# SOME MODELS AND MEASURES FOR LEARNING ON A BUDGET

by

Avishek Saha

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

December 2012

**The University of Utah Graduate School**

# STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Avishek Saha**

has been approved by the following supervisory committee members:

| | | |
|---|---|---|
| **Suresh Venkatasubramanian** | , Chair | 4th October, 2012<br>Date Approved |
| **Sanjoy Dasgupta** | , Member | 27th October, 2012<br>Date Approved |
| **Hal Daumé III** | , Member | 16th August, 2012<br>Date Approved |
| **Juliana Freire** | , Member | 15th October, 2012<br>Date Approved |
| **Ellen Riloff** | , Member | 16th August, 2012<br>Date Approved |

and by **Alan L. Davis** , Chair of

the Department of **School of Computing**

and by **Charles A. Wight** , Dean of the Graduate School.

# ABSTRACT

Machine learning is the science of building predictive models from data that automatically improve based on past experience. To learn these models, traditional learning algorithms require labeled data. They also require that the entire dataset fits in the memory of a single machine. Labeled data are available or can be acquired for small and moderately sized datasets but curating large datasets can be prohibitively expensive. Similarly, massive datasets are usually too huge to fit into the memory of a single machine. An alternative is to distribute the dataset over multiple machines. Distributed learning, however, poses new challenges as most existing machine learning techniques are inherently sequential. Additionally, these distributed approaches have to be designed keeping in mind various resource limitations of real-world settings, prime among them being intermachine communication.

With the advent of big datasets machine learning algorithms are facing new challenges. Their design is no longer limited to minimizing some loss function but, additionally, needs to consider other resources that are critical when learning at scale. In this thesis, we explore different models and measures for learning with limited resources that have a budget. What budgetary constraints are posed by modern datasets? Can we reuse or combine existing machine learning paradigms to address these challenges at scale? How does the cost metrics change when we shift to distributed models for learning? These are some of the questions that have been investigated in this thesis. The answers to these questions hold the key to addressing some of the challenges faced when learning on massive datasets.

In the first part of this thesis, we present three different budgeted scenarios that deal with scarcity of labeled data and limited computational resources. The goal is to leverage transfer information from related domains to learn under budgetary constraints. Our proposed techniques comprise semisupervised transfer, online transfer and active transfer. In the second part of this thesis, we study distributed learning with limited communication. We present initial sampling based results, as well as, propose communication protocols for learning distributed linear classifiers.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

It feels great to thank the many people whose support and constant encouragement over the years made this thesis possible.

Foremost, I would like to thank my advisor Suresh Venkatasubramanian for the privilege, and the pleasure, of allowing me to work under him towards my PhD degree. His immense knowledge, deep insight and supreme enthusiasm have been truly inspiring during the course of this research work. I sincerely thank him, particularly for the freedom he gave me and for all the faith he had in me. His enjoyable company and relentless support through thick and thin have been an absolute honor.

I would also like to express my sincere gratitude to my committee members, Sanjoy Dasgupta, Hal Daumé III, Juliana Freire, Ellen Riloff and fellow collaborator Jeff Phillips. This thesis would not have materialized without their support.

There are many friends in Utah who made my PhD experience memorable and helped me preserve my sanity during times of stress. A few of them deserve particular mention: Piyush Rai (for countless enjoyable conversations on machine learning and (hindi) movies), Niladrish Chatterjee (for enjoyable cooking and soccer sessions), Protonu Basu (for the endless cups of *chai*), Anusua Trivedi (for the fun lady she is), Parasaran Raman (for the much needed distractions), and John Moeller (for informative tidbits on life, maths and otherwise). I thank all of them for making difficult times bearable and leisure times cherishable.

I wish to thank my parents, Aveek Kumar Saha and Chandralekha Saha and my entire extended family. Needless to say, this work would not have been completed without their constant encouragement, love and support.

Finally, but most importantly, I feel deeply indebted to my dear wife, Namrata Dey, for her understanding, patience and love. The past two years spent with her have been the best times of my life. Her joyful spirit can light up the darkest of times. She is my peace in this world of chaos. I hope she continues to encourage me in all my future endeavors.

# CHAPTER 1

# INTRODUCTION

“Big data” refers to datasets that are typically beyond the ability of a single (but powerful) machine or a modern data management systems to store and analyze. Big data is everywhere. Companies capture terabytes of information about customers, vendors, etc. from transaction web logs. Millions of handheld devices act as networked sensors that churn out a huge amount of information content. Microblogging and social networking sites generate massive volumes of user modeled personalized feeds. Despite the advantages of vast amounts of information that are contained in these datasets, the sheer size and scale of these datasets pose significant challenges in automated extraction of information. These big datasets need to be accurately captured and efficiently analyzed to mine valuable information.

Machine learning is a branch of computer science that improves automatically by learning from past experience. For example, machines can learn to identify spam mails, can recognize objects of interest from an image, identify probable occurrence of diseases by analyzing patient profiles and can even learn to drive a car without manual intervention. These algorithms specialize in finding patterns in datasets and thus, are crucial for automated analysis of massive chunks of data. Learning methods come in two flavors, (i) *supervised* where the system learns from “labeled” data, and (ii) *unsupervised* where no data labeling is required. Additionally, *online* learning algorithms learn on one data point at a time. Usually online learning is favored for efficiency reasons. However, in many cases, the problem settings are inherently online where new data arrives one at a time.

When learning a task, one possible approach is to design a (possibly sophisticated but complex) model and train it on small toy datasets. However, in the presence of massive datasets, an alternative approach that has gained much traction in recent years is to design simple but highly scalable systems and train them on lots of training data. Indeed, the advantages of huge amounts of training data are undeniable and have been highlighted in (Halevy et al., 2009). These advantages, of learning on large quantities of data, are conspicuous in many other fields as well, for example, information retrieval and machine translation. Unfortunately, training data requires labels and this labeling requirement of supervised learning is seldom met by big datasets. Curating and annotating these large datasets demands massive amounts of human time and effort. Consider, for example, image categorizers that require lots of manually annotated image data to learn an accurate model. Manual tagging is costly and time consuming; but copious amounts of untagged images are readily available from keyword based image search engines. Additionally, in many cases, these datasets are so big that they seldom fit into the memory of a computer. This calls for learning techniques that are allowed to either sample the data or see each data point only once. Consider, for example, Google News which provides personalized news feed to (millions of) subscribers. Each user/task has a small amount of data to start with but, when taken together for all users, the total amount of data is huge. Moreover, it is infeasible to store all news articles that interest all Google News readers and the learning algorithm is allowed to see the data for each task only once and then make a prediction on it.

In recent years, huge increases in computing power has brought dramatic changes to the field of machine learning. This has resulted in many new developments in machine learning that aim to build on advances in multicore and distributed computing to tackle challenges when learning at scale. However, traditionally the machine learning community has assumed sequential algorithms that run on a single processor. Unfortunately, this assumption no longer holds for big datasets. Moreover, in many cases, the data is inherently distributed, being collected at geographically distant locations. Consider, for example, a mobile network where sensors embedded in (millions of) individual phones collect data for local classifiers, but each node is unable to see data collected at other nodes and hence, needs to communicate in order to learn a global classifier. As a result, machine learning techniques need to be designed that go beyond single-processor approaches and perform well in a parallel setting or over a cluster of distributed machines.

This thesis is concerned with scenarios that arise when learning over big data. Learning over large datasets differ from traditional learning settings and require different models and measures of costs. Instead of addressing the issue of scalability of learning algorithms, in this thesis, we focus on different cost metrics and learning models that seek to learn on a budget. We believe that algorithms built on these principles and cost metrics address some of the core issues of big data learning. Thus, our work can be seen as a stepping stone towards designing scalable algorithms that learn under budgetary constraints. The need for efficient data analysis on huge datasets is also drawing interest from other research communities. In space research big telescopes generate astronomical amounts of image data. In neuroscience a cubic millimeter of brain maps to a petabyte of data! Thus, with the advent of big data, data analysis is undergoing a paradigm shift. When operating at such huge scale, new cost metrics emerge that are useful for large scale learning algorithms and reducing these cost metrics becomes a critical component in the overall performance improvement of the learning system. We propose and develop new learning models and measures that make learning possible and efficient when on a budget.

## 1.1 Thesis Statement

This thesis aims to explore some aspects of learning when on a budget. The chapters that follow address (i) transfer learning strategies on a label budget, (ii) transfer learning scenarios in online settings, (iii) transfer learning when labels are costly to acquire, and (iv) distributed learning under limited or minimal communication. Specifically, we focus on the following hypotheses:

1. strategies that leverage abundantly available unlabeled data to improve transfer learning.

2. strategies that learn transfer relationships from data rather than imposing a priori assumptions on the model.

3. strategies that leverage transfer information to reduce label query costs in active settings.

4. models and algorithms that seek to minimize communication when learning a global classifier over data distributed across multiple locations.

## 1.2 Organization of this Thesis

The rest of the thesis is organized into two parts. In Part I, we present models for transfer learning on a budget. In Part II, we study distributed models for learning classifiers with low communication. These two parts comprise a total of six chapters. In the following, we give an overview of each of these chapters.

**Chapter 2** This chapter surveys existing literature on some areas of machine learning relevant to this thesis. First, we provide a description of transfer learning. Next, we present existing work on budgeted learning. We consider three types of budgetary constraints, namely, scarcity of labels, limited availability of computational resources and costs associated with querying labels. Thus, in this section, we describe existing work on semisupervised learning, online learning and active learning. Thereafter, we discuss prior work on models and methods for distributed learning.

**Chapter 3** This chapter presents semisupervised approaches to transfer learning. We focus on domain adaptation, a subarea of transfer learning, and extend an existing supervised domain adaptation algorithm (EASYADAPT) to leverage the usefulness of unlabeled data. Our proposed algorithm is theoretically sound as shown by our derived generalization bounds. In addition, we empirically demonstrate its superior performance on sentiment classification and sequence labeling tasks.

**Chapter 4** This chapter proposes an online algorithm for multitask learning. Most existing work in multitask learning a priori assume some known form of task relatedness. In this work, we show that we can instead learn the task relationships from the data. Our formulation allows us to derive closed-form incremental update rules that are amenable for online settings.

**Chapter 5** Active learning strategies have been mostly proposed in the context of single tasks. This chapter studies whether active learning learning techniques are useful in transfer settings. More specifically, given a source domain (with labeled examples) and a target domain (with unlabeled examples), can we learn a classifier in the source and use this classifier to actively learn in the target? This work answers this question in the affirmative and shows that active learning strategies indeed benefit from information transferred from related domains, when available.

**Chapter 6** Contrary to existing machine learning algorithms that assume that all of the data is available at a single location, distributed machine learning algorithms can learn a classifier over data that are geographically distributed over multiple locations. One aspect of learning in a distributed fashion is to forego the need for a powerful machine, particularly when designing computationally intensive algorithms on large dataset, and instead learn using multiple low-end commodity hardware. The hope is that multiple moderately powerful machines can leverage distributed processing techniques to computationally outperform a single powerful machine based processing. However, another aspect, which has been mostly unaddressed until now, is to reduce communication between machines while carrying out distributed learning. In this chapter, we present a new model and algorithms for distributed learning that aims to minimize communication when learning a global classifier over data distributed across multiple machines.

**Chapter 7** Finally, this chapter concludes this thesis and provides future directions.

# CHAPTER 2

# RELEVANT WORK

In this chapter, we present background material that serves as a reference for the subsequent chapters of this thesis. We start with transfer learning and explain key definitions with examples. Next, we discuss learning under budgetary constraints where the budget could be on labels or on computational resources. The three budgeted learning areas that we highlight are semisupervised learning, online learning and active learning. Finally, we explore the existing landscape of distributed learning.

