCL uses unlabeled target domain data to find a new feature representation. This feature representation allows us to train an effective target predictor using source training data. This is by no means the only method for exploiting unlabeled data, though. In this section, we briefly review semi-supervised and unsupervised methods in text, with an emphasis on applicability for domain adaptation. Our survey here is necessarily brief, but see Zhu [70] for a more complete survey of semi-supervised learning methods.

2.4.1 Manifold regularization

Procedurally, the most similar methods to structural learning are those which learn a regularizer from unlabeled data [68, 9, 71]. Like structural learning, these methods regularize parameters by enforcing smoothness in some underlying subspace. The assumptions on the structure of the subspace are quite different, though. The simplest methods use the singular value decomposition of the unlabeled data to learn a basis for a linear subspace [68]. Rather than learning predictors, we simply choose the top singular vectors of the unlabeled data matrix to serve as our basis. While this may be an effective subspace for semi-supervised learning, there is no clear connection between squared reconstruction error and error of the predictive task.

One of the most natural ways to regularize predictors is to enforce smoothness between nearby points. That is, for a real-valued predictor, we require that the predictions be close for points that are close. How can we decide which points are close, though? One way to proceed is the data manifold assumption [9, 72]: We assume that the input instances $\mathbf{x}$ are sampled from a low dimensional manifold. The neighborhood graph on the unlabeled data can provide us with an indication of the structure of that manifold.

Suppose we construct a graph whose nodes are instances and whose edges indicate similarity between instances. Let S_{ij} be the square weight matrix for the edges in the

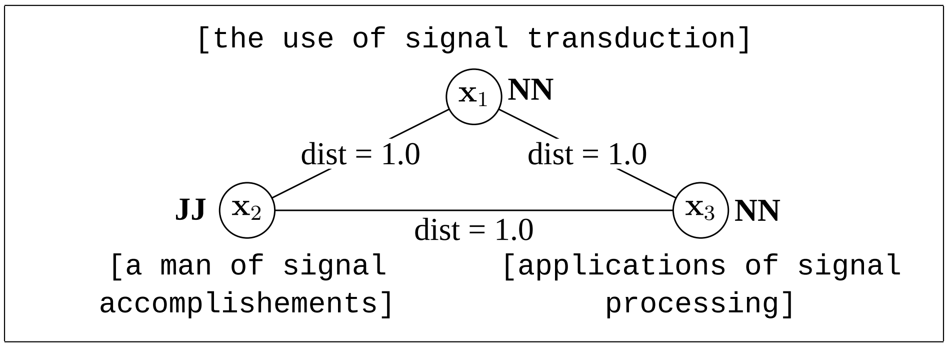

[the use of signal transduction]
$\mathbf{x}_1$ NN
dist = 1.0
dist = 1.0
JJ $\mathbf{x}_2$
$\mathbf{x}_3$ NN
dist = 1.0
[a man of signal accomplishements]
[applications of signal processing]

Figure 2.4: An example of how feature vector hamming distance can be misleading. Nodes in the graph represent instances, and for each instance we wish to tag the word "signal" with its part of speech. If we represent each instance using a three-word window, all instances have hamming distance 1 from one another. Distance in feature space is insufficient for classification.

graph. One common choice for the similarity function is a Gaussian kernel

$$S_{ij} = \exp\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{\sigma^2}\right) .$$

Given a weight matrix W and a hypothesis class $\mathcal{H}$, Belkin et al. [9] suggest the following regularized optimization problem

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) + \frac{\gamma}{N+U} \sum_{i,j=1}^{N+U} S_{ij} \left(f(x_i) - f(x_j)\right)^2 ,$$

where N is the number of labeled instances and U is the number of unlabeled instances. Belkin and Niyogi and others have suggested variants of this regularization, including learning linear combinations of the bottom eigenvectors of the Laplacian of the neighborhood graph [9, 73].

For the manifold regularizer to be effective, the design of the neighborhood graph must reflect the structure of the input space. For continuous feature spaces, a large unlabeled neighborhood graph may yield a good approximation to the true manifold. For discrete feature spaces, the hamming distance is often insufficient, even for large data sets. This is because changing just one feature can change the label of an instance, and hamming distance treats all features equally.

Figure 2.4 illustrates this. Although each instance has hamming distance 1 to each other instance, $\mathbf{x}_2$ is labeled differently from $\mathbf{x}_1$ and $\mathbf{x}_3$. Knowing that “transduction” and “processing” are more similar than either one is to “accomplishments” is necessary to give correct distances here.

2.4.2 Bootstrapping

Another paradigm for exploiting unlabeled data is bootstrapping [67, 17, 49, 21, 1, 47]. Bootstrapping methods begin with an initial classifier. They label unlabeled instances with this classifier. Then they choose some subset of the newly-labeled instances to create a new training set and retrain. There are many variants of bootstrapping for semi-supervised learning, and it is well beyond the scope of this thesis to analyze all of them in depth. Here we briefly discuss their relation to SCL and application to domain adaptation.

One of the most well-studied methods for bootstrapping is co-training [17]. Co-training was the first semi-supervised learning method to formally articulate the notion of two views. Blum and Mitchell [17] give an algorithm which trains classifiers for each view separately. Then the classifier for one view is used to label instances to train the other. Blum and Mitchell analyze their co-training algorithm in the PAC setting. They show that if each view is sufficient for classification and the views are conditionally independent given the label, then given initial weak predictors for each view, co-training finds an accurate model using only unlabeled data. Co-training is connected to structural learning via CCA. When analyzing CCA for multiple view regression, Kakade and Foster [40] also assume that each view is sufficient (has low regret) for regression. But the amount of conditional independence (correlation) appears as a term in their bound, rather than an assumption.

Jiang and Zhai [39] studied a bootstrapping method for domain adaptation of text processing models. They trained a model in the source domain. Then they labeled the target domain instances about which the model was most confident and retrained on them. They report good positive empirical results using this self-training approach on several text problems, but their work contains no theoretical analysis or discussion of when self-training could be effective for domain adaptation.

2.4.3 Covariate shift and instance weighting

One area problem that is very closely-related to domain adaptation is the problem of covariate shift (also called sample selection bias), which has been studied in the machine learning and statistics communities [59, 37]. Here we assume the conditional distributions $\Pr_{D_S}[y|\mathbf{x}]$ and $\Pr_{D_T}[y|\mathbf{x}]$ are identical, but the instance marginal distributions $\Pr_{\mathcal{D}_S}[\mathbf{x}]$ and $\Pr_{\mathcal{D}_T}[\mathbf{x}]$ are different.

Several researchers have studied algorithms for regression in this setting [59, 37]. Like our domain adaptation setting, they assumed that they had unlabeled data from both source and target domains. Unlike our application to text, though, they focused primarily on problems with low-dimensional, dense continuous features. They first estimate the posterior probability that a particular source instance has been drawn from the target distribution (for instance, under a kernel density estimate of the target distribution). Then they minimize an “instance re-weighted” source error objective. That is, each instance is given a weight α_i which depends on its weight under the estimated target. Similar to our previous minimization problem, we now minimize

$$\sum_{i=1}^{N} \alpha_i L\left(f(\mathbf{w}, \mathbf{x}_i, y_i))\right) + ||\mathbf{w}||_2^2 \,.$$

Standard kernel density estimators like the Gaussian kernel are effective in low dimensional, continuous spaces, but as we discuss in section 2.4.1, instance weighting is less effective for high-dimensional, sparse feature spaces such as those for text. From a theoretical standpoint, the covariate shift assumption is too weak to allow us to prove a concise, computable bound for target error under arbitrary source and target marginal distributions on unlabeled instances (see chapter 4).

2.5 Summary

This chapter described the structural correspondence learning (SCL) algorithm, a method for using unlabeled source and target data to learn a shared feature representation that is simultaneously effective for both domains. SCL uses the technique of structural learning [3] to find a low-dimensional linear subspace of the orginal feature space. If this subspace is effective, a good source predictor which uses it is automatically a good predictor for the target domain. *An essential component to SCL is the notion of a pivot feature. Pivot features are features that are important for classification and are shared between both source and target domains.* SCL can be seen as modeling a low-dimensional subspace of the non-pivot features that correlates as much as possible with the pivot features.

We related SCL and structural learning to canonical correlation analysis (CCA) [36], a statistical method for discovering correlating basis vectors for two multivariate random variables. Recent work by Kakade and Foster [40] has shown that under relatively weak assumptions, CCA on unlabeled data can discover an effective low-dimensional basis for prediction. Using this basis to find a predictor on labeled data can lead to much faster convergence. *We believe that in the future, this relationship will be helpful in formulating a theoretical analysis of when SCL can help for domain adaptation.*

Chapter 3

Experiments with SCL

This chapter examines in detail the application of SCL to two text processing tasks: sentiment analysis of product reviews and part of speech tagging. We treat sentiment analysis as a binary classification problem (either positive or negative sentiment). Part of speech tagging is a structured prediction problem, where the task is to label a sentence with a sequence of part of speech labels. For both of these problems, linear predictors achieve state-of-the-art results, making them ideal for an empirical investigation of SCL.

We examine the two tasks separately, but performing parallel experiments. First, we attempt to illustrate intuitively why SCL should help us in domain adaptation. We do this by examining how domain-specific features are represented in the low-dimensional subspace it discovers. The latter part of each section gives numerical error rates showing that SCL does indeed improve linear models for these problems. We first show that when we have no labeled target data, SCL significantly reduces error, sometimes by a relative amount of more than twenty percent. Second, we address the case where we do have a small amount of labeled target domain data. Under these circumstances, several authors have proposed techniques for combining source and target data effectively. We show that combining SCL with these methods yields still greater improvements, reducing error due to adaptation by as much as forty percent. The results in this chapter are drawn primarily from Blitzer et al. [16] and Blitzer et al. [15].

3.1 Adapting a sentiment classification system

A sentiment classification system receives as input a document and outputs a label indicating the sentiment (positive or negative) of the document. This problem has received considerable attention recently [51, 63, 32]. While movie reviews have been the most studied domain, sentiment analysis has been extended to a number of new domains, ranging from stock message boards to congressional floor debates [25, 61]. Research results have been deployed industrially in systems that gauge market reaction and summarize opinion from web pages, discussion boards, and blogs.

With such widely-varying domains, researchers and engineers who build sentiment classification systems need to collect and curate data for each new domain they encounter. Even in the case of market analysis, if automatic sentiment classification were to be used across a wide range of domains, the effort to annotate corpora for each domain may become prohibitive, especially since product features change over time. We envision a scenario in which developers annotate corpora for a small number of domains, train classifiers on those corpora, and then apply them to other similar corpora. The case for domain adaptation is immediately clear, since documents from different domains can vary widely in the ways they express sentiment.

We constructed a dataset for sentiment domain adaptation by selecting Amazon product reviews for four different product types: books, DVDs, electronics and kitchen appliances. Each review consists of a rating (0-5 stars), a reviewer name and location, a product name, a review title and date, and the review text. Reviews with rating > 3 were labeled positive, those with rating < 3 were labeled negative, and the rest discarded because their polarity was ambiguous. After this conversion, we had 1000 positive and 1000 negative examples for each domain, the same balanced composition as the polarity dataset [51]. In addition to the labeled data, we included between 3685 (DVDs) and 5945 (kitchen) instances of unlabeled data. The size of the unlabeled data was limited primarily by the number of reviews we could crawl and download from the Amazon website. Since we were able to obtain labels for all of the reviews, we also ensured that they were balanced between positive and negative examples, as well.

3.1.1 Problem setup and representation

We created feature vectors from each review by using the unigrams and bigrams from the review title and text. Each unigram and bigram is associated with one dimension of feature space, and the value for each dimension the count for that feature in that document. If we let $c(i,j)$ be the count of the jth feature in the ith document in the corpus, then the dimensions of the feature vector $\mathbf{x}_i$ have the form

$$\mathbf{x}_i^j = \frac{c(i,j)}{\sum_{k=1}^{d} c(i,k)} \ .$$

This roughly follows the setup of Pang et al. [51] for sentiment classification, although they also included trigrams in their features. In preliminary experiments, adding trigram features did not improve performance.

We split each labeled dataset into a training set of 1600 instances and a test set of 400 instances. All of our experiments use a classifier trained on the training set of one domain and tested on the test set of a possibly different domain. The total number of features varies among pairs of domains, but for all pairs, we used approximately 200,000 features (the dimensionality of $\mathbf{x}$ was 200,000). Our baseline is a linear classifier trained without adaptation, while the gold standard is an in-domain classifier trained on the same domain as it is tested. When we train supervised predictors, we minimize the Huber loss with stochastic gradient descent, as described in chapter 1. On the polarity dataset, this model matches the results reported by Pang et al. [51].

Pivot features and scaling

We chose pivot features for adapting sentiment classifiers using one of two procedures. The first procedure is exactly the method described for sentiment in section 2.2.1: Choose as pivots words and bigrams that occur more than k times in both the source and target

Selected by frequency only, not conditional entropy	Selected by conditional entropy only, not frequency
book one <num> so all	*a_must a_wonderful loved_it*
very about they like	*weak don't_waste awful*
good when	*highly_recommended and_easy*

Table 3.1: Top pivots selected by frequency, but not conditional entropy (left) and vice-versa (right)

domains. We set k to be the largest number such that we have at least 1000 pivots. We refer to this procedure as "selection by frequency".

Section 2.2.1 describes another criterion for choosing pivots: the conditional entropy of the label y given the presence that particular pivot feature. For a particular pair of domains, we use this by setting k to be the largest number such that we have at least 10,000 potential pivots. Then we sort these potential pivots by conditional entropy and choose the top 1000. We refer to this procedure as "selection by conditional entropy". Table 3.1 shows the set-symmetric differences between the two methods for pivot selection when adapting a classifier from books to kitchen appliances.