## 2.1 Transfer Learning

First, we introduce some notations. Let $\mathscr{X} \subset \mathbb{R}^d$ denote the instance space and $\mathscr{Y}$ $= \{-1,+1\}$ denote the label space. Let $\mathscr{D}$ denote a domain over some joint distribution $P(X,Y)$. The marginal distributions of $\mathscr{X}$ and $\mathscr{Y}$ are denoted by $P(X)$ and $P(Y)$, respectively and the conditional distributions are denoted by $P(X|Y)$ and $P(Y|X)$. In what follows, most definitions are drawn from the excellent survey by Pan and Yang (2010) on transfer learning. However, where appropriate, we deviate from the original definitions and make necessary modifications.

**Definition 1 (Domain Pan and Yang (2010))** *A* ***domain*** *$\mathscr{D}$ is defined as the tuple $\{\mathscr{X}, P(X,Y)\}$, where $\mathscr{X}$ represents the feature space of the set of instances $X = \{x_1, x_2, \ldots, x_m\} \in \mathscr{X}$ sampled from the marginal probability distribution $P(X)$. $P(X,Y)$ represents the joint distribution over the instance and label space.*

*Example:* Let us consider documents collected from multiple sources, namely, Wall Street Journal (WSJ), movie reviews from IMDB (MOV), technical paper abstracts (TECH), biomedical abstracts from PubMed (BIO). Here, the feature set (the bag of words) and their dimensions are different. Additionally, the joint probability distributions are also different. For example, the words “bulls” and “bears” have different meanings, frequency of occurrences and labels in WSJ and MOV. Hence, WSJ, MOV, TECH and BIO are different domains.

**Definition 2 (Task Pan and Yang (2010))** *Given a specific domain $\mathscr{D} = \{\mathscr{X}, P(X,Y)\}$, a* ***task*** *is represented by the tuple $\{\mathscr{Y}, f(\cdot)\}$, where $\mathscr{Y}$ denotes the label space and $f(\cdot)$ denotes the predictive function to be learned by the task.*

Note that that the goal of a predictive function is to predict the class label $y$ given an instance $x$. Hence, the predictive function $f(\cdot)$ is equivalent to the conditional probability $P(Y|X)$. In what follows, we will use the two interchangeably. *Example:* For WSJ the task could be POS (part-of-speech) tagging, for MOV the task could be sentiment classification (categorize reviews as positive or negative), for PUB the task could be drug identification or NER (named entity recognition). Each of these tasks have a different label set. For example, in PUB domain, the tasks NER and drug identification have different sets of labels although the domain is the same. Hence, these are two different tasks.

**Definition 3 (Transfer Learning (TL) Pan and Yang (2010))** *Consider a learning task $\mathscr{T}_S$ in source domain $\mathscr{D}_S$ and a learning task $\mathscr{T}_t$ in target domain $\mathscr{D}_t$. The goal of* ***transfer learning*** *is to learn a target function $f_t$ in the target domain with the help of the source function $f_S$ learned in the source domain, where $\mathscr{D}_S \neq \mathscr{D}_t$ and $\mathscr{T}_S \neq \mathscr{T}_S$.*

*Example:* Use a POS tagger (source task $\mathscr{T}_S$) trained on WSJ (source domain $\mathscr{D}_S$) to learn an NER (target task $\mathscr{T}_t$) on PUB domain (target domain $\mathscr{T}_t$).

**Definition 4 (Domain Adaptation (DA))** *Consider a learning task $\mathscr{T}_S$ in a source domain $\mathscr{D}_S$ and a learning task $\mathscr{T}_t$ in a target domain $\mathscr{D}_t$, where $\mathscr{D}_S \neq \mathscr{D}_t$ but $\mathscr{T}_S = \mathscr{T}_t$. Let $L_S(\sim P_S(X,Y))$ and $L_t(\sim P_t(X,Y))$ denote labeled data in source and target, respectively. Suppose we have $|L_S| = l_S \gg |L_t| = l_t$ (usually $l_t$ is very small or even zero). The goal of* ***domain adaptation*** *is to learn a predictive function $f_S$ in source (using plenty of source labeled data $L_S$ and maybe some labeled data $L_t$ from the target) such that $f_S$ reasonably approximates $f_t$ and predicts well in the target.*

According to the above definition, $\mathscr{D}_S \neq \mathscr{D}_t$ implies either $\mathscr{X}_S \neq \mathscr{X}_t$ or $P_S(X,Y) \neq P_t(X,Y)$ or both. In addition, $P_S(X,Y) \neq P_t(X,Y)$ can be manifested either as $P_S(Y|X)P_S(X) \neq$

$P_t(Y|X)P_t(X)$ which implies $P_S(X) \neq P_t(X)$ (since in DA the conditional probability distribution remains unchanged) or $P_S(X|Y)P_S(Y) \neq P_t(X|Y)P_t(Y)$ which implies $P_S(Y) \neq P_t(Y)$ (since in DA the conditional probability distribution remains unchanged). The former is known as *covariate shift* (Shimodaira, 2000) while the latter is called the *class imbalance* problem (Chan and Ng, 2006). *Example:* An example of $\mathscr{X}_S \neq \mathscr{X}_t$ is when POS taggers are trained on English text data but need to be predicted on French text, whereas an example of $P_S(X,Y) \neq P_t(X,Y)$ is when POS tags are learned on WSJ domain and need to be predicted for TECH domain.

**Definition 5 (Multitask Learning (MTL))** *Consider a learning task $\mathscr{T}_S$ in a source domain $\mathscr{D}_S$ and a learning task $\mathscr{T}_t$ in a target domain $\mathscr{D}_t$, where $\mathscr{D}_S = \mathscr{D}_t$ but $\mathscr{T}_S \neq \mathscr{T}_t$ (but are assumed to be related in some sense). Let $L_S(\sim P_S(X,Y))$ and $L_t(\sim P_t(X,Y))$ denote labeled data in source and target, respectively, such that both $|L_S| = l_S$ and $|L_t| = l_t$ are very **small**. The goal of **multitask learning** is to **simultaneously** learn the predictive functions $f_S$ and $f_t$ using the labeled data $(l_S + l_t)$ such that both $f_S$ and $f_t$ can predict well individually on unseen test data.*

Although the above definition of MTL is in context of two tasks, the more popular convention is to consider multiple tasks which are related in some manner. The relations between tasks are either assumed to be known a priori or can be learned from the data. With reference to the above definition, $\mathscr{T}_S \neq \mathscr{T}_t$ implies either $\mathscr{Y}_S \neq \mathscr{Y}_t$ or $f_S \neq f_t$ or both. As we have already mentioned, equivalent representations of $f_S$ and $f_t$ are given by $P_S(Y|X)$ and $P_t(Y|X)$. Hence, in MTL we have $P_S(Y|X) \neq P_t(Y|X)$. *Example:* Given three datasets of MOV, each of which has small amount of labeled data, the goal is to learn a sentiment classifier on the first dataset, an NER tagger on the second dataset and a POS tagger on the third dataset. In this case, $P(X)$ is same for all but $P(Y|X)$ or the predictor function varies for each task.

### 2.1.1 Domain adaptation

Most existing domain adaptation techniques can be categorized as either an *instance re-weighting* based approach or *change of representation* based approach. In addition, domain adaptation techniques have been proposed in terms of *learning the marginal* or *conditional distributions*, *bayesian learning techniques* and *ensemble methods*.

*Instance re-weighted* domain adaptation follows a re-weighting strategy for input data points (i.e., the instances). This can be achieved by increasing the weights of source instances that are close to the instances in the target domain, and decreases the weights of source instances that are far away from the instances in the target domain. As a result, the learner re-weights the (loss on the) instances of one domain to make it look similar to the other domain. Other approaches include a principled method of using nonparametric kernel density estimation (Shimodaira, 2000), transformation of the density ratio estimation problem into a problem of predicting whether an instance is from the source domain or from the target domain (Bickel and Scheffer, 2006, Zadrozny, 2004), transforming density estimation into a kernel mean matching problem in a reproducing kernel Hilbert space (Huang et al., 2007), and learning the instance ratio together with the classification model parameters (Bickel and Scheffer, 2006). However, the class distributions may also be different in the source and the target domains which is known as the *class imbalance* problem (Chan and Ng, 2006). Class imbalance problems are usually addressed (Chawla et al., 2002, Kubat and Matwin, 1997, Zhu, 2007) by oversampling the under-represented classes and undersampling the over-represented classes. As a result, the resampled training instances from the source domain have roughly the same class distribution as the data instances from the target domain.

Another approach to domain adaptation is *change of feature representations*. Feature representation based domain adaptation discovers a shared feature space either in the original feature space (Blitzer et al., 2006, Pan et al., 2010a), or in the transformed subspace (Dai et al., 2009, 2007a, Gupta et al., 2010, Ling et al., 2008, Long et al., 2010, Pan et al., 2010b, Wang et al., 2009, Zhuang et al., 2011). The hope is that the source and target distributions would be close to each other in this shared feature space. Feature correspondence, identified by modeling the relationships between cross-domain features, are a popular approach to discover shared features in the original feature space. On the other hand, dimensionality reduction presents a transformed subspace obtained by extracting an underlying common structure. Domain adaptation is caused due to differences in the joint distribution of the source and the target. While the representation of the labels remain the same across domains, the instances can have different feature representations. The existing feature representation transfer learning methods focus on learning either the marginal distribution or the conditional distribution for knowledge transfer. One popular technique is to assume that under some linear transformation, the source and domain agree on a common representation. Now the goal is to learn this linear transformation. This technique was formally analyzed in Ben-David et al. (2006). The authors proved that the generalization bound for domain adaptation is affected by the distance between the source and target domains. Satpal and Sarawagi (2007) proposed a simple transformation that selects a feature subset that minimizes an approximate distance between source and target distributions. The Structural Correspondence Learning (SCL) algorithm by Blitzer et al. (2006), which built on key ideas from Ando and Zhang (2005), obtained a low-rank representation amenable for domain adaptation using unlabeled data from the target domain. The fact that the distance between domains is indeed decreased by the low-rank representation of SCL has been empirically shown in Ben-David et al. (2006).

Among other techniques for domain adaptation, *learning the marginal distribution* can be achieved by Co-Clustering based Classification (CoCC) (Dai et al., 2007a) and Label Propagation (Wang et al., 2009). Transfer of common association between feature clusters and example classes, which can be regarded as learning the conditional distribution, can be achieved by collaborative dual Probabilistic Latent Semantic Analysis (PLSA) (Zhuang et al., 2010) and Matrix Tri-Factorization based classification (MTrick) (Zhuang et al., 2011). Joint subspace Nonnegative Matrix Factorization (Gupta et al., 2010) is a method for learning the marginal distribution only. It does not learn the conditional distribution which makes it difficult to be applied to cross-domain classification tasks. The idea of learning both the marginal and conditional distributions was pioneered in two methods on cross domain distribution adaptation. They are kernel mapping (Zhong et al., 2009) and dual knowledge transfer (Wang et al., 2011). Another proposed method (Dual Transfer Learning or DTL) (Long et al., 2012)) simultaneously learns the marginal and conditional distributions that exploits the duality between these two distributions which is a crucial step in knowledge transfer.

*Bayesian approaches* proposed for domain adaptation usually constructed a prior using labeled instances from the source domain and then estimated the model parameters for the target domain. Li and Bilmes (2007) showed how this general prior can be instantiated for generative classifiers and discriminative classifiers. A Bayesian prior for adapting a maximum entropy capitalizer across domains was proposed in Chelba and Acero (2006).

*Ensemble methods* form another family of techniques for domain adaptation. A mixture model of three components was proposed in Daumé and Marcu (2006). Of the three components, one was shared by both the source and the target domains, one was specific to the source, and one was specific to the target. A conditional expectation maximization (CEM) algorithm was used to learn this three-component mixture model using labeled data from both the source and the target. Storkey and Sugiyama (2006) considered a more general mixture model. In their model, which was learned using the expectation maximization (EM) algorithm, the source and the target domains shared more than one mixture components and no labels from the target are required. A boosting based algorithm to combine multiple weak learners so as to obtain a final domain adapted classifier was proposed in Dai et al. (2007c).

### 2.1.2 Multitask learning

Existing work in Multi Task Learning (MTL) can be categorized into two broad areas, (a) *techniques that assume that the task parameters lie close to each other*, and (b) *techniques that assume that multiple tasks share a common (possibly low dimensional) feature space*.

The notion of task parameters lying close to each other can be manifested as using either a (i) *regularized norm that brings the task parameters close*, or (ii) *imposing a common prior/hyperprior on task parameters that enforces task closeness*.

*Regularized norm that brings the task parameters close:* Evgeniou and Pontil (2004) is the first work that generalized the notion of "regularization" from single task learning settings to multitask learning and presented a kernel-based extension for Support Vector Machine (SVM) based MTL. They followed the intuition of hierarchical Bayes and assumed that all the weight vectors can be written as a summation of a mean weight vector ($w_0$) and a noise vector ($v_t$) where the noise vectors are small when the tasks (subscripted by $t$) are similar to each other. The tasks are related in a way that the true task models are all close to some mean model $w_0$ (which played the role of the mean of the Gaussian used for hierarchical Bayes). The goal was to estimate all noise vectors $v_t$ as well as the (common) $w_0$ simultaneously. Evgeniou et al. (2005) also used regularization to extend the aforementioned work in Evgeniou and Pontil (2004) to nonlinear classifiers using kernel methods. The key observation was that the $K$ tasks in $\mathbb{R}_d$ can be reduced to a single task in $\mathbb{R}_{Kd}$ by choosing a suitable embedding into a common RKHS space. This reduction allowed one to solve a multitask learning problem by running any kernel-based single-task learning algorithm with a "multitask kernel" (defined in the aforementioned papers). Regularization also played a role in clustering-based MTL. The Task-Clustering (TC) algorithm (Thrun and O'Sullivan, 1996) was the first work to propose clustering of related tasks. J. Abernethy and Vert (2006) assumed that the different tasks are in fact clustered into different groups, and that the weight vectors of tasks within a group are similar to each other. A key difference with Evgeniou et al. (2005), where a similar hypothesis was studied, was that J. Abernethy and Vert (2006) did not assume that the groups are known a priori and the goal was to both identify the clusters and to use them for multitask learning. An important situation that motivated this hypothesis was the case where most of the tasks are indeed related to each other, but a "few outlier tasks" are very different, in which case it may be better to impose similarity or low-dimensional constraints only on a subset of the tasks (thus forming a cluster) rather than to all tasks. Overall, the formulation of J. Abernethy and Vert (2006) was not constrained to the euclidean norm and considered arbitrary norms for penalization, thus generalizing the work of Evgeniou et al. (2005). Some other works that also considered clustering-based multitask learning are Kang et al. (2011), Xue et al. (2007a) and Xue et al. (2007b).

*Imposing a common prior and hierarchical sharing of task parameters (Bayesian):* Hierarchical Bayesian approaches had been proposed in Bakker and Heskes (2003), Heskes (2000). Bakker and Heskes (2003) adopted a hierarchical Bayesian approach in which some of the model parameters are shared (the same for all tasks) and others are more loosely connected through a joint prior distribution that can be learned from the data. They used a mixture of Gaussians for the upper level distribution instead of a single Gaussian. This leads to clustering the tasks, one cluster for each Gaussian in the mixture. A number of approaches for learning multiple tasks were Bayesian, where a probability model capturing the relations between the different tasks was estimated simultaneously with the model parameters for each of the individual tasks. In Allenby and Rossi (1998), Arora et al. (1998) a hierarchical Bayes model is estimated. First, it is assumed a priori that the parameters of the $T$ tasks to be learned are all sampled from an unknown Gaussian distribution. Then, an iterative Gibbs sampling based approach is used to simultaneously estimate both the individual functions and the parameters of the Gaussian distribution. In this model relatedness between the tasks is captured by this Gaussian distribution: the smaller the variance of the Gaussian the more related the tasks are. Zhang et al. (2005) proposed a unified probabilistic framework, where the task parameters share a common structure through latent variables. Other Bayesian based approaches include discovering latent hierarchy using Kingsman Coalescents (Daumé III, 2009), stick breaking processes (already mentioned above) (Xue et al., 2007a,b), and the Indian Buffet Process (IBP) (Rai and Daumé III, 2010). The IBP model assumed that the tasks share a subspace and hierarchically modeled this assumption using IBP to learn the multiple tasks. Semisupervised Bayesian approaches (Liu et al., 2009) that built on Xue et al. (2007b) combined semisupervised learning with MTL with the assumption that there exists a prior joint distribution over the parameters of the multiple tasks. Finally, an online Bayesian method was proposed in Pillonetto et al. (2010).

When multitask learning is formulated as learning a common yet latent feature representation, then most existing works can be divided into two categories: (i) *feature learning to discover a shared feature subset*, and (ii) *learning a low-dimensional linear/nonlinear subspace*.

*Feature learning to discover a shared feature subset:* The first paper (Argyriou et al., 2007a) in this line of work used existing features to learn a new feature representation by learning a $K \times d$ matrix that represented the coefficients of the learned features for the $K$ tasks. In addition, a $2-1$ norm had been enforced which resulted in sparsity of feature selection. Argyriou et al. (2008), which is an extended version of Argyriou et al. (2007a), additionally extended these results to non-linear classifiers (i.e., kernels) and also proved theoretical convergence bounds for an alternating minimization algorithm proposed in the shorter version. Argyriou et al. (2008) assumed that the tasks share a small subset of features (via the feature matrix), and formulated the problem as a squared $\ell_{2,1}$-norm regularized nonconvex optimization problem. This was achieved by adding a mixed-norm regularization term that favors a common sparsity profile in features shared by all tasks. Other than (Argyriou et al., 2007a), the linear subspace assumption has also been exploited (Rai and Daumé III, 2010) within a Bayesian setting where the authors use nonparametric methods like IBP to both infer the dimensionality of the low-dimensional subspace and additionally, enforce sparsity constraints. Argyriou et al. (2007b) used spectral techniques to learn the low-dimensional feature space.

*Learning a low-dimensional linear/nonlinear subspace:* J. Abernethy and Vert (2006) assumed an unknown low-dimensional subspace and penalized the trace norm of the weight matrix which enforced a low-rank solution. Thus, their approach constrained the different weight vectors to live in a low-dimensional subspace. Agarwal et al. (2010) generalized the linear subspace assumption of the above to the assumption of a nonlinear subspace. The key idea, which has also been exploited in standard manifold learning problems, was that the data (and their labels) does not change arbitrarily but instead follow some well-defined (in this case, manifold) structure. As a result, the parameters of related tasks must not vary arbitrarily, but rather, vary smoothly as if lying on a low-dimensional manifold. Thus, the proposed work removes linear constraints of the aforementioned papers and assumes that the tasks instead share a nonlinear subspace. The framed optimization problem is solved using an alternating minimization framework that learns the nonlinear subspace and task parameters simultaneously as in Argyriou et al. (2007b)). First, all task parameters were learned using a Single Task Learning (STL) method, and then these task parameters were used to learn the initial task manifold. The task-manifold was then used to relearn the task parameters using manifold regularization. Learning of manifold and task parameters was repeated until convergence.

*Modeling/Learning Task Relationships:* Argyriou et al. (2007b) modeled the task relationships using a function over the covariance matrix and used it to regularize the task parameter (weight vector) matrix. This did not model the relationships but instead imposed a structure by regularizing the covariance of the weight vector. Xue et al. (2007a,b) also modeled task relationships. Zhang and Yeung (2010) presented a probabilistic approach where the task relationships are not assumed a priori but instead learned from the data. On this note, Zhang and Schneider (2010) addressed a similar setting where the probabilistic approach was taken to simultaneously learn both task relationships as well as feature structure with added benefits of $\ell_1$ constraints that induced sparsity in the learned matrix. Kang et al. (2011) discussed how to learn which are the relevant and related tasks for multitask feature learning.

## 2.2 Budgeted Learning

### 2.2.1 Semisupervised learning

Semisupervised Learning (SSL) is a subtopic of machine learning that aims to learn from both labeled and unlabeled data samples. However, in contrast to the transfer learning paradigm where the labeled and unlabeled data instances are assumed to be drawn from different distributions, in semisupervised learning the labeled and unlabeled data are assumed to be sampled from the same distribution. SSL approaches can be broadly categorized as (i) *generative models and hybrid models*, (ii) *self-training and cotraining based techniques*, (iii) *low-density separator* based approaches, and (iv) *graph-based semisupervised learning*.

*Generative models* that use the Expectation Maximization (EM) algorithm have been shown to perform better (Nigam et al., 2000) than their discriminative counterparts (Baluja, 1999). Discriminative models often perform better (Liang and Jordan, 2008) in terms of classification accuracy when compared to generative models. On the contrary, generative models (Seeger, 2001) have proven useful in many machine learning algorithms that estimate and build on the underlying unlabeled data distribution. *Hybrid models* draw on the advantages of both generative and discriminative models to achieve improved performances in semisupervised settings. Thus, Fujino et al. (2005) extend generative mixture models using discriminative training approaches based on the maximum entropy principle. Other similar examples include Callison-Burch et al. (2004), Miller and Uyar (1997), Shahshahani and Landgrebe (1994).

*Self training* is a popular SSL method where the classifier is first trained with the small amount of labeled data and then used to classify the unlabeled data. The most confident unlabeled points and their predicted labels are added to the training set and the classifier is retrained. Also known as self-teaching or bootstrapping, self-training has been used for word sense disambiguation (Yarowsky, 1995), for identifying subjective nouns (Riloff et al., 2003), dialogue classification (Maeireizo et al., 2004) and parsing and machine translations (Rosenberg et al., 2005). On the other hand, *cotraining* (Blum and Mitchell, 1998, Mitchell, 1999) initially divides the feature space into two subsets and trains two separate classifiers, one each on the two subfeature sets, respectively. Thereafter, in a manner similar to self-training, each classifier then classifies the unlabeled data and instances on which both agree are the added to the training set. The classifier is then trained with the new training data. Cotraining with EM (Co-EM) and other related methods (Collins and Singer, 1999, Jones, 2005) for information extraction from text were based on cotraining. A cotraining algorithm for canonical correlation analysis that needed only one labeled point has been proposed in hua Zhou et al. (2007). The fact that cotraining also works with a single labeled point (in the extreme case) has been theoretically justified in Balcan and Blum (2006). A Probably Approximately Correct (PAC) style theoretical analysis for cotraining has been provided in Dasgupta et al. (2001). Cotraining is based on the assumptions that the features can be split into two sets, each subfeature set is sufficient to train a good classifier, and that the two sets are conditionally independent given the class. However, *multiview models* do not require the assumptions of cotraining but instead require multiple classifiers to train on the same labeled data and make similar predictions on the unlabeled data. Multiview learning has been applied to semisupervised regression (Brefeld et al., 2006, Sindhwani et al., 2005) and structured output spaces (Brefeld et al., 2005, Brefeld and Scheffer, 2006).