We set h, the number of singular vectors computed in the final SVD, to 50 throughout these experiments. We also followed section 2.2.2 in normalizing the parameters. For scaling, we first scaled the real-valued features $\Phi\mathbf{x}$ so that the average norm of the real-valued feature vector for each training instance was one

$$\frac{1}{N}\sum_{t=1}^{N} |\Phi\mathbf{x}_t| = 1 \ .$$

Then we set $\alpha = 0.1$ for all of the experiments (see section 2.2.2) , based on heldout data from the books and kitchen appliances data sets.

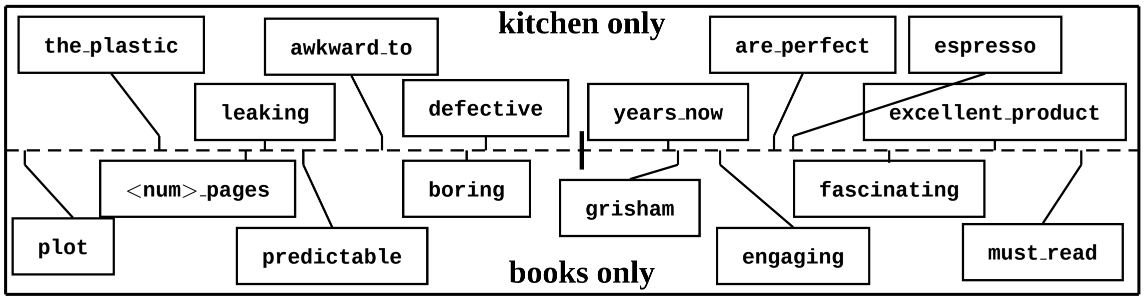


Figure 3.1: A discriminating projection of word features onto the real line, for books and kitchen appliances. Words on the left (negative valued) behave similarly each other for classification, but differently from words on the right (positive valued). Above the horizontal axis are words that only occur in the kitchen appliances domain. Below the horizontal axis are words that only occur in the books domain.

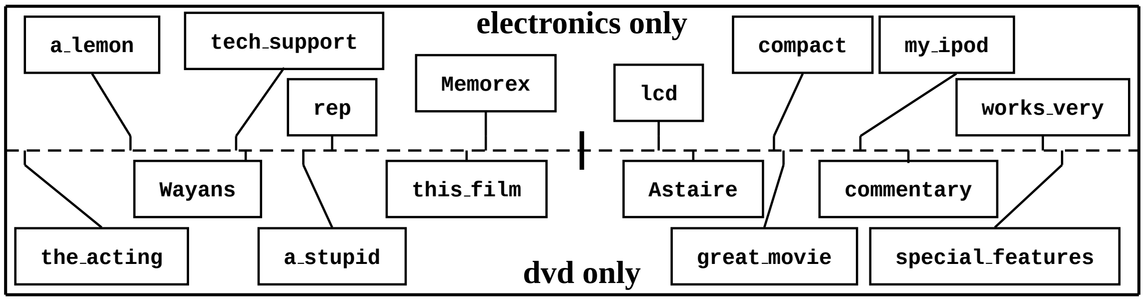


Figure 3.2: A discriminating projection of word features onto the real line, for the DVDs and electronics domains. Words on the left (negative valued) behave similarly each other for classification, but differently from words on the right (positive valued). Above the horizontal axis are words that only occur in the books domain. Below the horizontal axis are words that only occur in the DVDs domain.

3.1.2 The structure of Φ

In chapters 1 and 2, we motivated SCL by claiming that it would encode correspondences such as those in figure 1.5. Here we illustrate the correspondences that SCL actually does learn by examining the rows of the matrix Φ. Recall that the inner product of each row with an instance vector $\mathbf{x}$ yields a single new real-valued feature $\Phi_{[i,:]}\mathbf{x}$. From equation 2.1, we see that this new feature is associated with a weight $\mathbf{v}_i$. If we expand the inner product, for a particular instance $\mathbf{x}$, the score contributed by the new real-valued feature is

$$\mathbf{v}_i \Phi_{[i,:]}\mathbf{x} = \mathbf{v}_i \sum_j \Phi_{ij}\mathbf{x}_i \ .$$

In particular, we note that if different features $\mathbf{x}_{j1}$ and $\mathbf{x}_{j2}$ have similar entries in the ith row of Φ, then they effectively share a single parameter $\mathbf{v}_i$.

Figures 3.1 and 3.2 illustrate these projections. Each plot shows a single row of the matrix Φ, along with the most positively-valued and most negatively-valued features for that row. Let us briefly examine the projections from figure 3.1. The words above the horizontal axis never appear in our books training data, but we still may assign them some weight (via $\mathbf{v}_i$), as long as we observe the unqiue book-specific words depicted below the horizontal axis. For instance, since `predictable` and `leaking` have similar values under $\Phi_{[:,i]}$, when we observe `leaking` at test time, it will contribute to the decision rule as though we had observed `predictable`. Figure 3.2 illustrates a similar discriminating projection for DVDs and electronics.

These illustrations depict how SCL can intuitively perform well. But we note here that not all projections are discriminating by themselves. The final decision rule is the linear combination of the features $\Phi\mathbf{x}$. In practice, many projections are not visibly discriminating, and may not be related to the supervised task at all. With labeled source data, however, we can learn to ignore those projections. Indeed, we may be able to find a good linear predictor even when there is no single good discriminating projection.

3.1.3 Empirical results: only unlabeled target data

In this section, we investigate empirically the most commonly-encountered domain adaptation setting: We have a labeled training sample from a source domain and large unlabeled

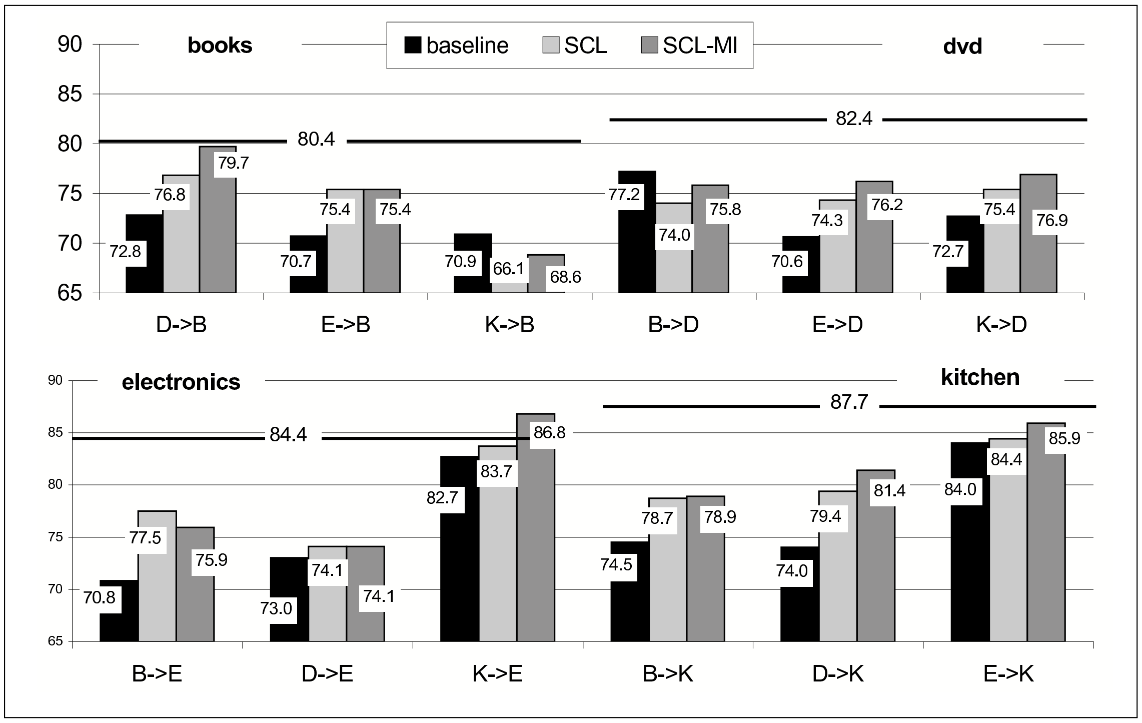


Figure 3.3: Sentiment classification accuracies for domain adaptation between all pairs using a supervised baseline, SCL and SCL-CE. Horizontal black lines are the accuracies of in-domain classifiers.

samples from both source and target domains. We first choose pivots according the procedures outlined in section 3.1.1. Then we find the matrix Φ using the unlabeled data from both domains and train a supervised model using the labeled source data, combining the SCL and original sparse features.

Figure 3.3 gives accuracies for domain adaptation across all pairs of our sentiment domains. The target domains are ordered clockwise from the top left: books, DVDs, kitchen appliances, and electronics. Each group of three bars represents a single source domain (denoted by the first letter). The three bars themselves are our supervised baseline (black), SCL with frequency-based pivots (light gray), and SCL with pointwise conditional entropy-based pivots (dark gray). The thick horizontal bars are the accuracies of the in-domain classifiers for these domains. These classifiers are trained on the 1600-instance training set and tested on the 400-instance test set of the same domain and represent a gold standard for the target domain.

To help interpret the results, the top left set of bars (from books to DVDs) shows that the baseline achieves 72.8% accuracy adapting from DVDs to books. Choosing pivots based on their conditional entropy with the labels and running SCL achieves a 79.7% accuracy and the in-domain gold standard is 80.4%. We say that the *adaptation loss* for the baseline model is 7.6% and the adaptation loss for the SCL-CE model is 0.7%. The *relative reduction in error due to adaptation* of SCL-CE for this test is 90.8%.

We can observe from these results that there is a rough grouping of our domains. Books and DVDs are similar, as are kitchen appliances and electronics, but the two groups are different from one another. Adapting classifiers from books to DVDs, for instance, is easier than adapting them from books to kitchen appliances. We note that when transferring from kitchen to electronics, SCL-CE actually outperforms the in-domain classifier. This is possible since the unlabeled data may contain information that the in-domain classifier does not have access to.

3.1.4 Empirical results: some labeled target data

domain	**features with similar values under** $\Phi_{[i,:]}$
books	book_was, reading_this, this_story, characters
kitchen appliances	was_defective, was_broken, noisy, this_purchase

Table 3.2: An example of a mis-alignment for the books and kitchen appliances domains. The kitchen appliances-specific features are associated with negative sentiment, but the books-specific features are not associated with either positive or negative sentiment.

The SCL-CE model improves over the baseline in 10 out of 12 cases, but there are two pairs of domains in which SCL-CE causes error to increase. This is because while SCL generally creates good representations for adaptation, it may also misalign features.

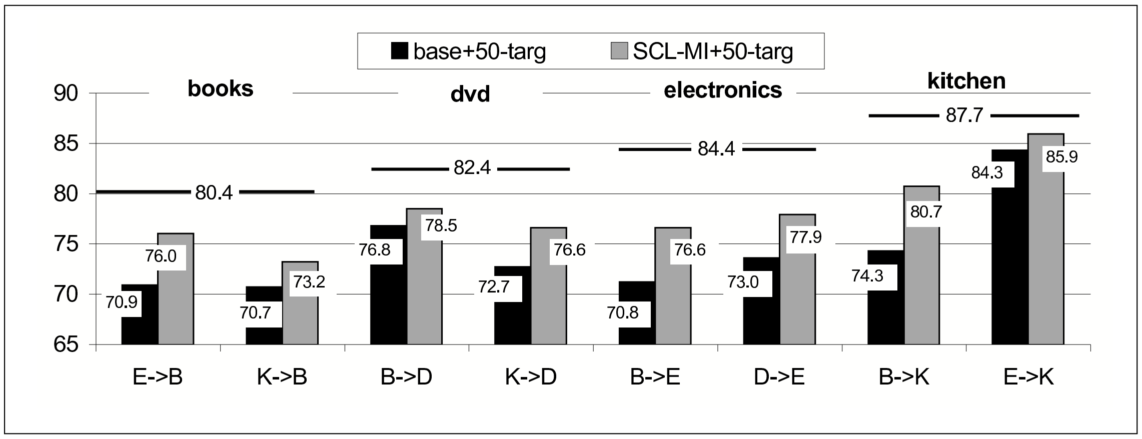


Figure 3.4: Sentiment classification accuracies for domain adaptation with 50 labeled target domain instances.

We designed the pivots to reflect intuitions about class membership, but we did not enforce class membership in our final SCL representation. Table 3.2 illustrates one example of mis-aligned features, for the books and kitchen appliances domain. For this pair of domains, the relative error due to adaptation increases by 24% under the SCL-CE model.

Despite the increase in error, though, we saw in section 3.1.2 that there are correct discriminating projections for domain paris such as books and kitchen appliances. Here we propose to exploit small amounts of labeled target domain data to slightly adjust the parameters $\mathbf{v}$ for the SCL features. The intuition is that even with very small amounts of target data, the number of SCL parameters is small enough to learn an effective correction from source to target. Using the notation of Ando and Zhang [3], we can write the supervised training objective of SCL on the source domain as

$$\min_{\mathbf{w},\mathbf{v}} \sum_i L\left(\mathbf{w}^T\mathbf{x}_i + \mathbf{v}^T\mathbf{\Phi}^T\mathbf{x}_i, y_i\right) + \lambda||\mathbf{w}||^2 + \mu||\mathbf{v}||^2 ,$$

where y is the label. The weight vector $\mathbf{w} \in \mathbb{R}^d$ weighs the original features, while $\mathbf{v} \in \mathbb{R}^k$ weighs the projected features. Ando and Zhang [3] suggest $\lambda = 10^{-4}, \mu = 0$, which we have used in our results so far.