Another well known assumption in SSL is that the learned decision boundary passes through *low-density regions*. Hence, the unlabeled data can be used to guided the linear decision boundary away from dense regions. This forms the basis of Transductive SVMs (TSVM) (Wang, 2007), an extension of SVMs with unlabeled data. TSVMs aim to place the decision boundary away from the dense regions by finding a linear separator that has maximum margin on the labeled and unlabeled data. Early algorithms (Bennett and Demiriz, 1998, Demiriz and Bennett, 2000, Fung and Mangasarian, 2001) for SVMs did not scale beyond a few hundred samples. SVM-light, a faster implementation that has achieved much popularity was proposed in Joachims (1999). The TSVM training problem has been framed as a semidefinite programming (SDP) in Bie and Cristianini (2003, 2006). Gaussian process based semisupervised models for TSVMs have been proposed in Chu et al. (2007b) and Lawrence and Jordan (2005).

*Graph based methods* for semisupervised learning have received much interest and research contributions. Most graph-based SSL approaches can be commonly modeled as a cost function to be optimized that contains a loss function and a regularizer. Hence, the individual approaches differ in their choice of the loss function and the regularizer. Semisupervised learning was posed as a graph MinCut problem in Blum and Chawla (2001). In their approach, the positive labels acted as sources and negative labels acted as sinks and the goal was to find a minimum set of edges necessary for flow from the sources to the sinks. A discrete Markov Random Field based approach was attempted in Zhu et al. (2003). But the inference problem in this case was rather difficult. A more difficult technique that involved the computation of marginal probabilities of the discrete random field was proposed in Getz et al. (2006). A loss function and Tikhonov regularizer based algorithm was proposed in Belkin et al. (2004). Manifold regularization, another popular graph regularization based semisupervised technique, was proposed in **?**. Manifold regularization was extended to kernels (Chapelle et al., 2003, Smola and Kondor, 2003) by showing that the spectral transformation of a Laplacian results in kernels suitable for semisupervised learning. Follow up work along these lines proposed the diffusion kernel (Kondor and Lafferty, 2002) and the spectral graph transducer (Joachims, 2003). Additional work using Markov random walks and density-sensitive connected graphs were explored in Szummer and Jaakkola (2001) and Chapelle and Zien (2005), respectively.

### 2.2.2 Online learning

In online learning the learner predicts the label for each sample and after each prediction the learner is presented with the true label. If the learner has made a mistake then it is allowed to use the true label to improve its classifier. Thus, in online setting, the learner incrementally improves its hypothesis and this continues as long as the learner receives new samples. Examples of online learning include stock market prediction where the learner predicts tomorrow's value of some particular stock. The true value of the stock is known the day after. Similarly, consider spam filtering where for each mail the inbox predicts "spam" or "ham". Whenever the inbox makes an incorrect prediction the user provides the true label. In the above examples we consider the 0-1 loss. Other loss functions appropriate in online settings are the absolute loss, the square loss, and the log loss (as discussed in Cesa-Bianchi et al. (1997), Foster and Vohra (1993), Vovk (1990)). In online learning, the performance of the online learner is measured by the number of mistakes made by the learner.

The earliest known online algorithm is perceptron (Novikoff, 1962, Rosenblatt, 1958).

The problem of online learning and its mistake bound analysis also has its roots in the problem of predicting from expert advice which was first solved using the Weighted Majority Algorithm (DeSantis et al., 1988, Littlestone and Warmuth, 1994, Vovk, 1990). An attribute efficient algorithm, known as Winnow, was developed in Littlestone (1988). Winnow algorithm had been extremely successful in practice for real-life applications. This spurred follow-up work for noisy cases (Littlestone, 1991). Also, the winnow algorithm was further improved in Auer and Warmuth (1998).

Some of the recent examples in the category of online algorithms include the Relaxed Online Maximum Margin Algorithm (ROMMA) (Li and Long, 2002), the Approximate Large Margin Algorithm (ALMA) (Gentile, 2001), the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003), the Naive Online Regularised-risk Minimisation Algorithm (NORMA) (Kivinen et al., 2004), and the Passive Aggresive (PA) online algorithm (Crammer et al., 2006). MIRA and PA are closely related. In fact, MIRA for binary classification is identical to basic PA. However, MIRA was designed for separable problems only, whereas PA applies to nonseparable problems. NORMA is based on a stochastic gradient approach. Dredze et al. (2008) proposed a Confidence Weighted (CW) online learning algorithm that used parameter-specific variable learning rates. This is similar to Second Order Perceptron (SOP) (Cesa-Bianchi et al., 2005). Both the algorithms maintained a weight vector and some statistics about previous examples. However, while the SOP modeled certainty with feature counts, CW learning modeled uncertainty with a Gaussian distribution.

Finally, Kalai and Vempala (2002) developed algorithms to solve online linear programming, which is a specific type of online convex programming. These algorithms were similar to the algorithms proposed in Singh et al. (2000) that applied gradient ascent to repeated games. Additionally, there has been extensive work on regret in repeated games and in the experts domain (Blackwell, 1956, Foster and Vohra, 1993).

There are several studies of online gradient descent and related update functions as proposed in Cesa-Bianchi et al. (1997), Herbster and Warmuth (2001), Kivinen and Warmuth (1997). These studies focused on prediction problems where the loss functions are convex Bregman divergences. However, Zinkevich (2003) considered arbitrary convex functions in problems that may or may not involve prediction. Additionally, stochastic gradient descent, which is also a type of online gradient descent (proposed in Bottou and Bousquet (2008)), has seen great success in large scale learning scenarios.

### 2.2.3 Active learning

Active learning is a subfield of machine learning where the learning algorithm is allowed to choose the instances to be labeled. Or in other words, the learner queries the labels of a subset of the training instance which, it thinks, are the most informative and more beneficial to the learning process. As a result, active learning is also known by other names such as Query Learning or Optimal Experimental Design (in the statistics literature). For many supervised learning systems to perform well the learner must often be trained on lots of labeled instances. However, in supervised learning tasks, such as speech recognition and information extraction, the labeling process is difficult, time-consuming and expensive. Active learning systems attempt to overcome the labeling bottleneck by asking queries in the form of unlabeled instances to be labeled by an oracle (e.g., a human annotator). In this way, the active learner aims to achieve high accuracy using as few labeled instances as possible, thereby minimizing the cost of obtaining labeled data. Active learning is well-motivated in many modern machine learning problems where data may be abundant but labels are scarce or expensive to obtain.

Existing techniques in active learning can be classified into three different types: (i) *membership query synthesis*, (ii) *stream-based selective sampling*, and (iii) *pool-based active learning*. In membership query learning, the learner may generate new unlabeled instances from the underlying distribution of the input space and request their labels. Although query synthesis fits many problem settings, it encounters a labeling problem whereby the arbitrary instances generated *de novo* could be awkward to label, particularly if the labeler (also called *oracle*) is a human annotator. In Lang and Baum (1992) membership query learning was used with human oracles to train a neural network to classify handwritten characters. In their experiments, many of the query images generated by the learner contained no recognizable symbols but only artificial hybrid characters that had no natural semantic meaning. In *stream-based selective sampling* (Cesa-Bianchi et al., 2006), data points arrive in a streamed fashion sampled from the actual distribution and the learner has to decide whether or not to query its label. This approach is also known as sequential active learning. For uniform distribution over the input space, selective sampling is similar to membership query learning. For nonuniform and (more importantly) unknown distributions, the queries will be sensible as they come from a real underlying distribution. In order to query an instance its informativeness measure is evaluated to make a biased random decision such that more informative instances are more likely to be queried. Stream based active learning bears similarity with online active learning strategies proposed in Dasgupta et al. (2009) for perceptrons. In *pool-based sampling*, queries are selectively drawn from a pool which is assumed to be static. *Pool-based active learning* (Lewis, 1995) assumes that availability of a small set of labeled data and a large pool of unlabeled data. Instances are queried in a greedy fashion, according to an informativeness measure used to evaluate all instances in the pool. Whereas in stream-based active learning the learner scans through the data sequentially and makes query decisions for each point in the stream, a pool based learner evaluates and ranks the entire collection before selecting the best query.

The aforementioned types of active learning employ numerous different strategies to estimate the informativeness of a sample. *Uncertainty Sampling* (Lewis, 1995) queries those instances whose labels the learner is most uncertain about. A popular example of an instance with the most uncertain label is the datapoint that lies to closest to the current decision boundary. *Query-by-committee (QBC)* algorithm (Seung et al., 1992), another query selection strategy, involves a committee of models that are all trained on the current labeled set and vote on the labels of the current instances. The instance on which the query disagrees the most is considered to be most informative sample. Muslea et al. (2000) constructed a committee of two models by partitioning the feature space. Query-by-boosting (Freund et al., 1997) and query-by-bagging (Abe and Mamitsuka, 1998) employed the ensemble methods of boosting (Freund and Schapire, 1995) and bagging (Breiman, 1996) to construct committees. Diversity among committee members has been addressed in an ensemble-method proposed in Melville and Mooney (2004). In *expected model change*, the learner selects the instances that would bring the greatest change to the current model if the label for the selected instance was known. On the contrary, *expected error reduction* measures how much the generalization error of the model is likely to be reduced (and not how much it is likely to change) after the label of candidate instance is queried. Generalization error can also be reduced indirectly by minimizing the output variance using *variance reduction* strategies that sometimes might yield closed form solutions.

Apart from traditional single task learning of linear and nonlinear classifiers, active learning has also been extended to other machine learning problem settings. Active learning for structured outputs (Roth and Small, 2006a,b) involve active learning strategies for prediction of structured outputs. Active learning algorithms for sequence labeling tasks using probabilistic sequence models like CRFs have been presented in Settles and Craven (2008). Active learning for other structured output tasks have also been proposed for probabilistic sequence models, such as Hidden Markov Models (HMMs) (Dagan and Engelson, 1995, Scheffer et al., 2001) and probabilistic context-free grammars (Baldridge and Osborne, 2004). Many domains with incomplete feature information use active learning for feature acquisition. *Active feature learning* allows the learner to request feature information but incurs additional costs. Zheng and Padmanabhan (2002) proposed to impute the missing feature values and then acquire those values for which the model is least confident. Incremental active feature acquisition (Melville and Mooney, 2004) acquired values for a few salient features at a time. In an opposite scenario to active learning, known as *active class selection*, a learner is allowed to query a known class label and obtaining each instance incurs a cost. Several active class selection query algorithms have been proposed in Lomasky et al. (2007). Most active learning strategies discussed apply to supervised learning strategies that required labels for learning. Hofmann and Buhmann (1998) proposed an active clustering algorithm that generated unlabeled instances that improve clustering performance as compared to random sampling. An active variant of constrained clustering is explored in Grira et al. (2005) where the learner is allowed to query "must-link" and "cannot-link" constraints on similar or dissimilar images. Huang and Mitchell (2006) proposed interactive acquisition of clustering constraints, whereas Andrzejewski et al. (2009) addressed a similar problem for features in a topic modeling setting. Active learning that builds on the clustering structure inherent in the data was studied in Dasgupta and Hsu (2008).

The basic assumptions of active learning settings are too idealistic and may not hold in many real-world settings. We describe a few directions that address practical extensions of traditional active learning scenarios. An SVM based active learning strategy by Brinker

(2003) explicitly incorporated *diversity among instances*. Active learning usually assumes that oracles are always correct. However, in real life, oracles can be noisy due to error in experimental measurements or fatigue of a human annotator. Sheng et al. (2008) proposed several heuristics that addressed oracle uncertainty. Donmez et al. (2009) allowed the annotators to have *different noise levels* and queried only more reliable annotators in subsequent iterations of the learning phases. Oracles with different noise can be analogously framed as oracles with different costs. In order to minimize the overall cost of training an accurate model simply reducing the number of labeled instances does not necessarily guarantee a reduction in overall labeling cost. This has been addressed in Baldridge and Osborne (2004) by prelabeling the instances in structured learning tasks like parsing. Kapoor et al. (2007) proposed an approach that took into account the varying labeling costs while selecting queries. Most work in active learning assumes that the object to be queried is similar to the target labels to be learned. For example, in document classification the learner must query a document and the oracle provides its label. Settles et al. (2008) introduced *alternative forms of query learning* in the context of multiple-instance active learning. Vijayanarasimhan and Grauman (2008, 2009) explored an approach that interleaved queries at varying levels of granularity and cost. Raghavan et al. (2006) proposed tandem learning, an alternative query framework that queries features and instances in tandem. Other methods that incorporated feature-based domain knowledge into supervised and semisupervised learning include Druck et al. (2009), Haghighi and Klein (2006), Mann and McCallum (2010). Traditional active learning has been proposed for single task learning settings. Reichart et al. (2008) addressed a *two-task active learning* scenario and proposed the strategies of alternating selection and rank combination for actively learning both tasks. Qi et al. (2008) proposed active learning strategies for *multilabel* scenarios where images may receive multiple labels from several binary classification tasks. Consider using *the training set selected via active learning for some model to be used to learn another model.* Lewis and Catlett (1994) showed that a training set constructed by an active naive Bayes learner using uncertainty sampling can be used for decision tree classifiers. Tomanek et al. (2007) also showed that information extraction data gathered by a MaxEnt model using QBC can be effectively reused to train a Conditional Random Field (CRF). Hwa (2001) successfully reused natural language parsing data selected by one type of parser to train other types of parsers. Finally, in real-life scenarios active learning cannot go on forever and needs a *stopping criterion*. Several methods based on the intrinsic measures of stability or self-confidence within the learner (when active learning ceases to be useful) have been proposed in Bloodgood and Vijay-Shanker (2009), Olsson and Tomanek (2009), Vlachos (2008).

## 2.3 Distributed Learning

Distributed machine learning has witnessed a recent surge in research interest. With the success of MapReduce (Dean and Ghemawat, 2004) and more distributed models (tailored to machine learning applications, such as, GraphLab (Low et al., 2012)) coming up, the area of distributed learning is receiving research interest like never before. Earliest work on distributed approaches for machine learning appeared in the mid and late nineties (Caragea et al., 2000, Provost and Hennessy, 1996). This was followed by more work in the early half of the last decade. The past few years have been the busiest in terms of research interests and new contributions. However, actual interest in distributed learning predates the earliest mentioned works but were mainly discussed in the related community of artificial intelligence. At this point we note that the thrust of almost all works on distributed learning was to minimize computation. Some of the early papers (Auer et al., 2002) do address communication cost but none of them provide rigorous analysis or theoretical bounds.

Provost and Hennessy (1996) proposed the Distributed Rule Learning (DRL) where the data is partitioned between $K$ different nodes and the goal is communicate in order to learn rules that appear satisfactory to all nodes. The authors assumed an *invariant-partitioning property* which required that rules that are globally satisfied (over the entire data) also appear satisfactory to at least one of the distributed nodes. The authors empirically justified the superior performance of DRL as compared to single node learners. Contrary to another work (Provost et al., 1996) by the same authors, where the global classifier is learned on expensive parallel machines, in this work the authors used a cluster of cheap workstations to achieve their goals. Caragea et al. (2000) proposed a naive approach for learning distributed SVMs where the nodes exchange respective support vectors and iteratively learn new classifiers. However, their algorithm is based on recomputing the vertices of the convex hull (of the positively and negatively labeled points on either side of the hyperplane) which is exponential in dimension and hence, practically infeasible. Auer et al. (2002) proposed a distributed model based on a single layer of parallel perceptrons that required low computation and communication. These savings mainly resulted from avoiding computation of high-precision analog weights of the perceptrons. A distributed learning for probabilistic models on heterogeneous data sources was proposed in Merugu and Ghosh (2005). The authors considered both horizontal and vertical partitioning of data and additionally addressed privacy constraints. They proposed iterative algorithms for model integration that built on maximum likelihood and maximum entropy. The problem of anomaly detection in networks was addressed by Huang et al. (2006) using Principal Component Analysis (PCA) over a distributed framework. Although their work aimed to reduce the communication overhead, it was limited to their specific problem setting and did not generalize to arbitrary classifiers. Their problem setup was similar to recent models of distributed streaming (Cormode et al., 2008).

A different line of work looked at inferencing schemes for distributed learning models. Chu et al. (2007a), Kowalczyk and Vlassis (2005) proposed parallelization of EM optimization algorithms. Newman et al. (2008) proposed two distributed Markov Chain Monte Carlo (MCMC) sampling scheme for Latent Dirichlet Allocation (LDA) (Blei et al., 2003) – (i) a simpler localized Gibbs sampling, and, (ii) a more complicated hierarchical Bayesian extension. A followup work (Asuncion et al., 2008) by the same authors proposed an asynchronous algorithm for distributed Gibbs sampling with applications in distributed LDA.

Distributed (or parallelized) versions of online algorithms have been addressed in Langford et al. (2009), Zinkevich et al. (2010). A lock-free, asynchronous approach to parallelized stochastic gradient was discussed in Recht et al. (2011).

More recently, researchers have looked into MapReduce implementations for machine learning algorithms. The first work to use MapReduce for machine learning was proposed in Chu et al. (2007a). The key contribution was to show that machine learning algorithms that fit the Statistical Query Model (Kearns, 1998) can be written in a “summation form” that facilitates parallel implementation on MapReduce. The success of this paper formed the basis for future development of Apache Mahout (Mahout, 2012), an open-source library of machine learning algorithms on MapReduce. Other MapReduce based algorithms proposed include parallel EM algorithms for online collaborative filtering (Das et al., 2007), parallelized boosting (Palit and Reddy, 2010), MapReduce based distributed tuning of machine learning algorithms (Ganjisaffar et al., 2011), MapReduce based k-means (Zhao et al., 2009), distributed topic models (Newman et al., 2009), and distributed decision trees (Ye et al., 2009), etc.

Despite the popularity and success of MapReduce as a scalable data-processing framework, a number of researchers in the machine learning community consider MapReduce to be unsuitable for machine learning applications (Low et al., 2012). The prime reasons that have been highlighted are inefficacy of MapReduce for iterative computations (as most machine learning algorithms proceed in iterations until convergence) and lack of support for sparse dependencies (since many machine learning algorithms examine and update only a small subset of the parameter variables, for example, when estimating the conditional distribution of a random variable). Numerous alternatives have been considered, such as Picolo (Power and Li, 2010), Dryad (Isard et al., 2007), MapReduce Online (Condie et al., 2010), Twister (Ekanayake et al., 2010), Nexus (Hindman et al., 2009), Spark (Zaharia et al., 2010), Surfer (Chen, Weng, He, and Yang, Chen et al.), Pearce et al. (Pearce et al., 2010), OptiML (Chafi et al., 2011) and, more recently, GraphLab (Low et al., 2012) and SystemML (Ghoting et al., 2011). However, all the proposed frameworks have drawbacks of their own. While MapReduce and Dryad do not support iterative computations, their extensions MapReduce Online, Twister, Nexus, and Spark, lack support for sparse asynchronous dependencies. Picolo and Pearce et al. lack sequential consistency necessary to ensure correctness of machine learning algorithms in distributed settings. GraphLab and OptiML address most of the above concerns. However, none of these models address the communication bottleneck in distributed machine learning settings.

Optimization lies at the heart of many machine learning algorithms. Hence, it is imperative that distributed optimization techniques need to be developed on which distributed learning algorithms can be built. The Alternating Direction Method of Multipliers (ADMM) is an existing framework, proposed much earlier in the 1970s, which has been shown (Boyd et al., 2011) to be well suited for distributed convex optimization tasks. The method combines the augmented Lagrangian method with dual-descent techniques to devise incremental update rules that have convergence guarantees and are well-suited for distributed scenarios. It can be applied to a wide variety of settings; for example, support vector machines, sparse logistic regression, lasso, etc. Other distributed optimization algorithms include distributed dual averaging (Duchi et al., 2010) and techniques proposed in Ouyang and Gray (2011).

Online variants of distributed machine learning techniques have been addressed in Dekel et al. (2010b). Researchers have proposed online variants of ADMM to learn ranking functions in streamed settings (Duh et al., 2011). On the more applicative side, distributed learning algorithms for specific problem settings have been designed for question answering (Sonntag, 2004) and natural language processing (Mann et al., 2009, McDonald et al., 2010).

# PART I

# BUDGETED TRANSFER LEARNING

# CHAPTER 3

# SEMISUPERVISED TRANSFER

A domain adaptation approach for Natural Language Processing (NLP) tasks, termed EASYADAPT (EA), augments the *source domain* feature space using features from labeled data in *target domain* (Daumé III, 2007). EA is simple, easy to extend and implement as a preprocessing step and most importantly is agnostic of the underlying classifier. However, EA requires labeled data in both source and target, and hence applies to fully supervised domain adaptation settings *only*. In this work, we propose a semisupervised approach to leverage unlabeled data for EASYADAPT, which we call EA++, and theoretically as well as empirically demonstrate its superior performance over EA. At this point we note that, in this work *supervised domain adaptation* implies the presence of labeled data in both *source* and *target* and *unsupervised domain adaptation* implies labeled data in only *source*. In *semisupervised domain adaptation*, we also have access to both labeled and unlabeled data in *target*.

As mentioned earlier, EA is remarkably general in the sense that it can be used as a preprocessing step in conjunction with any base classifier. However, one of the prime limitations of EA is its incapability to leverage unlabeled data. Given its simplicity and generality, it would be interesting to extend EA to semisupervised settings. In this work, we propose EA++, a coregularization based semisupervised extension to EA. We also present Rademacher complexity based generalization bounds for EA and EA++. Our generalization bounds also apply to the approach proposed in Evgeniou and Pontil (2004) for domain adaptation setting, where we are only concerned with the error on target domain. The closest to our work is a recent work (Chang et al., 2010) that theoretically analyzes EASYADAPT. Their paper investigates the necessity to combine *supervised* and *unsupervised* domain adaptation (which the authors refer to as *labeled* and *unlabeled* adaptation frameworks, respectively) and analyzes the combination using mistake bounds (which is limited to perceptron-based online scenarios). In addition, their work points out that EASYADAPT is limited to only supervised domain adaptation. On the contrary, our work extends EASYADAPT to semisupervised settings and presents generalization bound based theoretical analysis which specifically demonstrate why EA++ is better than EA.

For example, Domain Adaptation Machine (DAM) (Duan et al., 2009) is a semisupervised extension of SVMs for domain adaptation and presents extensive empirical results. Nevertheless, in almost all of the above cases, the proposed methods either use specifics of the datasets or are customized for some particular base classifier and hence, it is not clear how the proposed methods can be extended to other existing classifiers.

There exists prior work on supervised domain adaptation (and multitask learning) that can be related to EASYADAPT. An algorithm for multitask learning using shared parameters was proposed for multitask regularization (Evgeniou and Pontil, 2004) wherein each task parameter was represented as sum of a mean parameter (that stays same for all tasks) and its deviation from this mean. SVMs were used as the base classifiers and the algorithm was formulated in the standard SVM dual optimization setting. Subsequently, this framework was extended to online multidomain setting in Dredze et al. (2010). Prior work on semisupervised approaches to domain adaptation also exists in the literature. Extraction of specific features from the available dataset was proposed (Arnold and Cohen, 2008, Blitzer et al., 2006) to facilitate the task of domain adaptation. Co-adaptation (Tur, 2009), a combination of cotraining and domain adaptation, can also be considered as a semisupervised approach to domain adaptation. A semisupervised EM algorithm for domain adaptation was proposed in Dai et al. (2007b). Similar to graph based semisupervised approaches, a label propagation method was proposed (Xing et al., 2007) to facilitate domain adaptation.