Suppose now that we have trained source model weight vectors $\mathbf{w}_s$ and $\mathbf{v}_s$. A small amount of target domain data is probably insufficient to significantly change $\mathbf{w}$, but we can correct $\mathbf{v}$, which is much smaller. We augment each labeled target instance $\mathbf{x}_j$ with the label assigned by the source domain classifier [30, 16]. Then we solve

$$\min_{\mathbf{w},\mathbf{v}} \sum_j L\left(\mathbf{w}'\mathbf{x}_j + \mathbf{v}'\theta\mathbf{x}_j, y_j\right) + \lambda||\mathbf{w}||^2 \\ +\mu||\mathbf{v} - \mathbf{v}_s||^2 \,.$$

Since we don't want to deviate significantly from the source parameters, we set $\lambda = \mu = 10^{-1}$.

Figure 3.4 shows the corrected SCL-CE model using 50 target domain labeled instances. We chose this number since we believe it to be a reasonable amount for a single engineer to label with minimal effort. For each target domain we show adaptation from only the two domains on which SCL-CE performed the worst relative to the supervised baseline. For example, the book domain shows only results from electronics and kitchen, but not DVDs. As a baseline, we used the label of the source domain classifier as a feature in the target, but did not use any SCL features. We note that the baseline is very close to just using the source domain classifier, because with only 50 target domain instances we do not have enough data to relearn all of the parameters in $\mathbf{w}$. As we can see, though, relearning the 50 parameters in $\mathbf{v}$ is quite helpful. The corrected model *always* improves over the baseline for every possible transfer, including those not shown in the figure.

The idea of using the regularizer of a linear model to encourage the target parameters to be close to the source parameters has been used previously in domain adaptation. In particular, Chelba and Acero [18] showed how this technique can be effective for capitalization adaptation. The major difference between our approach and theirs is that we penalize deviation from the source parameters for the weights $\mathbf{v}$ of projected features, while they work with the weights of the original features. As we may expect, for our small amount of labeled target data and large number of features, attempting to penalize $\mathbf{w}$ using $\mathbf{w}_s$ performed no better than our baseline. Because we only need to learn to ignore projections that misalign features, we can make much better use of our labeled data by adapting only 50 parameters, rather than 200,000.

Table 3.3 summarizes the results of sections 3.1.3 and 3.1.4. Structural correspondence

dom \ model	base	base +targ	scl	scl-mi	scl-mi +targ	large-ul Φ-only
books	8.9	9.0	7.4	5.8	**4.4**	1.8
dvd	8.9	8.9	7.8	6.1	**5.3**	3.8
electronics	8.3	8.5	5.9	5.5	**4.8**	1.3
kitchen	10.2	9.9	7.0	5.6	**5.1**	3.9
average	9.1	9.1	7.1	5.8	**4.9**	2.7

Table 3.3: Summary of sentiment classification results. Each row shows the average loss due to adaptation for each method for a single target domain, averaged over all source domains. The bottom row shows the loss averaged over all source domains runs.

learning reduces the error due to transfer by 21%. Choosing pivots by mutual information allows us to further reduce the error to 36%. Finally, by adding 50 instances of target domain data and using this to correct the misaligned projections, we achieve an average relative reduction in error of 46%.

The final column of table 3.3 shows a new set of results obtained by training on a much larger dataset. In this case, we doubled the amount of unlabeled data in each domain and trained on only the low-dimensional representation of the data. While these numbers represent a significant improvement even over the previous SCL results, we note that they are only directly comparable to the original supervised baseline.[1] They demonstrate that in many cases, the SCL representation alone can give a significant improvement over the baseline, even without using the original features.

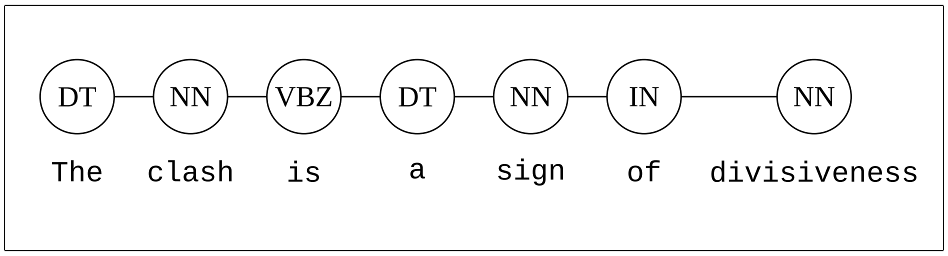


Figure 3.5: A chain-structured graphical representation of a sentence and its part of speech tag sequence. We factor the label sequence (and the ζ vector) along the edges of the graph to allow for efficient inference [42, 19].

3.2 Adapting a part of speech tagger

A part of speech tagger takes as input a sentence and outputs a sequence of labels indicating the part of speech tags for each word (figure 3.5). Part of speech tagging is a canonical problem in text processing, and it serves as a first step in many pipelined systems, including higher-level syntactic processing [21, 47], information extraction [56, 52], and machine translation [66]. Because of their fundamental role, part of speech tagging systems must be deployed in a variety of domains. In this section, we show how to use SCL to adapt a tagger from a standard resource, the Penn treebank Wall Street Journal (WSJ) corpus [46] to a new corpus of MEDLINE abstracts [52].

The Penn BioIE project [52] focuses on building information extraction and natural language processing systems for biomedical text. We obtained a corpus from this project consisting of 200,000 sentences that were chosen by searching MEDLINE for abstracts pertaining to cancer, in particular genomic variations and mutations. This corpus contains as a subset a smaller corpus consisting of 1061 sentences that have been annotated by linguists with part of speech tags. The Penn treebank corpus consists of forty thousand annotated sentences. In this section our goal is to adapt a tagger trained on the treebank corpus to perform well on the MEDLINE corpus.

[1]In addition to the extra unlabeled data, they also use a more recent version of the SGD optimization algorithm.

	word	**prefix**	**suffix**
left	`The`	`The`, `Th`	`The`, `he`
middle	`clash`	`clas`, `cla`, `cl`	`lash`, `ash`, `sh`
right	`is`	`is`	`is`
left-mid	`The_clash`	`The_clas`, `The_cla`, `The_cl`,`Th_cl`,...	`The_lash`, `The_ash` `The_sh`, `he_sh`,...
left-right	`The_is`	`The_is`, `Th_is`,	`The_is`, `he_is`
mid-right	`clash_is`	`clas_is`, `cla_is`, `cl_is`	`lash_is`, `ash_is`, `sh_is`

Table 3.4: The types of features we use for part of speech tagging. Each cell represents one type, and the entries of the cells are example instantiations for the edge of the graph in figure 3.5 ending with `sign`. In this case, the middle word is `sign`.

3.2.1 Problem setup and representation

The part of speech tagset we use consists of the standard Penn treebank tagset, augmented with two new tags: HYPH (for hyphens) and AFX (for common post-modifiers of biomedical entities such as genes). These tags were introduced due to the importance of hyphenated entities in biomedical text, and are used for 1.8% of the words in the test set. Any tagger trained only on WSJ text will automatically predict wrong tags for those words.

We treat part of speech tagging as a sequence labeling problem. Figure 3.5 shows an example sentence, together with a graph depicting how we factor the label. When the label (and the ζ vector) factors along the edges of the graph, we may perform efficient inference using dynamic programming [42, 19]. This allows us to compute the best sequence for a particular sentence given a model. With the ability to perform efficient inference, we can use any number of models to learn an appropriate linear model. We choose the discriminative online large-margin learning algorithm MIRA [22].

Now that we have chosen an appropriate factorization, we must choose features for a particular edge in the graph. Table 3.4 depicts the feature types we use, together with examples of the features that would be instantiated for the edge ending in `sign`. We generate the $\boldsymbol{\zeta}$ vector by concatenating each of these features with the set of possible labels. We also add one entry to the $\boldsymbol{\zeta}$ vector for the identity of the tags at each end of the edge. In the example from table 3.4, this entry would be `DT-NN`. Each of these features, and each of the entries in the $\boldsymbol{\zeta}$ vector is represented as a single binary number: 1 if it is present in an instance and 0 if it is not present. In total, across 200,000 sentences from both domains, we created 5 million features (the dimensionality of $\mathbf{x}$ was 5 million).

Pivot features and model choices

We chose pivot features using the method from section 2.2.1. In all the experiments of this section, we use left, middle, and right words that occur more than 50 times in both corpora. State of the art part of speech taggers require feature types of different granularities. The total number of types is equal to 18, the number of entries in table 3.4. We perform a per-type dimensionality reduction, and for each type sub-matrix, we set $h = 25$. This gives us a total of 450 dense features. Just as for sentiment classification we normalize and scale the real-valued features. We first scaled the real-valued features so that the average 1-norm of the sparse and dense features is the same for each training instance

$$\frac{1}{N}\sum_{t=1}^{N} |\Phi \mathbf{x}_t| = \frac{1}{N}\sum_{t=1}^{N} |\mathbf{x}_t| \ .$$

Then we set $\alpha = 1$ for all of the experiments, based on heldout Wall Street Journal data.

3.2.2 The structure of Φ

As in section 3.1.2, in this section we explore the structure of the SCL representation by representing pictorially the entries of the matrix Φ. Unlike for sentiment classification, in these experiments we have divided up the matrix Φ into several separate matrices depending on the feature type. Figure 3.6 shows entries Φ_{ij} in a single row $\Phi_{[i,:]}$ of the projection matrix for the `middle word` type. As before, we chose a discriminating projection. Part of speech tagging is a multiclass problem so a single dimension will not allow us to make

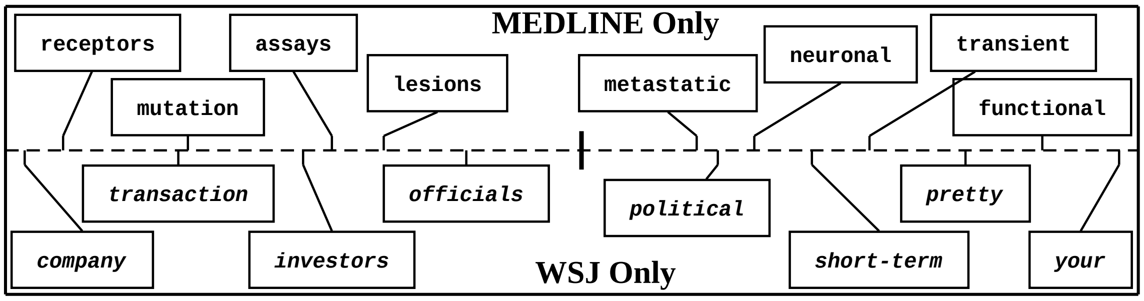


Figure 3.6: An example projection of word features onto $\mathbb{R}$. Words on the left (negative valued) behave similarly to each other for classification, but differently from words on the right (positive valued). The projection distinguishes nouns from adjectives and determiners in both domains.

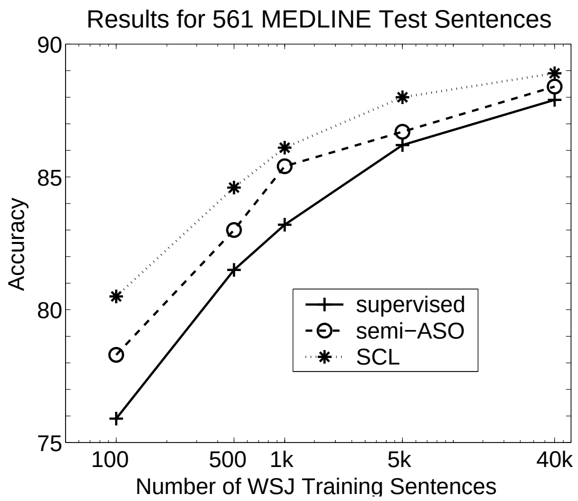
all possible discriminations. The row we chose distinguishes between nouns (negative under $\Phi_{[i,:]}$) and adjectives (positive under $\Phi_{[i,:]}$).

Again, we emphasize that when training a tagger in the Wall Street Journal, we can implicitly assign weight to the MEDLINE-only features above the horizontal axis via the projection $\Phi_{[i,:]}$. Since the word `receptors` has a similar value under $\Phi_{[i,:]}$ to the WSJ-specific words `company` and `transaction`, we can incorporate it into a decision rule at test time in MEDLINE. Finally, we wish to emphasize once again that while these projections can give us clues as to how SCL can improve predictor accuracy, the actual decision rule is a linear combination of 450 real-valued features. Even without a single good discriminating projection, we may still be able to find a good linear model.

3.2.3 Empirical results: only unlabeled target data

For part of speech tagging, we only have one pair of domains. Moreover, we only have a large amount of labeled data for a single domain (the Wall Street Journal). Thus for this problem, we cannot investigate the performance of SCL across varying pairs of domains. Instead we investigate learning curves for SCL with increasing amounts of source data. The graph in figure 3.7(a) shows three curves. The solid curve is a supervised MIRA

(a)



(b) Accuracy on test set

	Words	
Model	All	Unknown
MXPOST[53]	87.2	65.2
supervised	87.9	68.4
semi-ASO	88.4	70.9
SCL	**88.9**	**72.0**

(c) Statistical Significance (McNemar's) for all words

Null Hypothesis	**p-value**
semi-ASO vs. super	0.0015
SCL vs. super	2.1×10^{-12}
SCL vs. semi-ASO	0.0003

Figure 3.7: Part of speech tagging results with unlabeled target training data

baseline which does not use any unlabeled data. The dashed curve is a semi-supervised baseline using ASO. Here we treated the target domain as unlabeled and learned an ASO representation from 200,000 MEDLINE sentences, but we didn't attempt to choose common pivots or induce correspondences. Finally, the dotted curve is the SCL model. For the points on this curve, we learned an SCL representation from 100,000 Wall Street Journal and MEDLINE sentences, following the procedures from section 3.2.1.