## 3.1 Background

In this section, we introduce notations and provide a brief overview of EASYADAPT (Daumé III, 2007).

### 3.1.1 Problem setup and notations

Let $\mathscr{X} \subset \mathbb{R}^d$ denote the instance space and $\mathscr{Y} = \{-1,+1\}$ denote the label space. Let $\mathscr{D}_s(x,y)$ be the source distribution and $\mathscr{D}_t(x,y)$ be the target distribution. We have a set of source labeled examples $L_s(\sim \mathscr{D}_s(x,y))$ and a set of target labeled examples $L_t(\sim \mathscr{D}_t(x,y))$, where $|L_s| = l_s \gg |L_t| = l_t$. We also have target unlabeled data denoted by $U_t(\sim \mathscr{D}_t(x))$, where $|U_t| = u_t$. Our goal is to learn a hypothesis $\mathbf{h} : \mathscr{X} \mapsto \mathscr{Y}$ having low expected error with respect to the target domain. In this work, we consider *linear hypotheses* only. However, the proposed techniques extend to nonlinear hypotheses, as mentioned in Daumé III (2007). Source and target empirical errors for hypothesis $\mathbf{h}$ are denoted by $\hat{\varepsilon}_s(\mathbf{h}, f_s)$ and $\hat{\varepsilon}_t(\mathbf{h}, f_t)$, respectively, where $f_s$ and $f_t$ are the true source and target labeling functions. Similarly, the corresponding expected errors are denoted by $\varepsilon_s(\mathbf{h}, f_s)$ and $\varepsilon_t(\mathbf{h}, f_t)$. We will use shorthand notations of $\hat{\varepsilon}_s$, $\hat{\varepsilon}_t$, $\varepsilon_s$ and $\varepsilon_t$ wherever the intention is clear from context.

### 3.1.2 EasyAdapt (EA)

Let us denote $\mathbb{R}^d$ as the *original* space. EA operates in an *augmented* space denoted by $\breve{\mathscr{X}} \subset \mathbb{R}^{3d}$ (for a single pair of source and target domain). For $k$ domains, the *augmented* space blows up to $\mathbb{R}^{(k+1)d}$. The augmented feature maps $\Phi^s, \Phi^t : \mathscr{X} \mapsto \breve{\mathscr{X}}$ for source and target domains are defined as $\Phi^s(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x}, \mathbf{0} \rangle$ and $\Phi^t(\mathbf{x}) = \langle \mathbf{x}, \mathbf{0}, \mathbf{x} \rangle$, where $\mathbf{x}$ and $\mathbf{0}$ are vectors in $\mathbb{R}^d$, and $\mathbf{0}$ denotes a zero vector of dimension $d$. The first $d$-dimensional segment corresponds to commonality between source and target, the second $d$-dimensional segment corresponds to the source domain while the last segment corresponds to the target domain. Source and target domain examples are transformed using these feature maps and the augmented features so constructed are passed onto the underlying supervised classifier. One of the most appealing properties of EASYADAPT is that it is agnostic of the underlying supervised classifier being used to learn in the *augmented* space. Almost any *standard supervised learning approach* (e.g., SVMs, perceptrons) can be used to learn a *linear hypothesis* $\breve{\mathbf{h}} \in \mathbb{R}^{3d}$ in the augmented space. Let us denote $\breve{\mathbf{h}} = \langle \mathbf{g_c}, \mathbf{g_s}, \mathbf{g_t} \rangle$, where each of $\mathbf{g_c}$, $\mathbf{g_s}$, $\mathbf{g_t}$ is of dimension $d$, and represent the *common*, *source-specific* and *target-specific* components of $\breve{\mathbf{h}}$, respectively. During prediction on target data, the incoming target sample $\mathbf{x}$ is transformed to obtain $\Phi^t(\mathbf{x})$ and $\breve{\mathbf{h}}$ is applied on this transformed sample. This is equivalent to applying $(\mathbf{g_c} + \mathbf{g_t})$ on $\mathbf{x}$. An intuitive insight into why this simple algorithm works so well in practice and outperforms most state-of-the-art algorithms is given in Daumé III (2007). Briefly, it can be thought to be simultaneously training two hypotheses: $\mathbf{h_s} = (\mathbf{g_c} + \mathbf{g_s})$ for source domain and $\mathbf{h_t} = (\mathbf{g_c} + \mathbf{g_t})$ for target domain. The commonality between the domains is represented by $\mathbf{g_c}$, whereas $\mathbf{g_s}$ and $\mathbf{g_t}$ capture the idiosyncrasies of the source and target domain, respectively.

## 3.2 EA++: EA Using Unlabeled Data

As discussed in the previous section, the EASYADAPT algorithm is attractive because it performs very well empirically and can be used in conjunction with any underlying supervised *linear classifier*. One drawback of EASYADAPT is its inability to leverage unlabeled target data which is usually available in large quantities in most practical scenarios. In this section, we extend EA to semisupervised settings while maintaining the desirable classifier-agnostic property.

### 3.2.1 Motivation

In multiview approach to semisupervised learning (Sindhwani et al., 2005), different hypotheses are learned using different *views* of the dataset. Thereafter, unlabeled data is utilized to coregularize these learned hypotheses by making them agree on unlabeled samples. In domain adaptation, the source and target data come from two different distributions. However, if the source and target domains are *reasonably close*, we can employ a similar form of regularization using unlabeled data. A prior coregularization based idea to harness unlabeled data in domain adaptation tasks demonstrated improved empirical results (Duan et al., 2009). However, their technique applies for the particular base classifier they consider and hence, does not extend to other supervised classifiers.

### 3.2.2 EA++: EASYADAPT with unlabeled data

In our proposed semisupervised approach the source and target hypotheses are made to agree on unlabeled data. We refer to this algorithm as EA++. Recall that EASYADAPT learns a *linear hypothesis* $\breve{\mathbf{h}} \in \mathbb{R}^{3d}$ in the *augmented* space. The hypothesis $\breve{\mathbf{h}}$ contains common, source-specific and target-specific subhypotheses and is expressed as $\breve{\mathbf{h}} = \langle \mathbf{g_c}, \mathbf{g_s}, \mathbf{g_t} \rangle$. In *original* space (ref. section 3.1.2), this is equivalent to learning a source specific hypothesis $\mathbf{h_s} = (\mathbf{g_c} + \mathbf{g_s})$ and a target specific hypothesis $\mathbf{h_t} = (\mathbf{g_c} + \mathbf{g_t})$.

In EA++, we want the source hypothesis $\mathbf{h_s}$ and the target hypothesis $\mathbf{h_t}$ to agree on the unlabeled data. For an unlabeled target sample $\mathbf{x}_i \in U_t \subset \mathbb{R}^d$, the goal of EA++ is to make the predictions of $\mathbf{h_s}$ and $\mathbf{h_t}$ on $\mathbf{x}_i$, agree with each other. Formally, it aims to achieve the following condition:

$$\begin{aligned} \mathbf{h_s} \cdot \mathbf{x_i} \approx \mathbf{h_t} \cdot \mathbf{x_i} &\iff (\mathbf{g_c} + \mathbf{g_s}) \cdot \mathbf{x_i} \approx (\mathbf{g_c} + \mathbf{g_t}) \cdot \mathbf{x_i} \\ \iff (\mathbf{g_s} - \mathbf{g_t}) \cdot \mathbf{x_i} \approx 0 &\iff \langle \mathbf{g_c}, \mathbf{g_s}, \mathbf{g_t} \rangle \cdot \langle \mathbf{0}, \mathbf{x_i}, -\mathbf{x_i} \rangle \approx 0. \end{aligned} \tag{3.1}$$

The above expression leads to the definition of a new feature map $\Phi^u : \mathscr{X} \mapsto \breve{\mathscr{X}}$ for unlabeled data given by $\Phi^u(\mathbf{x}) = \langle \mathbf{0}, \mathbf{x}, -\mathbf{x} \rangle$. Every unlabeled target sample is transformed using the map $\Phi^u(.)$. The augmented feature space that results from the application of three feature maps, namely, $\Phi^s(\cdot)$, $\Phi^t(\cdot)$ and $\Phi^u(\cdot)$ on source labeled samples, target labeled samples and target unlabeled samples are summarized in Figure 3.1.

As shown in Eq. 3.1, during the training phase, EA++ assigns a predicted value close to 0 for each unlabeled sample. However, it is worth noting that during the test phase, EA++ predicts labels from two classes: $+1$ and $-1$. This warrants further exposition of the implementation specifics which is deferred until the next subsection.

### 3.2.3 Implementation

In this section, we present implementation specific details of EA++. For concreteness, we consider SVM as the base supervised learner. However, these details hold for other *supervised linear classifiers*. In the dual form of SVM optimization function, the labels are multiplied with features. Since, we want the predicted labels for unlabeled data to be 0 (according to Eq. 3.1), multiplication by zero will make the unlabeled samples ineffective in the dual form of the cost function. To avoid this, we create as many copies of $\Phi^u(\mathbf{x})$ as there are labels and assign each label to one copy of $\Phi^u(\mathbf{x})$. For the case of binary classification we create two copies of every augmented unlabeled sample and assign $+1$ label to one copy and $-1$ to the other. The learner attempts to balance the loss of the two copies, and tries to make the prediction on unlabeled sample equal to 0. Figure 3.2 shows the curves of the hinge loss for class $+1$, class $-1$ and their summation. The effective loss for each unlabeled sample is similar to the sum of losses for $+1$ and $-1$ classes (shown in Figure 3.2c).

## 3.3 Generalization Bounds

In this section, we present Rademacher complexity based generalization bounds for EA and EA++. First, we define hypothesis classes for EA and EA++ using an alternate

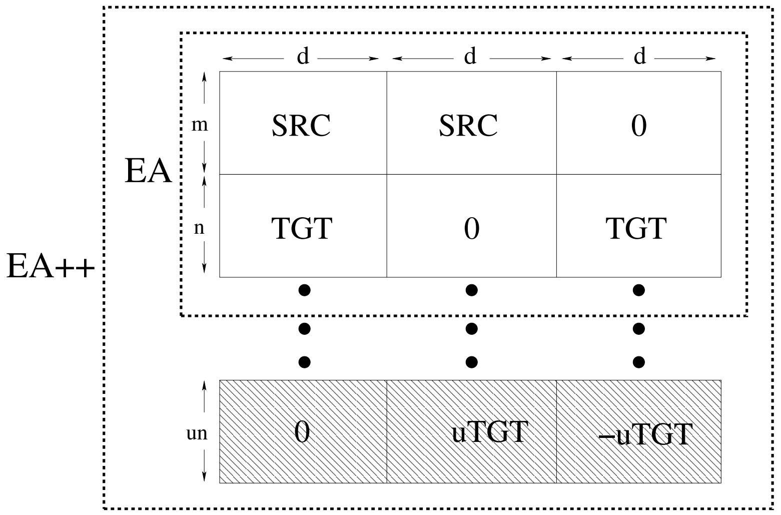


**Figure 3.1**. Diagrammatic representation of feature augmentation in EA and EA++

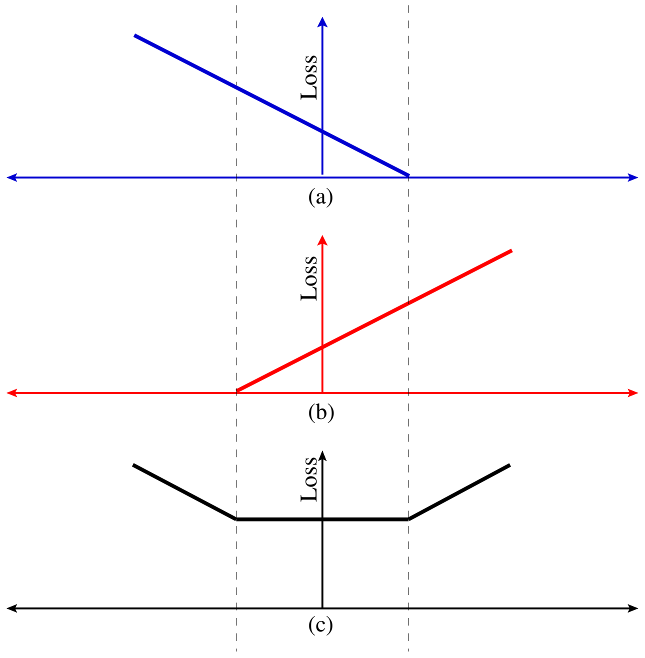


**Figure 3.2**. Loss functions for class $+1$, class $-1$ and their summation.

formulation. Second, we present a theorem (Theorem 3.3.1) which relates *empirical* and *expected* error for the general case and hence, applies to both the source and target domains. Third, we prove Theorem 3.3.2 which relates the *expected target error* to the *expected source error*. Fourth, we present Theorem 3.3.3 which combines Theorem 3.3.1 and Theorem 3.3.2 so as to relate the *expected target error* to *empirical* errors in source and target (which is the main goal of the generalization bounds presented in this work). Finally, all that remains is to bound the Rademacher complexity of the various hypothesis classes.

### 3.3.1 Define hypothesis classes for EA and EA++

Our goal now is to define the hypothesis classes for EA and EA++ so as to make the theoretical analysis feasible. Both EA and EA++ train hypotheses in the augmented space $\breve{\mathscr{X}} \subset \mathbb{R}^{3d}$. The augmented hypothesis $\breve{h}$ is trained using data from both domains, and the three subhypotheses $(\mathbf{g_c}+\mathbf{g_s}+\mathbf{g_t})$ of $d$-dimension are treated in a different manner for source and target data. We use an alternate formulation of the hypothesis classes and work in the original space $\mathscr{X} \subset \mathbb{R}^{d}$. As discussed briefly in section 3.1.2, EA can be thought to be simultaneously training two hypotheses $\mathbf{h_s} = (\mathbf{g_c}+\mathbf{g_s})$ and $\mathbf{h_t} = (\mathbf{g_c}+\mathbf{g_t})$ for source and target domains, respectively. We consider the case when the underlying supervised classifier in augmented space uses a square $L_2$-norm regularizer of the form $||\breve{\mathbf{h}}||^2$ (as used in SVM). This is equivalent to imposing the regularizer $(||\mathbf{g_c}||^2+||\mathbf{g_s}||^2+||\mathbf{g_t}||^2) = (||\mathbf{g_c}||^2+||\mathbf{h_s}-\mathbf{g_c}||^2+||\mathbf{h_t}-\mathbf{g_c}||^2)$. Differentiating this regularizer w.r.t. $\mathbf{g_c}$ gives $\mathbf{g_c} = (\mathbf{h_s}+\mathbf{h_t})/3$ at the minimum, and the regularizer reduces to $\frac{1}{3}(||\mathbf{h_s}||^2+||\mathbf{h_t}||^2+||\mathbf{h_s}-\mathbf{h_t}||^2)$. Thus, EA can be thought to be minimizing the sum of empirical source error on $\mathbf{h_s}$, empirical target error on $\mathbf{h_t}$ and this regularizer. The cost function $\mathscr{Q}_{EA}(\mathbf{h_1},\mathbf{h_2})$ can now be written as:

$$\begin{aligned} &\alpha\hat{\varepsilon}_s(\mathbf{h_1})+(1-\alpha)\hat{\varepsilon}_t(\mathbf{h_2})+\lambda_1||\mathbf{h_1}||^2+\lambda_2||\mathbf{h_2}||^2+\lambda||\mathbf{h_1}-\mathbf{h_2}||^2 \\ &\text{and} \quad (\mathbf{h_s},\mathbf{h_t}) = \underset{\mathbf{h_1},\mathbf{h_2}}{\arg\min}\, \mathscr{Q}_{EA} \end{aligned} \tag{3.2}$$

The EA algorithm minimizes this cost function over $\mathbf{h_1}$ and $\mathbf{h_2}$ jointly to obtain $\mathbf{h_s}$ and $\mathbf{h_t}$. The EA++ algorithm uses target unlabeled data, and encourages $\mathbf{h_s}$ and $\mathbf{h_t}$ to agree on unlabeled samples (Eq. 3.1). This can be thought of as having an additional regularizer of the form $\sum_{i\in U_t}(\mathbf{h_s}(x_i)-\mathbf{h_t}(x_i))^2$ in the cost function. The cost function for EA++ (denoted as $\mathscr{Q}_{++}(\mathbf{h_1},\mathbf{h_2})$) can then be written as:

$$\begin{aligned} &\alpha\hat{\varepsilon}_s(\mathbf{h_1})+(1-\alpha)\hat{\varepsilon}_t(\mathbf{h_2})+\lambda_1||\mathbf{h_1}||^2+\lambda_2||\mathbf{h_2}||^2+\lambda||\mathbf{h_1}-\mathbf{h_2}||^2 \\ &\qquad +\lambda_u\sum_{i\in U_t}(\mathbf{h_1}(x_i)-\mathbf{h_2}(x_i))^2 \end{aligned} \tag{3.3}$$

Both EA and EA++ give equal weights to source and target empirical errors, so $\alpha$ turns out to be 0.5. We use hyperparameters $\lambda_1$, $\lambda_2$, $\lambda$, and $\lambda_u$ in the cost functions to make them more general. However, as explained earlier, EA implicitly sets all these hyperparameters $(\lambda_1, \lambda_2, \lambda)$ to the same value (which will be $0.5(\frac{1}{3}) = \frac{1}{6}$ in our case, since the weights in the entire cost function are multiplied by $\alpha = 0.5$). The hyperparameter for unlabeled data $(\lambda_u)$ is 0.5 in EA++. We assume that the loss $L(y, \mathbf{h}.\mathbf{x})$ is bounded by 1 for the zero hypothesis $\mathbf{h} = \mathbf{0}$. This is true for many popular loss functions including square loss and hinge loss when $y \in \{-1, +1\}$. One possible way (Rosenberg and Bartlett, 2007) of defining the hypotheses classes is to substitute trivial hypotheses $\mathbf{h_1} = \mathbf{h_2} = \mathbf{0}$ in both the cost functions which makes all regularizers and coregularizers equal to zero and thus, bounds the cost functions $\mathscr{Q}_{EA}$ and $\mathscr{Q}_{++}$. This gives us $\mathscr{Q}_{EA}(\mathbf{0},\mathbf{0}) \leq 1$ and $\mathscr{Q}_{++}(\mathbf{0},\mathbf{0}) \leq 1$ since $\hat{\varepsilon}_s(\mathbf{0}), \hat{\varepsilon}_t(\mathbf{0}) \leq 1$. Without loss of generality, we also assume that final source and target hypotheses can only reduce the cost function as compared to the zero hypotheses. Hence, the final hypothesis pair $(\mathbf{h_s}, \mathbf{h_t})$ that minimizes the cost functions is contained in the following paired hypothesis classes for EA and EA++,

$$\begin{aligned} \mathscr{H} &:= \{(\mathbf{h_1},\mathbf{h_2}) : \lambda_1 ||\mathbf{h_1}||^2 + \lambda_2 ||\mathbf{h_2}||^2 + \lambda ||\mathbf{h_1} - \mathbf{h_2}||^2 \leq 1\} \\ \mathscr{H}_{++} &:= \{(\mathbf{h_1},\mathbf{h_2}) : \lambda_1 ||\mathbf{h_1}||^2 + \lambda_2 ||\mathbf{h_2}||^2 + \lambda ||\mathbf{h_1} - \mathbf{h_2}||^2 \\ &\quad + \lambda_u \sum_{i \in U_t} (\mathbf{h_1}(x_i) - \mathbf{h_2}(x_i))^2 \leq 1\} \end{aligned} \tag{3.4}$$

The source hypothesis class for EA is the set of all $h_1$ such that the pair $(h_1, h_2)$ is in $\mathscr{H}$. Similarly, the target hypothesis class for EA is the set of all $h_2$ such that the pair $(h_1, h_2)$ is in $\mathscr{H}$. Consequently, the source and target hypothesis classes for EA can be defined as:

$$\begin{aligned} \mathscr{J}^s_{EA} &:= \{\mathbf{h_1} : \mathscr{X} \mapsto \mathbb{R}, (\mathbf{h_1},\mathbf{h_2}) \in \mathscr{H}\} \qquad \text{and} \\ \mathscr{J}^t_{EA} &:= \{\mathbf{h_2} : \mathscr{X} \mapsto \mathbb{R}, (\mathbf{h_1},\mathbf{h_2}) \in \mathscr{H}\} \end{aligned} \tag{3.5}$$

Similarly, the source and target hypothesis classes for EA++ are defined as:

$$\begin{aligned} \mathscr{J}^s_{++} &:= \{\mathbf{h_1} : \mathscr{X} \mapsto \mathbb{R}, (\mathbf{h_1},\mathbf{h_2}) \in \mathscr{H}_{++}\} \qquad \text{and} \\ \mathscr{J}^t_{++} &:= \{\mathbf{h_2} : \mathscr{X} \mapsto \mathbb{R}, (\mathbf{h_1},\mathbf{h_2}) \in \mathscr{H}_{++}\} \end{aligned} \tag{3.6}$$

Furthermore, we assume that our hypothesis class is comprised of real-valued functions over an RKHS with reproducing kernel $k(\cdot,\cdot), k : \mathscr{X} \times \mathscr{X} \mapsto \mathbb{R}$. Let us define the kernel matrix and partition it corresponding to source labeled, target labeled and target unlabeled data as shown below:

$$K = \begin{pmatrix} A_{s\times s} & C_{s\times t} & D_{s\times u} \\ C'_{t\times s} & B_{t\times t} & E_{t\times u} \\ D'_{u\times s} & E'_{u\times t} & F_{u\times u} \end{pmatrix}, \tag{3.7}$$

where 's', 't' and 'u' indicate terms corresponding to source labeled, target labeled and target unlabeled, respectively.

### 3.3.2 Relate empirical and expected error (for both source and target)

Having defined the hypothesis classes, we now proceed to obtain generalization bounds for EA and EA++. We have the following standard generalization bound based on the Rademacher complexity of a hypothesis class (Rosenberg and Bartlett, 2007).