The horizontal axis of figure 3.7(a) shows increasing amounts of source training data. With one hundred sentences of training data, structural correspondence learning gives a 19.1% relative reduction in error over the supervised baseline, and it consistently outperforms both baseline models. Figure 3.7(b) gives results for 40,000 sentences, and Figure 3.7(c) shows corresponding significance tests, with $p < 0.05$ being significant. We use a McNemar paired test for labeling disagreements [31]. Even when we use all the

WSJ training data available, the SCL model significantly improves accuracy over both the supervised and ASO baselines.

SCL is designed to improve the accuracies for unknown words (words that have never before been seen in the WSJ). For our final set of experiments, we investigated unknown word accuracy more directly. The second column of Figure 3.7(b) gives unknown word accuracies on the biomedical data. Of thirteen thousand test instances, approximately three thousand were unknown. For unknown words, SCL gives a relative reduction in error of 19.5% over MXPOST [53], a common out-of-the-box baseline, even with 40,000 sentences of source domain training data.

Improving a parser in the target domain

At the beginning of section 3.2, we motivated our investigation of part of speech tagging by emphasizing its importance as a first step in many pipe-lined text processing systems. Here we show that improving a part of speech tagger in a new domain can improve a dependency parser in the new domain as well. We use the parser described by McDonald et al. [48]. That parser assumes that a sentence has been PoS-tagged before parsing, so it is a straightforward match for our experiments here.

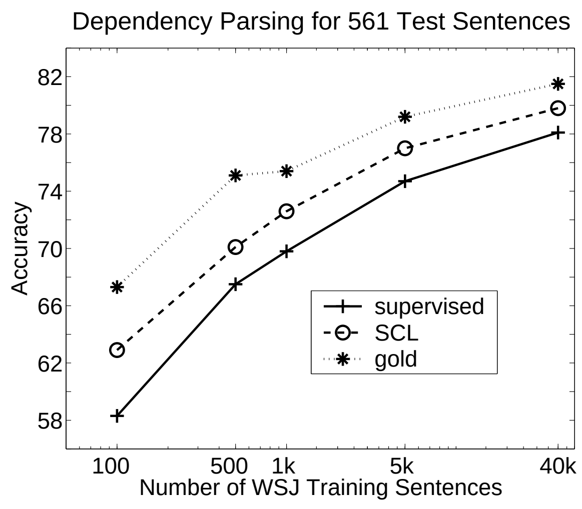
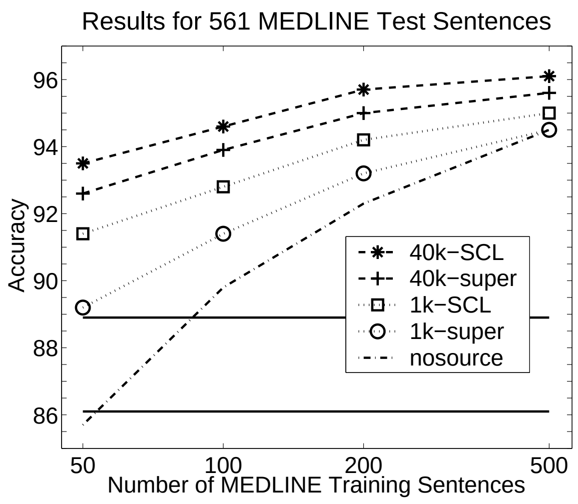


Figure 3.8: Dependency parsing results using different part of speech taggers

Figure 3.8 shows dependency parsing accuracy for increasing amounts of Wall Street Journal data. At test time, we tag the MEDLINE data with one of three taggers. Then we use the output of this tagger as input to the dependency parser. The first tagger (thick black line) is a supervised baseline. The second (dashed line) is trained using SCL. The third (dotted line) is the gold standard, where instead of using an automatic tagger we just use the correct tags from the annotation. The SCL tags consistently improve parsing performance over the tags output by the supervised tagger, closing the gap between the baseline and the gold standard by about 50%.

3.2.4 Empirical results: some labeled target data

(a)



(b) 500 target training sentences

Model	Testing Accuracy
nosource	94.5
1k-super	94.5
1k-SCL	**95.0**
40k-super	95.6
40k-SCL	**96.1**

(c) McNemar's Test

Null Hypothesis	p-value
1k-super vs. nosource	0.732
1k-SCL vs. 1k-super	0.0003
40k-super vs. nosource	2×10^{-12}
40k-SCL vs. 40k-super	6×10^{-7}

Figure 3.9: PoS tagging results with no target labeled training data

We now examine a setting in which we have a small amount of labeled MEDLINE data. As with sentiment classification, the key idea is to use the labeled target data to make adjustments to a model trained in the source domain. Unlike in sentiment classification, however, for part of speech tagging we exploit the structured nature of the problem. In particular, we note here that the output of the source classifier gives information not only about the tag of the current word, but also about the tag of nearby words. In order to use this information, we first train a tagger on the Wall Street Journal. Then we create features from the output of the source model and use these features for training and testing in the target [30]. We use as features the currently predicted tag and all tag bigrams in a 5-word window around the current word. For example, suppose the source tagger labels `with normal signal transduction` as `IN JJ JJ NN`. One feature we would create for the window centered at `signal` is the feature `source_left_mid_tag_bigram=JJ_JJ`.

Naturally we expect the accuracy of the source-trained tagger in the target domain to affect the quality of the features we create. In these experiments, we show how using features from an SCL-based source tagger can significantly improve the final target-trained tagger. Figure 3.9(a) plots tagging accuracy for varying amounts of MEDLINE training data. The two horizontal lines are the fixed accuracies of the SCL WSJ-trained taggers using one thousand and forty thousand sentences of training data. The five learning curves are for taggers trained with varying amounts of target domain training data. They use features on the outputs of taggers from section 3.2.3. The legend indicates the kinds of features used in the target domain (in addition to the standard features). For example, "40k-SCL" means that the tagger uses features on the outputs of an SCL source tagger trained on forty thousand sentences of WSJ data. "nosource" indicates a target tagger that did not use any tagger trained on the source domain. With 1000 source domain sentences and 50 target domain sentences, using SCL tagger features gives a 20.4% relative reduction in error over using supervised tagger features and a 39.9% relative reduction in error over using no source features.

Figure 3.9(b) is a table of accuracies for 500 target domain training sentences, and Figure 3.9(c) gives corresponding significance scores. With 1000 source domain sentences and 500 target domain sentences, using supervised tagger features gives no improvement over using no source features. Using SCL tagger features still does, however.

3.3 Related work

In this chapter we showed how SCL can improve linear discriminative models, both when we have no labeled target data at all, and when we have small amounts of labeled target data. Related work that does not use any target labeled data is scarce, but we address it, as well as other mdethods for semi-supervised learning in section 2.4. Here we focus on related work which uses labeled target data, related work on sentiment classification and part of speech tagging in general, and finally the small amount of work on adapting sentiment classifiers and PoS taggers.

3.3.1 Using labeled target data for domain adaptation

Although it is preferable to avoid labeling data in the target domain altogether, labeled data is still by far the most useful resource to have. Furthermore, when we do have some labeled data, we should be able to exploit it as best as possible. Our intention in this chapter is not to advocate one method for using labeled target data in particular. We already showed that SCL is compatible with two common methods for incorporating labeled target data, and here we briefly examine the space of such methods.

We combine SCL with the method of Chelba and Acero [18] in section 3.1.4. They begin by training a linear model on the source domain. Then they use maximum a posteriori estimation of the weights of a maximum entropy target domain classifier. The prior is Gaussian with mean equal to the weights of the source domain classifier. Florian et al. [30] describe the method we use in section 3.2.4. They train a model on the target domain using the output of the source model as a feature.

In addition to these methods, there have been several other investigations of domain adaptation. Roark and Bacchiani [54] use a Dirichlet prior on the multinomial parameters of a generative parsing model to combine a large amount of training data from a source corpus (WSJ), and small amount of training data from a target corpus (Brown). Daume and Marcu [28] use an empirical Bayes model to estimate a latent variable model grouping instances into domain-specific or common across both domains. They also jointly estimate the parameters of the common classification model and the domain specific classification models. Daume [27] gives a simple feature duplication method that performs surprisingly well when the target training data yields a sufficiently accurate model on its own. Because SCL combines easily with linear classification methods, we emphasize that it is compatible with any of these schemes for exploiting labeled data.

3.3.2 Sentiment classification

Sentiment classification has advanced considerably since the work of Pang et al. [51], which we use as our baseline. Thomas et al. [61] use discourse structure present in congressional records to perform more accurate sentiment classification. Pang and Lee [50] treat sentiment analysis as an ordinal ranking problem. In our work we only show improvement for the basic model, but all of these new techniques also make use of lexical features. Thus we believe that our adaptation methods could be also applied to those more refined models.

While work on domain adaptation for sentiment classifiers is sparse, it is worth noting that other researchers have investigated unsupervised and semisupervised methods for domain adaptation. The work most similar in spirit to ours that of Turney [63]. He used the difference in mutual information with two human-selected features (the words “excellent” and “poor”) to score features in a completely unsupervised manner. Then he classified documents according to various functions of these mutual information scores. We stress that our method improves a supervised baseline. While we do not have a direct comparison, we note that [63] performs worse on movie reviews than on his other datasets, the same type of data as the polarity dataset.

We also note the work of Aue and Gammon [7], who performed a number of empirical tests on domain adaptation of sentiment classifiers. Most of these tests were unsuccessful.

Their most significant results were on combining a number of source domains. They observed that source domains closer to the target helped more. In preliminary experiments we confirmed these results. Adding more labeled data always helps, but diversifying training data does not. For example, when classifying kitchen appliances, for any fixed amount of labeled data, it is always better to draw from electronics as a source than use some combination of all three other domains.

3.3.3 Part of speech tagging

While the literature on unsupervised part of speech tagging is quite large, to the best of our knowledge, we are the first to adapt part of speech taggers to new domains. Lease and Charniak [43] adapt a WSJ parser to biomedical text without any biomedical treebanked data. However, they assume other labeled resources in the target domain. In section 3.2.3 we give similar parsing results, but we adapt a source domain tagger to obtain the part of speech resources rather than using gold tags. Finally, McClosky and Charniak [47] use a self-training technique to adapt a natural language parser to a new domain. They don't apply their technique to biomedical text, but they do show significant gains for the Brown corpus. At the same time, for very different domains such as conversational speech, their self-training technique does not give a large improvement over their baseline parser.

3.4 Summary

The chapter described experiments demonstrating the use of structural correspondence learning for adapting sentiment classifiers and part of speech taggers. *For both tasks, SCL significantly improves a state-of-the-art discriminative model using on unlabeled data*. In the case of sentiment classfication, SCL gives a relative reduction in error due to adaptation of 36%. *When combined with simple methods for using labeled data [18, 30], for using labeled data SCL can give even greater improvement.* For both tasks, we demonstrated settings under which SCL can reduce error by more than 40%.

Chapter 4

Learning bounds for domain adaptation

An important component to better understanding domain adaptation is a formal characterization of when adaptation techniques work, as well as how to best exploit the resources we have. This chapter develops a theoretical framework for domain adaptation and comprises the work of Ben-David et al. [10] and Blitzer et al. [14]. We first show how to use this framework to prove bounds on the target error for classifiers which are trained in a source domain. We then demonstrate how to use the bound to estimate the adaptation error for the sentiment classification task. This is the focus of section 4.2 and comprises work from Blitzer et al. [15].

Section 4.3 gives a bound on the true target error of a model trained to minimize a convex combination of empirical source and target errors. The bound is relevant for scenarios where a limited amount of target data is available, such as those corresponding to the experiments of sections 3.1.4 and 3.2.4). It describes an intuitive tradeoff between the quantity of the source data and the accuracy of the target data. Furthermore, under relatively weak assumptions we can compute it from finite labeled and unlabeled samples of the source and target distributions. We use the task of sentiment classification to demonstrate that our bound makes correct predictions about model error with respect to the distance between source and target domains and the number of training instances.

Finally, we extend our theory to the case in which we have multiple sources of training data, each of which may be drawn according to a different distribution and may contain a different number of instances. Several authors have empirically studied a special case of this in which each *instance* is weighted separately in the loss function, and instance weights are set to approximate the target domain distribution [37, 13, 24, 39]. We give a uniform convergence bound for algorithms that minimize a convex combination of multiple empirical source errors and we show that these algorithms can outperform standard empirical error minimization.

4.1 A rigorous model of domain adaptation

We formalize domain adaptation for binary classification as follows. A *domain* is a pair consisting of a distribution $\mathcal{D}$ on $\mathcal{X}$ and a labeling function $f : \mathcal{X} \rightarrow [0, 1]$.[1] Initially we consider two domains, a *source* domain $\langle \mathcal{D}_S, f_S \rangle$ and a *target* domain $\langle \mathcal{D}_T, f_T \rangle$.

A *hypothesis* is a function $h : \mathcal{X} \rightarrow \{0, 1\}$. The probability according the distribution $\mathcal{D}_S$ that a hypothesis h disagrees with a labeling function f (which can also be a hypothesis) is defined as

$$\epsilon_S(h, f) \quad = \quad \mathrm{E}_{\mathbf{x} \sim \mathcal{D}_S} \left[\, |h(\mathbf{x}) - f(\mathbf{x})| \, \right] \, .$$

When we want to refer to the *error* of a hypothesis, we use the shorthand $\epsilon_S(h) = \epsilon_S(h, f_S)$. We write the empirical error of a hypothesis on the source domain as $\hat{\epsilon}_S(h)$. We use the parallel notation $\epsilon_T(h, f)$, $\epsilon_T(h)$, and $\hat{\epsilon}_T(h)$ for the target domain.

We measure the distance between two distributions $\mathcal{D}$ and $\mathcal{D}'$ using a hypothesis class-specific distance measure. Let $\mathcal{H}$ be a hypothesis class for instance space $\mathcal{X}$, and $\mathcal{A}_{\mathcal{H}}$ be the set of subsets of $\mathcal{X}$ that are the support of some hypothesis in $\mathcal{H}$. We define the distance between two distributions as:

$$d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{Z_h \in \mathcal{A}_{\mathcal{H}}} \left| \mathrm{Pr}_{\mathcal{D}} \left[Z_h \right] - \mathrm{Pr}_{\mathcal{D}'} \left[Z_h \right] \right| \, .$$

[1]This notion of domain is not the domain of a function. To avoid confusion, we will always mean a specific distribution and function pair when we say domain.

For our purposes, the distance $d_{\mathcal{H}}$ has an important advantage over other methods for comparing distributions such as L_1 distance or the KL divergence: we can compute $d_{\mathcal{H}}$ using finite samples from the distributions $\mathcal{D}$ and $\mathcal{D}'$ when $\mathcal{H}$ has finite VC dimension [12]. Furthermore, as the following theorem shows, we can compute a finite-sample approximation to $d_{\mathcal{H}}$ by finding a classifier $h \in \mathcal{H}$ that maximally discriminates between instances from $\mathcal{D}$ and $\mathcal{D}'$.

Theorem 2 *Let $\hat{\epsilon}.(h,1)$,$\hat{\epsilon}.(h,0)$ indicate the empirical error with respect to a particular distribution of hypothesis h with respect to the constant functions 1 and 0. For fixed samples $\mathcal{U}_S$,$\mathcal{U}_T$ from the source and target domains, both of size m', the empirical $d_{\mathcal{H}}$ distance is*

$$d_{\mathcal{H}}(\mathcal{U}_S,\mathcal{U}_T) = 2 - 2\min_{h\in\mathcal{H}}\left[\hat{\epsilon}_S(h,1) + \hat{\epsilon}_T(h,0)\right] \ .$$

The proof of this theorem is in appendix A.1. It relies on the one-to-one correspondence between hypotheses $h \in \mathcal{H}$ and halfspaces $Z_h \in \mathcal{A}_{\mathcal{H}}$.

We call the hypothesis that performs the best on the combined source and target distribution the ideal hypothesis:

$$h^* = \operatorname*{argmin}_{h\in\mathcal{H}} \epsilon_S(h) + \epsilon_T(h) \ .$$

We denote the combined error of h^* by $\lambda = \epsilon_S(h^*) + \epsilon_T(h^*)$. The ideal hypothesis explicitly embodies our notion of adaptability. When it performs poorly, we cannot expect to learn a good target classifier by minimizing source error. On the other hand, for the kinds of tasks mentioned in at the beginning of the chapter, we expect λ to be small. If this is the case, we can reasonably approximate target error using source error and the distance between $\mathcal{D}_S$ and $\mathcal{D}_T$.

The key element of our analysis is the error of one hypothesis with respect to another. We now define the symmetric difference space, which captures explicitly the halfspaces where two hypotheses disagree. We define the symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ as

$$\mathcal{H}\Delta\mathcal{H} = \{h(\mathbf{x}) \oplus h'(\mathbf{x}) : h, h' \in \mathcal{H}\} \ ,$$

where $\oplus$ is the XOR operator. Each hypothesis $g \in \mathcal{H}\Delta\mathcal{H}$ labels as positive all points x on which a given pair of hypotheses in $\mathcal{H}$ disagree.

We illustrate the kind of result available in this setting with the following bound on the target error in terms of the source error, the difference between labeling functions f_S and f_T, and the distance between the distributions $\mathcal{D}_S$ and $\mathcal{D}_T$. This bound is essentially a restatement of the main theorem of Ben-David et al. [10], correcting a mistake in both the statement and proof of their theorem.

Theorem 3 *Let $\mathcal{H}$ be a hypothesis space of VC-dimension d and $\mathcal{U}_S$, $\mathcal{U}_T$ be unlabeled samples of size m' each, drawn from $\mathcal{D}_S$ and $\mathcal{D}_T$, respectively. Let $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ be the empirical distance on $\mathcal{U}_S, \mathcal{U}_T$, induced by the symmetric difference hypothesis space. With probability at least $1 - \delta$ (over the choice of the samples), for every $h \in \mathcal{H}$,*

$$\epsilon_T(h) \le \epsilon_S(h) + \frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S, \mathcal{U}_T) + 4\sqrt{\frac{2d\log(2m') + \log(\frac{4}{\delta})}{m'}} + \lambda \ .$$

The corrected proof of this result can be found Appendix A.2. The main step in the proof is a variant of the triangle inequality in which the sides of the triangle represent errors of one decision rule with respect to another [10, 23]. The bound is relative to λ. When the combined error of the ideal hypothesis is large, there is no classifier that performs well on both the source and target domains, so we cannot hope to find a good target hypothesis by training only on the source domain. On the other hand, for small λ (the most relevant case for domain adaptation), theorem 3 shows that source error and unlabeled $\mathcal{H}\Delta\mathcal{H}$-distance are important quantities for computing target error.

4.2 Measuring adaptability with the $\mathcal{H}\Delta\mathcal{H}$ distance

Even with only unlabeled data, the $\mathcal{H}\Delta\mathcal{H}$-distance gives us a clue about how much adaptation loss we can expect for a particular pair of domains. To illustrate how this can be useful, we study a setting where an engineer knows roughly her domains of interest but does not have any labeled data yet. In that case, she can ask the question “Which sources

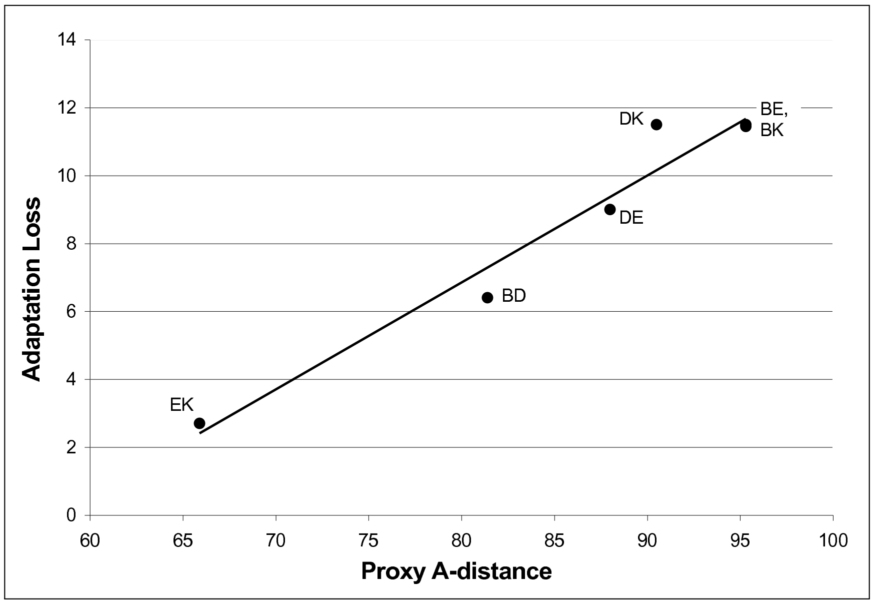


Figure 4.1: The proxy $\mathcal{H}\Delta\mathcal{H}$-distance between each domain pair plotted against the average adaptation loss. Each pair of domains is labeled by their first letters: EK indicates the pair electronics and kitchen.

should I label to obtain the best performance over all my domains?" On our product domains, for example, if we are interested in classifying reviews of kitchen appliances, we know from chapter 3 that it would be foolish to label reviews of books or DVDs rather than electronics. We show how to select source domains using only unlabeled data and the SCL representation.

We would like to use the $\mathcal{H}\Delta\mathcal{H}$-distance directly, but finding a maximally discriminating symmetric difference of linear classifiers is NP hard[2]. Instead, we approximate $d_{\mathcal{H}\Delta\mathcal{H}}$ by training a linear classifier to discriminate between the two domains. We use a standard hinge loss (normalized by dividing by the number of instances) and apply the quantity $1 - \big(\text{hinge loss}\big)$ in place of the actual $d_{\mathcal{H}\Delta\mathcal{H}}$. Let $\zeta(\mathcal{U}_S, \mathcal{U}_T)$ be our approximation to $d_{\mathcal{H}\Delta\mathcal{H}}$, computed from source and target unlabeled data. For domains that can be perfectly separated with margin, $\zeta(\mathcal{U}_S, \mathcal{U}_T) = 1$. For domains that are indistinguishable, $\zeta(\mathcal{U}_S, \mathcal{U}_T) = 0$.

[2]Even finding a maximally discriminating linear separator is NP-hard[12]

To decide which domains to label, for each pair of domains we compute the SCL representation. Then we create a data set where each instance $\mathbf{\Phi}^T\mathbf{x}$ is labeled with the identity of the domain from which it came and train a linear classifier. Figure 4.1 is a correlation plot between the proxy $\mathcal{H}\Delta\mathcal{H}$-distance and the adaptation error. The two are positively correlated.

Suppose we wanted to label two domains out of the four in such a way as to minimize our error on all the domains. Using the proxy $\mathcal{H}\Delta\mathcal{H}$-distance as a criterion, we observe that we would choose one domain from either books or DVDs, but not both, since then we would not be able to adequately cover electronics or kitchen appliances. Similarly we would also choose one domain from either electronics or kitchen appliances, but not both.

4.3 A learning bound combining source and target data

Theorem 3 shows how to relate source and target error. As sections 3.1.4 and 3.2.4 show though, we can often achieve significant improvement if we also have a small amount of labeled data in the target domain. We now proceed to give a learning bound for empirical error minimization using combined source and target training data. At train time a learner receives a sample $S = (S_T, S_S)$ of m instances, where S_T consists of βm instances drawn independently from $\mathcal{D}_T$ and S_S consists of $(1-\beta)m$ instances drawn independently from $\mathcal{D}_S$. The goal of a learner is to find a hypothesis that minimizes target error $\epsilon_T(h)$. When β is small, as in domain adaptation, minimizing empirical target error may not be the best choice. We analyze learners that instead minimize a convex combination of empirical source and target error:

$$\hat{\epsilon}_\alpha(h) = \alpha\hat{\epsilon}_T(h) + (1-\alpha)\hat{\epsilon}_S(h)$$

We denote as $\epsilon_\alpha(h)$ the corresponding weighted combination of true source and target errors, measured with respect to $\mathcal{D}_S$ and $\mathcal{D}_T$.

We bound the target error of a domain adaptation algorithm that minimizes $\hat{\epsilon}_\alpha(h)$. The proof of the bound has two main components, which we state as lemmas below. First we bound the difference between the target error $\epsilon_T(h)$ and weighted error $\epsilon_\alpha(h)$. Then we bound the difference between the true and empirical weighted errors $\epsilon_\alpha(h)$ and $\hat{\epsilon}_\alpha(h)$. The proofs of these lemmas, as well as the proof of Theorem 4, are in Appendix A.3.

Lemma 1 *Let h be a hypothesis in class $\mathcal{H}$. Then*

$$|\epsilon_\alpha(h) - \epsilon_T(h)| \leq (1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right) .$$

The lemma shows that as α approaches 1, we rely increasingly on the target data, and the distance between domains matters less and less. The proof uses a similar technique to that of Theorem 3.

Lemma 2 *Let $\mathcal{H}$ be a hypothesis space of VC-dimension d. If a random labeled sample of size m is generated by drawing βm points from $\mathcal{D}_T$ and $(1-\beta)m$ points from $\mathcal{D}_S$, labeling them according to f_S and f_T, respectively, then with probability at least $1-\delta$ (over the choice of the samples), for every $h \in \mathcal{H}$*

$$|\hat{\epsilon}_\alpha(h) - \epsilon_\alpha(h)| < \sqrt{\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}}\sqrt{\frac{d\log(2m) - \log\delta}{2m}} .$$

The proof is similar to standard uniform convergence proofs [64, 6], but it uses Hoeffding's inequality in a different way because the bound on the range of the random variables underlying the inequality varies with α and β. The lemma shows that as α moves away from β (where each instance is weighted equally), our finite sample approximation to $\epsilon_\alpha(h)$ becomes less reliable.

Theorem 4 *Let $\mathcal{H}$ be a hypothesis space of VC-dimension d. Let $\mathcal{U}_S$ and $\mathcal{U}_T$ be unlabeled samples of size m' each, drawn from $\mathcal{D}_S$ and $\mathcal{D}_T$ respectively. Let S be a labeled sample of size m generated by drawing βm points from $\mathcal{D}_T$ and $(1-\beta)m$ points from $\mathcal{D}_S$, labeling them according to f_S and f_T, respectively. If $\hat{h} \in \mathcal{H}$ is the empirical minimizer of $\hat{\epsilon}_\alpha(h)$ on S and $h_T^* = \min_{h\in\mathcal{H}} \epsilon_T(h)$ is the target error minimizer, then with probability at least*

$1-\delta$ *(over the choice of the samples),*

$$\epsilon_T(\hat{h}) \le \epsilon_T(h_T^*) + 2\sqrt{\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}}\sqrt{\frac{d\log(2m) - \log\delta}{2m}} + \\ 2(1-\alpha)\left(\frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S,\mathcal{U}_T) + 4\sqrt{\frac{2d\log(2m') + \log(\frac{4}{\delta})}{m'}} + \lambda\right) .$$

When $\alpha = 0$ (that is, we ignore target data), the bound is identical to that of Theorem 3, but with an empirical estimate for the source error. Similarly when $\alpha = 1$ (that is, we use only target data), the bound is the standard learning bound using only target data. At the optimal α (which minimizes the right hand side), the bound is always at least as tight as either of these two settings. Finally note that by choosing different values of α, the bound allows us to effectively trade off the small amount of target data against the large amount of less relevant source data.

4.4 Evaluating the bound from theorem 4

We evaluate our theory by comparing its predictions to empirical results. While ideally theorem 4 could be directly compared with test error, this is not practical because λ is unknown, $d_{\mathcal{H}\Delta\mathcal{H}}$ is computationally intractable [10], and the VC dimension d is too large to be a useful measure of complexity. Instead, we develop a simple approximation of theorem 4 that we can compute from unlabeled data. For many adaptation tasks, λ is small (there exists a classifier which is simultaneously good for both domains), so we ignore it here. We approximate $d_{\mathcal{H}\Delta\mathcal{H}}$ using the technique of section 4.2. Finally we replace the VC dimension sample complexity term with a tighter constant C. The resulting approximation to the bound of Theorem 4 is

$$f(\alpha) = \sqrt{\frac{C}{m}\left(\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}\right)} + (1-\alpha)\zeta(\mathcal{U}_S,\mathcal{U}_T) . \tag{4.1}$$

Our experimental results are for the task of sentiment classification. We use the data provided described in chapter 3 and augment it with four additional domains. This gives us

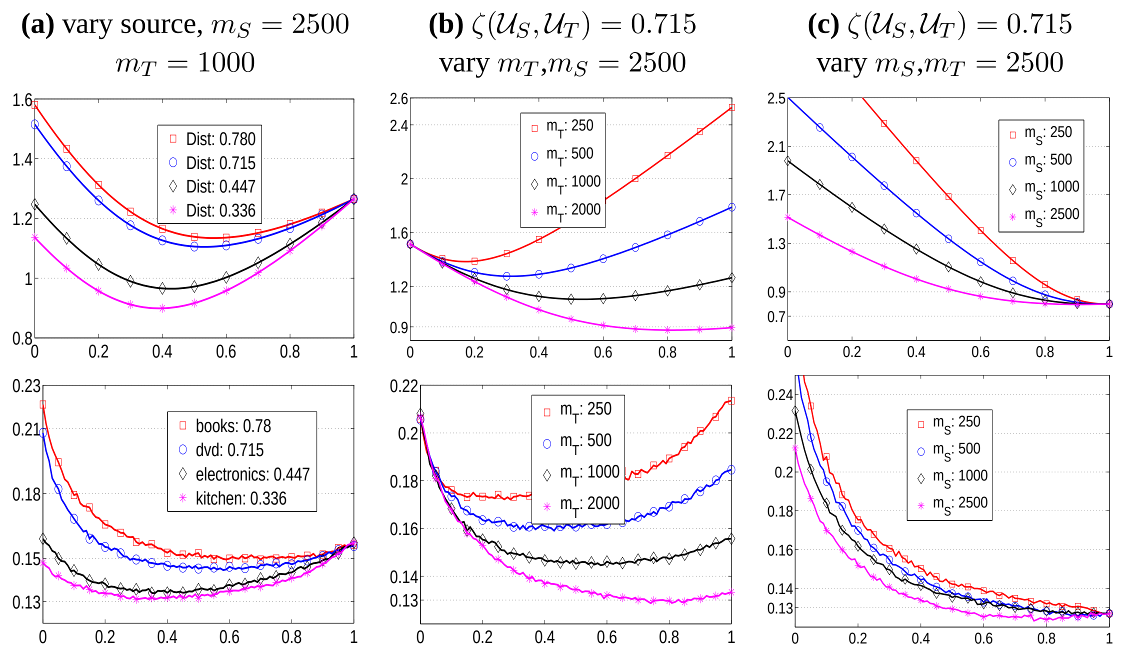


Figure 4.2: Comparing the bound from theorem 4 with test error for sentiment classification. Each column varies one component of the bound. For all plots, the y-axis shows the error and the x-axis shows α. Plots on the top row show the value given by the bound, and plots on the bottom row show the empirical test set error. Column (a) depicts different distances among domains. Column (b) depicts different numbers of target instances, and column (c) represents different numbers of source instances.

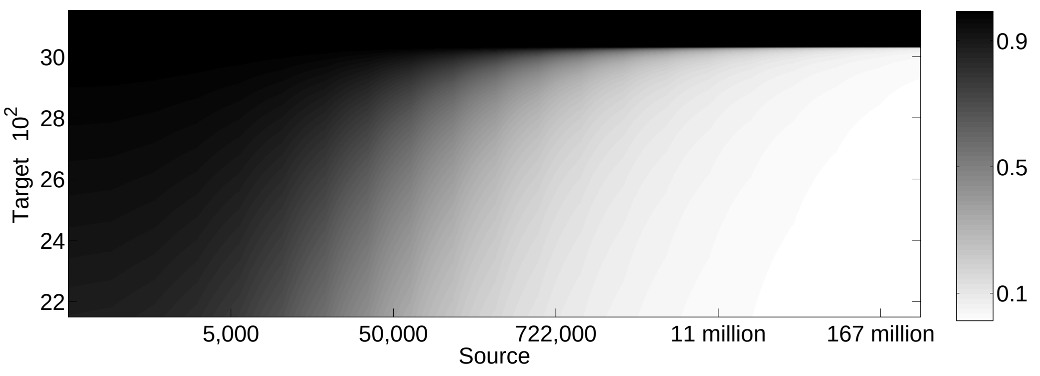


Figure 4.3: An illustration of the phase transition between preferring either source or target training data. The value of α which minimizes the bound is indicated by the intensity, where black means $\alpha = 1$. We fix $C_1 = 1600$ and $\zeta(\mathcal{U}_S, \mathcal{U}_T) = 0.715$, as in figure 4.2. The x-axis shows the number of source instances (log-scale). The y-axis shows the number of target instances. A phase transition occurs at 3,130 target instances. With more target instances than this, it is more effective to ignore even an infinite amount of source data.

a total of eight types of products: apparel, books, DVDs, electronics, kitchen appliances, music, video, and a catchall category "other". As before, the task is binary classification: given a review, predict whether it is positive (4 or 5 out of 5 stars) or negative (1 or 2 stars). We chose the "apparel" domain as our target domain, and all of the plots on the bottom row of figure 4.2 are for this domain. We obtain empirical curves for the error as a function of α by training a classifier using a weighted hinge loss. Suppose the target domain has weight α and there are βm target training instances. Then we scale the loss of target training instance by $\frac{\alpha}{\beta}$ and the loss of a source training instance by $\frac{1-\alpha}{1-\beta}$.

Figure 4.2 shows a series of plots of equation 4.1 (top row) coupled with corresponding plots of test error (bottom row) as a function of α for different amounts of source and target data and different distances between domains. In each column, a single parameter (distance, number of target instances m_T, or number of source instances m_S) is varied while the other two are held constant. Note that $\beta = \frac{m_T}{m_T + m_S}$. The plots on the top row of figure 4.2 are not meant to be numerical proxies for the true error (For the source domains "books" and "dvd", the distance alone is well above $\frac{1}{2}$). Instead, they are scaled to illustrate that the bound is similar in shape to the true error curve and that relative relatinships are preserved. Note that by choosing a different C in equation 4.1 for each curve, one can achieve complete control over their minima. In order to avoid this, we only use a single value of $C = 1600$ for all 12 curves on the top side of Figure 4.2.

First note that in every pair of plots, the empirical error curves, like the bounds, have a roughly convex shape. Furthermore the value of α which minimizes the bound also has low empirical error for each corresponding curve. This suggests that choosing α to minimize the bound of Theorem 4 and subsequently training a classifier to minimize the empirical error $\hat{\epsilon}_\alpha(h)$ can work well in practice, provided we have a reasonable measure of complexity. Column (a) shows that more distant source domains result in higher target error. Column (b) illustrates that for more target data, we have not only lower error in general, but also a higher minimizing α.

Finally, column (c) depicts the limitation of distant source data. With enough target data, no matter how much source data we include, we always prefer to use only the target data. Intuitively this is because for any source domain with non-zero distance from the target, we cannot achieve zero error relative to the best target hypothesis. This is reflected in our bound as a phase transition in the optimal value of α (Figure 4.3). As we increase the number of target instances, once the number crosses the threshold $m_T = \frac{C}{\zeta(\mathcal{U}_S, \mathcal{U}_T)^2}$, using source data can only add noise, and thus we always prefer to use only target data.

4.5 Learning from multiple sources

We now explore an extension of our theory to the case of multiple source domains. We are presented with data from N distinct sources. Each source S_j is associated with an unknown underlying distribution $\mathcal{D}_j$ over input points and an unknown labeling function f_j. From each source S_j, we are given m_j labeled training instances, and our goal is to use these instances to train a model to perform well on a target domain $\langle \mathcal{D}_T, f_T \rangle$, which may or may not be one of the sources. This setting is motivated by several domain adaptation algorithms [37, 13, 39, 24] that weigh the loss from training instances depending on how "far" they are from the target domain. That is, each training instance is its own source domain.

As in the previous sections, we will examine algorithms that minimize convex combinations of training errors over the labeled examples from each source domain. As before, we let $m_j = \beta_j m$ with $\sum_{j=1}^{N} \beta_j = 1$. Given a vector $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_N)$ of domain weights with $\sum_j \alpha_j = 1$, we define the empirical $\boldsymbol{\alpha}$-weighted error of function h as

$$\hat{\epsilon}_{\boldsymbol{\alpha}}(h) = \sum_{j=1}^{N} \alpha_j \hat{\epsilon}_j(h) = \sum_{j=1}^{N} \frac{\alpha_j}{m_j} \sum_{x \in S_j} |h(x) - f_j(x)| \, .$$

The true $\boldsymbol{\alpha}$-weighted error $\epsilon_{\boldsymbol{\alpha}}(h)$ is defined in the analogous way. Let $\mathcal{D}_{\boldsymbol{\alpha}}$ be a mixture of the N source distributions with mixing weights equal to the components of $\boldsymbol{\alpha}$. Finally, analogous to λ in the single-source setting, we define the error of the multi-source ideal hypothesis to be

$$\gamma = \min_h \{\epsilon_T(h) + \epsilon_{\boldsymbol{\alpha}}(h)\} = \min_h \{\epsilon_T(h) + \sum_{j=1}^{N} \alpha_j \epsilon_j(h)\} \, .$$

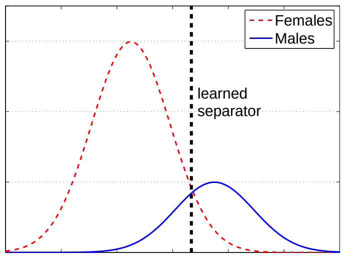
The following theorem gives a learning bound for empirical error minimization using the empirical $\boldsymbol{\alpha}$-weighted error.

Theorem 5 *Suppose we are given m_j labeled instances from source S_j for $j = 1 \ldots N$. For a fixed vector of weights $\boldsymbol{\alpha}$, let $\hat{h} = \operatorname{argmin}_{h \in \mathcal{H}} \hat{\epsilon}_{\boldsymbol{\alpha}}(h)$, and let $h_T^* = \operatorname{argmin}_{h \in \mathcal{H}} \epsilon_T(h)$. Then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ (over the choice of samples from each source),*

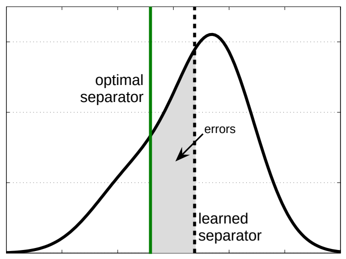
$$\epsilon_T(\hat{h}) \leq \epsilon_T(h_T^*) + 2\sqrt{\sum_{j=1}^{N} \frac{\alpha_j^2}{\beta_j}} \sqrt{\frac{d \log 2m - \log \delta}{2m}} + 2\left(\gamma + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(D_{\boldsymbol{\alpha}}, D_T)\right) \, .$$

The full proof is in appendix A.4. Like the proof of Theorem 4, it is split into two parts. The first part bounds the difference between the $\boldsymbol{\alpha}$-weighted error and the target error

(a) Source. More boys than girls



(b) Target. Separator from uniform mixture is suboptimal



(c) Weighting sources to match target is optimal

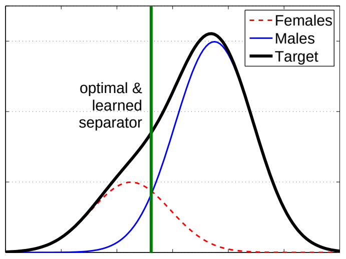


Figure 4.4: A 1-dimensional example illustrating how non-uniform mixture weighting can result in minimal error. We observe one feature, which we use to predict gender. **(a)** At train time we observe more females than males. **(b)** Learning by uniformly weighting the training data causes us to learn a suboptimal decision boundary, **(c)** but by weighting the males more highly, we can match the target data and learn an optimal classifier.

similar to lemma 1. The second is a uniform convergence bound for $\hat{\epsilon}_{\boldsymbol{\alpha}}(h)$ similar to lemma 2.

Theorem 5 reduces to Theorem 4 when we have only two sources, one of which is the target domain (that is, we have some small number of target instances). It is more general, though, because by manipulating $\boldsymbol{\alpha}$ we can effectively change the source domain. At the same time, we must pay for this generality by strengthening our assumptions. Now we demand that for every $\boldsymbol{\alpha}$-weighted convex combination of sources, there exists a hypothesis h^* which has low error on both the combination of sources and the target domain. Second, we measure distance between the target and a mixture of sources, rather than between the target and a single source.

One question we might ask is whether there exist settings where a non-uniform weighting can lead to a significantly lower value of the bound than a uniform weighting. This can happen if some non-uniform weighting of sources accurately approximates the target domain. As a hypothetical example, suppose we are trying to predict gender from height (Figure 4.4). Each instance is drawn from a gender-specific Gaussian. In this example, we can find the optimal classifier by weighting the "males" and "females" components of the source to match the target.

4.6 Related work

Domain adaptation is a widely-studied area, and we cannot hope to cover every aspect and application of it here. Instead, in this section we focus on other theoretical approaches to domain adaptation. While we do not explicitly address the relationship in this thesis, we note that domain adaptation is closely related to the setting of covariate shift, which has been studied in statistics. The covariate shift setting is one where $\Pr_S[\mathbf{x}] \neq \Pr_T[\mathbf{x}]$, but $\Pr_S[y|\mathbf{x}] = \Pr_T[y|\mathbf{x}]$. In addition to the work of Huang et al. [37], several other authors have considered learning by assigning separate weights to the components of the loss function corresponding to separate instances. Bickel at al. [13] and Jiang and Zhai [39] suggest promising empirical algorithms that in part inspire our Theorem 5. We hope that our work can help to explain when these algorithms are effective. Dai et al. [24] considered weighting instances using a transfer-aware variant of boosting, but the learning bounds they give are no stronger than bounds which completely ignore the source data.

Crammer et al. [23] consider learning when the marginal distribution on instances is the same across sources but the labeling function may change. This corresponds in our theory to cases where $d_{\mathcal{H}\Delta\mathcal{H}} = 0$ but λ is large. Like us they consider multiple sources, but their notion of weighting is less general. They consider only including or discarding a source entirely.

4.7 Summary

This chapter described a theoretical framework for domain adaptation. *A key part of this framework is the $\mathcal{H}\Delta\mathcal{H}$-distance, a measure of divergence between distributions that is directly related to classification error.* We can use the $\mathcal{H}\Delta\mathcal{H}$-distance to estimate the relative loss due to adaptation for different pairs of domains from only *unlabeled* data.

The main theoretical result of this chapter is theorem 4, a learning bound for a procedure which minimizes a convex combination of empirical source and target errors. This bound can be used to decide on an effective tradeoff between source and target training data.

Chapter 5

Conclusion

Adapting statistical models to new domains is a crucial part of applying text processing systems in the real world. Domain adaptation addresses the situation in which we possess a large amount of labeled data from a source domain to train a model but little or no labeled data from a target domain where we wish to apply the model. To the best of our knowledge, this thesis represents the first attempt to address domain adaptation for text by learning representations which minimize the divergence between source and target domains.

Linear discriminative models for text achieve state-of-the-art results by creating features based on vocabulary items. Algorithms for estimating linear models assume that the training and testing data are drawn from the same distribution, but for domain adaptation, this is not true. Differences in vocabulary create different distributions over features, and this difference in the feature space leads empirically to significant increases in error. The first part of this thesis, in chapters two and three, introduced an algorithm for domain adaptation called structural correspondence learning (SCL) and examined its performance on two text processing tasks. The second part, in chapter four, gave a formal definition for domain adaptation and proved generalization bounds for the setting when training data and testing data are drawn from different distributions.

SCL is a variant of the structural learning paradigm of Ando and Zhang [3], a semisupervised method which uses unlabeled data to discover a predictive subspace. SCL uses the techniques of structural learning to discover a subspace which is simultaneously predictive for both source and target domains. The key concept behind SCL is the selection of pivot features which link the two domains. With these pivot features in hand, we learn a representation by finding a linear projection Φ from our original feature space onto a low-dimensional subspace that is most predictive of the presence of pivots in a particular instance. This subspace implicitly aligns features from different domains because if two non-pivot features are both predictive for the same set of pivots, then these features are mapped to the same area of the low-dimensional subspace. We can then train models using the projection of an instance onto this feature subspace, with the intention that they will generalize better to the target domain. In chapter three, we demonstrated the effectiveness of SCL on models for sentiment classification and part of speech tagging. We showed that with only unlabeled target data, SCL can significantly improve the performance of a state-of-the-art linear model, and we explored combining SCL with methods for incorporating both source and target labeled data. In situations when we have a small amount of target data, SCL can make an even larger improvement.

In chapter 4, we developed a formal framework for analyzing differing source and target domains. Standard generalization theory bounds the difference in training and test performance when training and test sets are samples from the same distribution. Our theory yields bounds on the difference in performance that are based on the divergence between the training and test distributions. While the divergence between arbitrary distributions is not measureable in general, we showed how a simple assumption can allow us to measure the divergence using only unlabeled data. If we assume that there exists a single predictor which is effective in both domains, then we can represent the divergence using the hypothesis class from which our predictors are drawn. We call this divergence the $\mathcal{H}\Delta\mathcal{H}$-divergence, and it allows us to prove a generalization bound which we can compute from finite samples of unlabeled data.

The representation learned by SCL significantly decreases the $\mathcal{H}\Delta\mathcal{H}$-distance between the two domains, and as such significantly decreases the value of the generalization bound. We can also use the $\mathcal{H}\Delta\mathcal{H}$-divergence to prove a bound for the setting in which we have available both source and target labeled data. In this setting, we examined algorithms which minimize convex combinations of source and target error. The resulting generalization bound intuitively captures the tradeoff between using the large but biased source training data and using the small but unbiased target training data. We showed for the sentiment classification dataset that our bound makes accurate predictions about the relative error of different amounts of source and target training data and different divergence among domains.

We examined both the practical and theoretical sides of domain adaptation. The theory we developed addresses representation abstractly, though, and it doesn't give any insights into when and how SCL can perform well. In chapter two, we linked SCL to the statistical method of canonical correlation analysis (CCA). Kakade and Foster [40] showed how CCA can be useful in a multi-view semisupervised learning setting. They proposed first learning a representation using CCA on unlabeled data. Then they used that representation when estimating the parameters of a supervised linear model. They showed that the resulting CCA-based representation results in good predictors under a simple assumption: The optimal classifier from each view alone must have low regret with respect to the joint optimal classifier from both views.

Unfortunately, this assumption is too strong to permit an analysis of SCL analogous to their analysis of CCA. But it does seem natural to relax the assumptions of Kakade and Foster to incorporate an intermediate representation: one in which one view is sufficient for learning an optimal classifier for nearly every instance, but where we don't know a priori which view it is. Such an analysis would provide direct theoretical justification for SCL, and we believe it will ultimately lead to new, simpler algorithms for domain adaptation.

Appendix A

Appendix

A.1 Proof of theorem 2

Let $Z_h \in \mathcal{A}_\mathcal{H}$ denote the halfspace associated with hypothesis h, and let Z_h^C be the complement of this halfspace. We need to show that

$$\begin{aligned} 2 - 2\min_{h \in \mathcal{H}} \left[\hat{\epsilon}_S(h,1) + \hat{\epsilon}_T(h,0)\right] &= \max_{h \in \mathcal{H}} \left|\Pr_{\mathcal{U}_S}\left[Z_h\right] - \Pr_{\mathcal{U}_T}\left[Z_h\right]\right| \\ \max_{h \in \mathcal{H}} \left[2 - 2\epsilon_S(h,1) + \epsilon_T(h,0)\right] &= \max_{h \in \mathcal{H}} \left|\Pr_{\mathcal{U}_S}\left[Z_h\right] - \Pr_{\mathcal{U}_T}\left[Z_h\right]\right| \end{aligned}$$

It suffices to show that for every $h \in \mathcal{H}$

$$2 - 2\left[\epsilon_S(h,1) + \epsilon_T(h,0)\right] = \left|\Pr_{\mathcal{U}_S}\left[Z_h\right] - \Pr_{\mathcal{U}_T}\left[Z_h\right]\right| \ .$$

Let $I\left[\mathbf{x} \in \mathcal{U}_S\right]$ be the indicator function which is 1 when the vector $\mathbf{x}$ is a member of the sample. Below, when we write $\sum_\mathbf{x}$, this indicates a summation over only those $\mathbf{x}$ in our joint sample $\mathcal{U}_S \bigcup \mathcal{U}_T$.

$$2 - 2\left[\epsilon_S(h,1) + \epsilon_T(h,0)\right] = 2 - \frac{2}{m}\left(\sum_{\mathbf{x},h(\mathbf{x})=0} I\left[\mathbf{x} \in \mathcal{U}_T\right] + \sum_{\mathbf{x},h(\mathbf{x})=1} I\left[\mathbf{x} \in \mathcal{U}_S\right]\right)$$

$$
\begin{aligned}
&= \frac{1}{m}\sum_{\mathbf{x},h(\mathbf{x})=0}\left(I\left[\mathbf{x}\in\mathcal{U}_T\right]+I\left[\mathbf{x}\in\mathcal{U}_S\right]\right)+\frac{1}{m}\sum_{\mathbf{x},h(\mathbf{x})=1}\left(I\left[\mathbf{x}\in\mathcal{U}_T\right]+I\left[\mathbf{x}\in\mathcal{U}_S\right]\right)\\
&\quad -\frac{2}{m}\left(\sum_{\mathbf{x},h(\mathbf{x})=0} I\left[\mathbf{x}\in\mathcal{U}_T\right]+\sum_{\mathbf{x},h(\mathbf{x})=1} I\left[\mathbf{x}\in\mathcal{U}_S\right]\right)\\
&= \frac{1}{m}\sum_{\mathbf{x},h(\mathbf{x})=0}\left(I\left[\mathbf{x}\in\mathcal{U}_S\right]-I\left[\mathbf{x}\in\mathcal{U}_T\right]\right)+\frac{1}{m}\sum_{\mathbf{x},h(\mathbf{x})=1}\left(I\left[\mathbf{x}\in\mathcal{U}_T\right]-I\left[\mathbf{x}\in\mathcal{U}_S\right]\right)\\
&= \frac{1}{2}\left|\Pr_{\mathcal{U}_S}\left[Z_h\right]-\Pr_{\mathcal{U}_T}\left[Z_h\right]\right|+\frac{1}{2}\left|\Pr_{\mathcal{U}_S}\left[Z_h^C\right]-\Pr_{\mathcal{U}_T}\left[Z_h^C\right]\right|\\
&= \left|\Pr_{\mathcal{U}_S}\left[Z_h\right]-\Pr_{\mathcal{U}_T}\left[Z_h\right]\right|
\end{aligned}
$$

■

The last step follows from the identity $\Pr_{\mathcal{U}}\left[Z_h^C\right]=1-\Pr_{\mathcal{U}}\left[Z_h\right]$.

A.2 Proof of theorem 3

Below we use $\triangle$**ineq** to indicate that a line of the proof follows by application of the triangle inequality [10, 23].

$$
\begin{aligned}
\epsilon_T(h) &\leq \epsilon_T(h^*)+\epsilon_T(h,h^*) \qquad \triangle\textbf{ineq}\\
&\leq \epsilon_T(h^*)+\epsilon_S(h,h^*)+\left|\epsilon_T(h,h^*)-\epsilon_S(h,h^*)\right| \qquad \triangle\textbf{ineq}\\
&\leq \epsilon_T(h^*)+\epsilon_S(h,h^*)+\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T)\\
&\leq \epsilon_T(h^*)+\epsilon_S(h)+\epsilon_S(h^*)+\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) \qquad \triangle\textbf{ineq}\\
&= \epsilon_S(h)+\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T)+\lambda \qquad \triangle\textbf{ineq}\\
&\leq \epsilon_S(h)+\frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S,\mathcal{U}_T)+4\sqrt{\frac{2d\log(2m')+\log(\frac{4}{\delta})}{m'}}+\lambda
\end{aligned}
$$

The last step in the proof is an application of theorem 3.4 from [12], together with the observation that since we can represent every $g\in\mathcal{H}\Delta\mathcal{H}$ as a linear threshold network of depth 2 with 2 hidden units, the VC dimension of $\mathcal{H}\Delta\mathcal{H}$ is at most twice the VC dimension of $\mathcal{H}$ [6]. ■

A.3 Proof of main theorem

A.3.1 Proof of lemma 1

$$
\begin{aligned}
|\epsilon_\alpha(h) - \epsilon_T(h)| &= (1-\alpha)|\epsilon_S(h) - \epsilon_T(h)| \\
&\le (1-\alpha)\left[|\epsilon_S(h) - \epsilon_S(h,h^*)| + |\epsilon_S(h,h^*) - \epsilon_T(h,h^*)| + |\epsilon_T(h,h^*) - \epsilon_T(h)|\right] \\
&\le (1-\alpha)\left[\epsilon_S(h^*) + |\epsilon_S(h,h^*) - \epsilon_T(h,h^*)| + \epsilon_T(h^*)\right] \qquad \triangle\textbf{ineq} \\
&\le (1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) + \lambda\right)
\end{aligned}
$$

■

A.3.2 Proof of lemma 2

We begin by restating Hoeffding's inequality.

Hoeffding's inequality

If $X_1, X_2, \ldots, X_n$ are independent and $a_i \le X_i \le b_i (i = 1, 2, \ldots, n)$, then for $\epsilon > 0$

$$
\Pr\left[|\bar{X} - E[\bar{X}]| \ge \epsilon\right] \le 2e^{-2n^2\epsilon^2/\sum_{i=1}^{n}(b_i-a_i)^2},
$$

where $\bar{X} = (X_1 + \cdots + X_n)/n$.

Let $X_1, \ldots, X_{\beta m}$ be random variables that take on the values $\frac{\alpha}{\beta}|h(x) - f_T(x)|$ for the βm instances $x \in S_T$. Similarly, let $X_{\beta m+1}, \ldots, X_m$ be random variables that take on the values $\frac{1-\alpha}{1-\beta}|h(x) - f_S(x)|$ for the $(1-\beta)m$ instances $x \in S_S$. Note that $X_1, \ldots, X_{\beta m} \in [0, \frac{\alpha}{\beta}]$ and $X_{\beta m+1}, \ldots, X_m \in [0, \frac{1-\alpha}{1-\beta}]$. Then

$$
\begin{aligned}
\hat{\epsilon}_\alpha(h) &= \alpha\hat{\epsilon}_T(h) + (1-\alpha)\hat{\epsilon}_S(h) \\
&= \alpha\frac{1}{\beta m}\sum_{x\in S_T}|h(x) - f_T(x)| + (1-\alpha)\frac{1}{(1-\beta)m}\sum_{x\in S_S}|h(x) - f_S(x)| \\
&= \frac{1}{m}\sum_{i=1}^{m} X_i.
\end{aligned}
$$

Furthermore, by linearity of expectations

$$
\begin{aligned}
E[\hat{\epsilon}_\alpha(h)] &= \frac{1}{m}\left(\beta m \frac{\alpha}{\beta}\epsilon_T(h) + (1-\beta)m\frac{1-\alpha}{1-\beta}\epsilon_S(h))\right) \\
&= \alpha\epsilon_T(h) + (1-\alpha)\epsilon_S(h) = \epsilon_\alpha(h).
\end{aligned}
$$

So by Hoeffding's inequality the following holds for every h.

$$
\begin{aligned}
\Pr\left[|\hat{\epsilon}_\alpha(h) - \epsilon_\alpha(h)| \geq \epsilon\right] &\leq 2\exp\left(\frac{-2m^2\epsilon^2}{\sum_{i=1}^m \text{range}^2(X_i)}\right) \\
&= 2\exp\left(\frac{-2m^2\epsilon^2}{\beta m\left(\frac{\alpha}{\beta}\right)^2 + (1-\beta)m\left(\frac{1-\alpha}{1-\beta}\right)^2}\right) \\
&= 2\exp\left(\frac{-2m\epsilon^2}{\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}}\right).
\end{aligned}
$$

The remainder of the proof for hypothesis classes of finite VC dimension follows a standard argument. In particular, the reduction to a finite hypothesis class using the growth function does not change [64, 6]. This, combined with the union bound gives us the probability that there exists *any* hypothesis $h \in \mathcal{H}$, $|\hat{\epsilon}_\alpha(h) - \epsilon_\alpha(h)| \geq \epsilon$. Substituting δ for the probability and solving gives the lemma

$$
\epsilon = \sqrt{\left(\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}\right)\frac{d\log(2m) - \log\delta}{2m}}
$$

■

A.3.3 Proof of theorem 4

The proof follows the standard set of steps for proving learning bounds [6], using Lemma 1 to bound the difference between target and weighted errors and Lemma 2 for the uniform convergence of empirical and true weighted errors. Below we use L1, L2, and Thm3 to indicate that a line of the proof follows by application of Lemma 1, Lemma 2, or

Theorem 3 respectively.

$$\epsilon_T(\hat{h}) \leq \epsilon_\alpha(\hat{h}) + (1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T)+\lambda\right) \text{ (L1)}$$

$$\leq \hat{\epsilon}_\alpha(\hat{h}) + \sqrt{\left(\frac{\alpha^2}{\beta}+\frac{(1-\alpha)^2}{1-\beta}\right)\frac{d\log(2m)-\log\delta}{2m}}+(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T)+\lambda\right) \text{ (L2)}$$

$$\leq \hat{\epsilon}_\alpha(h_T^*) + \sqrt{\left(\frac{\alpha^2}{\beta}+\frac{(1-\alpha)^2}{1-\beta}\right)\frac{d\log(2m)-\log\delta}{2m}}+(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T)+\lambda\right)$$

$$\leq \epsilon_\alpha(h_T^*) + 2\sqrt{\left(\frac{\alpha^2}{\beta}+\frac{(1-\alpha)^2}{1-\beta}\right)\frac{d\log(2m)-\log\delta}{2m}}+(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T)+\lambda\right) \text{ (L2)}$$

$$\leq \epsilon_T(h_T^*) + 2\sqrt{\left(\frac{\alpha^2}{\beta}+\frac{(1-\alpha)^2}{1-\beta}\right)\frac{d\log(2m)-\log\delta}{2m}}+2(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T)+\lambda\right) \text{(L1)}$$

$$\leq \epsilon_T(h_T^*) + 2\sqrt{\left(\frac{\alpha^2}{\beta}+\frac{(1-\alpha)^2}{1-\beta}\right)\frac{d\log(2m)-\log\delta}{2m}} +$$
$$2(1-\alpha)\left(\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S,\mathcal{U}_T)+4\sqrt{\frac{2d\log(2m')+\log(\frac{4}{\delta})}{m'}}+\lambda\right) \text{ (Thm 3)}$$

■

A.4 Proof of theorem 5

Lemma 3 *Let h be a hypothesis in class $\mathcal{H}$. Then $|\epsilon_{\boldsymbol{\alpha}}(h)-\epsilon_T(h)| \leq d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\boldsymbol{\alpha}},\mathcal{D}_T)+\gamma$,*

Proof:

$$|\epsilon_{\boldsymbol{\alpha}}(h)-\epsilon_T(h)| \leq [|\epsilon_{\boldsymbol{\alpha}}(h)-\epsilon_{\boldsymbol{\alpha}}(h,h^*)| + |\epsilon_{\boldsymbol{\alpha}}(h,h^*)-\epsilon_T(h,h^*)| +$$
$$|\epsilon_T(h,h^*)-\epsilon_T(h)|] \qquad \triangle\textbf{ineq}$$

$$\leq [\epsilon_{\boldsymbol{\alpha}}(h^*) + |\epsilon_{\boldsymbol{\alpha}}(h,h^*)-\epsilon_T(h,h^*)| + \epsilon_T(h^*)] \qquad \triangle\textbf{ineq}$$
$$\leq (\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\boldsymbol{\alpha}},\mathcal{D}_T)+\gamma)$$

■

Lemma 4 *Let $\mathcal{H}$ be a hypothesis space of VC-dimension d. If a random labeled sample of size m is generated by drawing $\beta_j m$ points from $\mathcal{D}_j$, and labeling them according to f_j, then with probability at least $1-\delta$ (over the choice of the samples), for every $h \in \mathcal{H}$:*

$$|\hat{\epsilon}_{\boldsymbol{\alpha}}(h) - \epsilon_{\boldsymbol{\alpha}}(h)| < \sqrt{\sum_j \frac{\alpha_j^2}{\beta_j}} \sqrt{\frac{d \log(2m) - \log \delta}{2m}}$$

Proof: Because of its similarity to the proof of Lemma 2 (in Appendix A.3.2), we will omit some details of this proof. Let $X_1, \ldots, X_{\beta_j m}$ be random variables that take on the values $\frac{\alpha_j}{\beta_j}|h(x) - f_j(x)|$ for the $\beta_j m$ instances $x \in S_j$. Note that $X_1, \ldots, X_{\beta_j m} \in [0, \frac{\alpha_j}{\beta_j}]$. Then

$$\hat{\epsilon}_{\boldsymbol{\alpha}}(h) = \sum_{j=1}^{N} \alpha_j \hat{\epsilon}_j(h) = \sum_{j=1}^{N} \alpha_j \frac{1}{\beta_j m} \sum_{x \in S_j} |h(x) - f_j(x)| = \frac{1}{m} \sum_{i=1}^{m} X_i.$$

By linearity of expectations again, we have $E[\hat{\epsilon}_{\boldsymbol{\alpha}}(h)] = \epsilon_{\boldsymbol{\alpha}}(h)$.

By Hoeffding's inequality the following holds for every h.

$$\begin{aligned}\Pr\left[|\hat{\epsilon}_{\boldsymbol{\alpha}}(h) - \epsilon_{\boldsymbol{\alpha}}(h)| \geq \epsilon\right] &\leq 2\exp\left(\frac{-2m^2\epsilon^2}{\sum_{i=1}^{m} \mathrm{range}^2(X_i)}\right) \\ &= 2\exp\left(\frac{-2m\epsilon^2}{\sum_j \frac{\alpha_j^2}{\beta_j}}\right).\end{aligned}$$

The remainder of the proof is identical to the proof of lemma 2. ■

The proof of theorem 5 uses lemmas 3 and 4 and follows an identical argument to the proof of Theorem 4.

Bibliography

[1] Steven Abney. Bootstrapping. In *40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[2] Steven Abney. Understanding the yarowsky algorithm. *Computational Linguistics*, 7, 2004.

[3] Rie Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

[4] Rie Kubota Ando. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the 10th Conference on Natural Language Learning*, 2006.

[5] Rie Kubota Ando, Mark Dredze, and Tong Zhang. Trec 2005 genomics track experiments at ibm watson. In *Proceedings of the Fourteenth Text Retrieval Conference*, 2005.

[6] Martin Anthony and Peter Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[7] Anthony Aue and Michael Gamon. Customizing sentiment classifiers to new domains: a case study. http://research.microsoft.com/ anthaue/, 2005.

[8] Peter Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[9] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: a geometric framework for lerning from lableed and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[10] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Neural Information Processing Systems NIPS*, 2007.

[11] Shai Ben-David, Nadav Eiron, and Phil Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66:496–514, 2003.

[12] Shai Ben-David, Johannes Gehrke, and Daniel Kifer. Detecting change in data streams. In *Very Large Databases (VLDB)*, 2004.

[13] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *ICML*, 2007.

[14] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman. Learning bounds for domain adaptation. In *Neural Information Processing Systems NIPS*, 2008.

[15] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes, and blenders. domain adaptation for sentiment classification. In *Association for Computational Linguistics ACL*, 2007.

[16] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing EMNLP*, 2006.

[17] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Workshop on Computational Learning Theory*, 1998.

[18] Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *EMNLP*, 2004.

[19] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing EMNLP*, 2002.

[20] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing EMNLP*, 2002.

[21] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Empirical Methods in Natural Language Processing*, 1999.

[22] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006.

[23] Koby Crammer, Michael Kearns, and Jenn Wortman. Learning from multiple sources. In *NIPS*, 2007.

[24] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *ICML*, 2007.

[25] Sanjiv Das and Mike Chen. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of Athe Asia Pacific Finance Association Annual Conference*, 2001.

[26] Sanjoy Dasgupta, Michael Littman, and David McAllester. Pac generalization bounds for co-training. In *Neural Information Processing Systems*, 2001.

[27] Hal Daume. Frustratingly easy domain adaptation. In *Association for Computational Linguistics ACL*, 2007.

[28] Hal Daume and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 2006.

[29] Jason Farquhar, David Hardoon, Hongying Meng, John Shawe-Taylor, and Sandor Szedmak. Two-view learning: Svm-2k, theory and practice. In *Neural Information Processing Systems*, 2005.

[30] R. Florian, H. Hassan, A.Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. A statistical model for multilingual entity detection and tracking. In *Human Language Technologies and the North American Association for Computational Linguistics*, 2004.

[31] Larry Gillick and Stephen Cox. Some statistical issues in the comparison of speech recognition algorithms. In *ICASSP*, 1989.

[32] Andrew Goldberg and Xiaojin Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*, 2004.

[33] David Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis; an overview with application to learning methods. Technical Report CSD-TR-03-02, Royal Holloway, University of London, 2003.

[34] Trevor Hastie, Rob Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[35] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[36] H. Hotelling. The most predictable criterion. *Journal of Educational Psychology*, 1935.

[37] Jiayuan Huang, Alex Smola, Arthur Gretton, Karsten Borgwardt, and B. Schoelkopf. Correcting sample selection bias by unlabeled data. In *NIPS*, 2007.

[38] Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[39] Jing Jiang and Chengxiang Zhai. Instance weighting for domain adaptation. In *ACL*, 2007.

[40] Sham Kakade and Dean Foster. Multi-view regression via canonical correlation analysis. In *Conference on Learning Theory*, 2007.

[41] Michael Kearns and Umesh Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[42] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning ICML*, 2001.

[43] Matthew Lease and Eugene Charniak. Parsing biomedical literature. In *IJCNLP*, 2005.

[44] Chang Liu and Hwee Tou Ng. Learning predictive structures for semantic role labeling of nombank. In *Association for Computational Linguistics ACL*, 2007.

[45] Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[46] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[47] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *North American Association for Computational Linguistics*, 2006.

[48] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *ACL*, 2005.

[49] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Learning to classify text from labeled and unlabeled documents. In *National Conference on Artificial Intelligence AAAI*, 1998.

[50] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of Association for Computational Linguistics*, 2005.

[51] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of Empirical Methods in Natural Language Processing*, 2002.

[52] PennBioIE. Mining The Bibliome Project, 2005. http://bioie.ldc.upenn.edu/.

[53] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Empirical Methods in Natural Language Processing EMNLP*, 1996.

[54] Brian Roark and Michiel Bacchiani. Supervised and unsupervised PCFG adaptation to novel domains. In *HLT-NAACL*, 2003.

[55] David Rosenberg and Peter Bartlett. The rademacher complexity of co-regularized kernel classes. In *Conference on Artificial Intelligence and Statistics*, 2007.

[56] Eric Tjong Kim Sang. The conference on natural language learning shared task: Language-independent named entity recognition, 2003.

[57] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *North American Association for Computational Linguistics NAACL*, 2003.

[58] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

[59] Masashi Sugiyama. Input-dependent estimation of generalization error under covariate shift. *Statistics and Decisions*, 23:249–279, 2005.

[60] Ben Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2007.

[61] Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

[62] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of Human Language Technologies / North American Association for Computational Linguistics HLT/NAACL*, 2003.

[63] Peter Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of Association for Computational Linguistics*, 2002.

[64] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[65] Vladimir Vapnik and Alexi Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[66] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the Association for Computational Linguistics*, 2002.

[67] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Association for Computational Linguistics*, 1995.

[68] Sarah Zelikovitz and Haym Hirsh. Using lsi for text classification in the presence of background text. In *Conference on Information and Knowledge Management*, 2001.

[69] Tong Zhang. Solving large-scale linear prediction problems using stochastic gradient descent algorithms. In *International Conference on Machine Learning ICML*, 2004.

[70] Jerry Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin – Madison, 2007.

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

[72] Xiaojin Zhu, Jaz Kandola, Zoubin Ghahramani, and John Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Neural Information Processing Systems*, 2004.

[73] 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.