**Theorem 3.3.1** *Suppose the uniform Lipschitz condition holds for $L : \mathscr{Y}^2 \to [0,1]$, i.e., $|L(\hat{y_1},y) - L(\hat{y_2},y)| \le M|\hat{y_1} - \hat{y_2}|$, where $y, \hat{y_1}, \hat{y_2} \in \mathscr{Y}$ and $\hat{y_1} \ne \hat{y_2}$. Then for any $\delta \in (0,1)$ and for m samples $(X_1,Y_1),(X_2,Y_2),\ldots,(X_m,Y_m)$ drawn i.i.d. from distribution $\mathscr{D}$, we have with probability at least $(1-\delta)$ over random draws of samples,*

$$\varepsilon(f) \le \hat{\varepsilon}(f) + 2M\hat{R}_m(\mathscr{F}) + \frac{1}{\sqrt{m}}(2 + 3\sqrt{ln(2/\delta)/2}).$$

*where $f \in \mathscr{F}$ is the class of functions mapping $\mathscr{X} \mapsto \mathscr{Y}$, and $\hat{R}_m(\mathscr{F})$ is the empirical Rademacher complexity of $\mathscr{F}$ defined as $\hat{R}_m(\mathscr{F}) := E_\sigma[\sup_{f\in\mathscr{F}} |\frac{2}{m}\sum_{i=1}^m \sigma_i h_2(x_i)|]$.*

If we can bound the complexity of hypothesis classes $\mathscr{J}^s_{EA}$ and $\mathscr{J}^t_{EA}$, we will have a uniform convergence bound on the difference of expected and empirical errors ($|\varepsilon_t(h) - \hat{\varepsilon}_t(h)|$ and $|\varepsilon_s(h) - \hat{\varepsilon}_s(h)|$) using Theorem 3.3.1. However, in domain adaptation setting we are also interested in the bounds that relate expected target error to total empirical error on source and target samples. The following sections aim to achieve this goal.

### 3.3.3 Relate source expected error and target expected error

The following theorem provides a bound on the difference of expected target error and expected source error. The bound is in terms of $\eta_s := \varepsilon_s(f_s, f_t)$, $\nu_s := \varepsilon_s(h^*_t, f_t)$ and $\nu_t := \varepsilon_t(h^*_t, f_t)$, where $f_s$ and $f_t$ are the source and target labeling functions, and $h^*_t$ is the optimal target hypothesis in target hypothesis class. It also uses $d_{\mathscr{H}\Delta\mathscr{H}}(\mathscr{D}_s, \mathscr{D}_t)-$

distance (Blitzer et al., 2007a), which is defined as $\sup_{h_1,h_2 \in \mathscr{H}} 2|\varepsilon_s(h_1,h_2) - \varepsilon_t(h_1,h_2)|$. The $d_{\mathscr{H}\Delta\mathscr{H}}-$distance measures the distance between two distribution using a hypothesis class-specific distance measure. If the two domains are close to each other, $\eta_s$ and $d_{\mathscr{H}\Delta\mathscr{H}}(\mathscr{D}_s, \mathscr{D}_t)$ are expected to be small. On the contrary, if the domains are far apart, these terms will be big and the use of extra source samples may not help in learning a better target hypothesis. These two terms also represent the notion of adaptability in our case.

**Theorem 3.3.2** *Suppose the loss function is M-Lipschitz as defined in Theorem 3.3.1, and obeys triangle inequality. For any two source and target hypotheses $h_s, h_t$ (which belong to different hypotheses classes), we have*

$$\varepsilon_t(h_t, f_t) - \varepsilon_s(h_s, f_s) \leq M||h_t - h_s||E_s\left[\sqrt{k(x,x)}\right] + \frac{1}{2}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s, D_t) + \eta_s + \nu_s + \nu_t$$

*where $\mathscr{H}_t$ is the target hypothesis class, and $k(\cdot,\cdot)$ is the reproducing kernel for the RKHS. $\eta_s$, $\nu_s$, and $\nu_t$ are defined as above.*

**Proof.** Please see Appendix A.1. ■

### 3.3.4 Relate target expected error with source and target empirical errors

EA and EA++ learn source and target hypotheses jointly. So the empirical error in one domain is expected to have its effect on the generalization error in the other domain. In this section, we aim to bound the target expected error in terms of source and target empirical errors. The following theorem achieves this goal.

**Theorem 3.3.3** *Under the assumptions and definitions used in Theorem 3.3.1 and Theorem 3.3.2, with probability at least $1-\delta$ we have*

$$\begin{aligned}\varepsilon_t(h_t, f_t) \leq\ & \frac{1}{2}(\hat{\varepsilon}_s(h_s, f_s) + \hat{\varepsilon}_t(h_t, f_t)) + \frac{1}{2}(2M\hat{R}_m(\mathscr{H}_s) + 2M\hat{R}_m(\mathscr{H}_t)) \\ & + \frac{1}{2}\left(\frac{1}{\sqrt{l_s}} + \frac{1}{\sqrt{l_t}}\right)(2 + 3\sqrt{ln(2/\delta)/2}) \\ & + \frac{1}{2}M||h_t - h_s||E_s\left[\sqrt{k(x,x)}\right] + \frac{1}{4}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s, D_t) + \frac{1}{2}(\eta_s + \nu_s + \nu_t)\end{aligned}$$

*for any $h_s$ and $h_t$. $\mathscr{H}_s$ and $\mathscr{H}_t$ are the source hypothesis class and the target hypothesis class, respectively.*

**Proof.** We first use Theorem 3.3.1 to bound $(\varepsilon_t(h_t) - \hat{\varepsilon}_t(h_t))$ and $(\varepsilon_s(h_s) - \hat{\varepsilon}_s(h_s))$. The above theorem directly follows by combining these two bounds and Theorem 3.3.2. ■

This bound provides a better understanding of how the target expected error is governed by both source and target empirical errors, and hypotheses class complexities. This behavior is expected since both EA and EA++ learn source and target hypotheses jointly. We also note that the bound in Theorem 3.3.3 depends on $||h_s - h_t||$, which apparently might give an impression that the best possible thing to do is to make source and target hypotheses equal. However, due to joint learning of source and target hypotheses (by optimizing the cost function of Eq. 3.2), making the source and target hypotheses close will increase the source empirical error, thus loosening the bound of Theorem 3.3.3. Noticing that $||h_s - h_t||^2 \leq \frac{1}{\lambda}$ for both EA and EA++, the bound can be made independent of $||h_s - h_t||$ although with a sacrifice on the tightness. We note that Theorem 3.3.1 can also be used to bound the target generalization error of EA and EA++ in terms of only target empirical error. However, if the number of labeled target samples is extremely low, this bound can be loose due to inverse dependency on number of target samples. Theorem 3.3.3 bounds the target expected error using the averages of empirical errors, Rademacher complexities, and sample dependent terms. If the domains are reasonably close and the number of labeled source samples is much higher than target samples, this can provide a tighter bound compared to Theorem 3.3.1.

Finally, we need the Rademacher complexities of source and target hypothesis classes (for both EA and EA++) to be able to use Theorem 3.3.3, which are provided in the next sections.

### 3.3.5 Bound the complexity of EA and EA++ hypothesis classes

The following two theorems bound the Rademacher complexity of the target hypothesis classes for EA and EA++, respectively.

**Theorem 3.3.4** *For the hypothesis class* $\mathscr{J}^t_{EA}$ *defined in Eq. 3.5 we have,* $\frac{1}{\sqrt[4]{2}} \frac{2C^t_{EA}}{l_t} \leq \hat{R}_m(\mathscr{J}^t_{EA}) \leq \frac{2C^t_{EA}}{l_t}$ *where,* $\hat{R}_m(\mathscr{J}^t_{EA}) = E_\sigma \sup_{h_2 \in \mathscr{J}^t_{EA}} |\sum_i \sigma_i h_2(x)|$, *and*

$$(C^t_{EA})^2 = \left( \frac{1}{\lambda_2 + \left( \frac{1}{\lambda_1} + \frac{1}{\lambda} \right)^{-1}} \right) tr(B)$$

*and $B$ is the kernel submatrix defined as in Eq. 3.7.*

**Proof.** Please see Appendix A.2. ■

The complexity of target class decreases with an increase in the values of hyperparameters. It decreases more rapidly with change in $\lambda_2$ compared to $\lambda$ and $\lambda_1$, which is also expected since $\lambda_2$ is the hyperparameter directly influencing the target hypothesis. The kernel block submatrix corresponding to source samples does not appear in the bound. This result in conjunction with Theorem 3.3.1 gives a bound on the target generalization error.

To be able to use the bound of Theorem 3.3.3, we need the Rademacher complexity of the source hypothesis class. Due to the symmetry of paired hypothesis class (Eq. 3.4) in $h_1$ and $h_2$ up to scalar parameters, the complexity of source hypothesis class can be similarly bounded by $\frac{1}{\sqrt[4]{2}} \frac{2C^s_{EA}}{l_s} \le \hat{R}_m(\mathscr{J}^s_{EA}) \le \frac{2C^s_{EA}}{l_s}$, where $(C^s_{EA})^2 = \left( \frac{1}{\lambda_1 + \left( \frac{1}{\lambda_2} + \frac{1}{\lambda} \right)^{-1}} \right) tr(A)$, and $A$ is the kernel block submatrix corresponding to source samples.

**Theorem 3.3.5** *For the hypothesis class $\mathscr{J}^t_{++}$ defined in Eq. 3.6 we have, $\frac{1}{\sqrt[4]{2}} \frac{2C^t_{++}}{l_t} \le \hat{R}_m(\mathscr{J}^t_{++}) \le \frac{2C^t_{++}}{l_t}$ where, $\hat{R}_m(\mathscr{J}^t_{++}) = E_\sigma \sup_{h_2 \in \mathscr{J}^t_{++}} |\sum_i \sigma_i h_2(x)|$ and*

$$(C^t_{++})^2 = \left( \frac{1}{\lambda_2 + \left( \frac{1}{\lambda_1} + \frac{1}{\lambda} \right)^{-1}} \right) tr(B) - \lambda_u \left( \frac{\lambda_1}{\lambda\lambda_1 + \lambda\lambda_2 + \lambda_1\lambda_2} \right)^2 tr\left( E(I + kF)^{-1} E' \right)$$

*where $k = \frac{\lambda_u(\lambda_1+\lambda_2)}{\lambda\lambda_1+\lambda\lambda_2+\lambda_1\lambda_2}$.*

**Proof.** Please see Appendix A.3. ■

The second term in $(C^t_{++})^2$ is always positive since the trace of a positive definite matrix is positive. So, the unlabeled data results in a reduction of complexity over the labeled data case (Theorem 3.3.4). The *trace* term in the reduction can also be written as $\sum_i ||E_i||^2_{(I+kF)^{-1}}$, where $E_i$ is the $i$'th column of matrix $E$ and $||\cdot||^2_Z$ is the norm induced by a positive definite matrix $Z$. Since $E_i$ is the vector representing the inner product of $i$'th target sample with all unlabeled samples, this means that the reduction in complexity is proportional to the *similarity* between target unlabeled samples and target labeled samples. This result in conjunction with Theorem 3.3.1 gives a bound on the target generalization error in terms of target empirical error.

To be able to use the bound of Theorem 3.3.3, we need the Rademacher complexity of source hypothesis class too. Again, as in case of EA, using the symmetry of paired hypothesis class $\mathscr{H}_{++}$ (Eq. 3.4) in $h_1$ and $h_2$ up to scalar parameters, the complexity of source hypothesis class can be similarly bounded by $\frac{1}{\sqrt[4]{2}} \frac{2C^s_{++}}{l_s} \leq \hat{R}_m(\mathscr{J}^s_{++}) \leq \frac{2C^s_{++}}{l_s}$, where $(C^s_{++})^2 = \left( \frac{1}{\lambda_1 + \left( \frac{1}{\lambda_2} + \frac{1}{\lambda} \right)^{-1}} \right) tr(A) - \lambda_u \left( \frac{\lambda_2}{\lambda\lambda_1 + \lambda\lambda_2 + \lambda_1\lambda_2} \right)^2 tr\left( D(I + kF)^{-1} D' \right)$, and $k$ is defined similarly as in Theorem 3.3.5. The *trace* term can again be interpreted as before, which implies that the reduction in source class complexity is proportional to the *similarity* between source labeled samples and target unlabeled samples.

## 3.4 Experiments

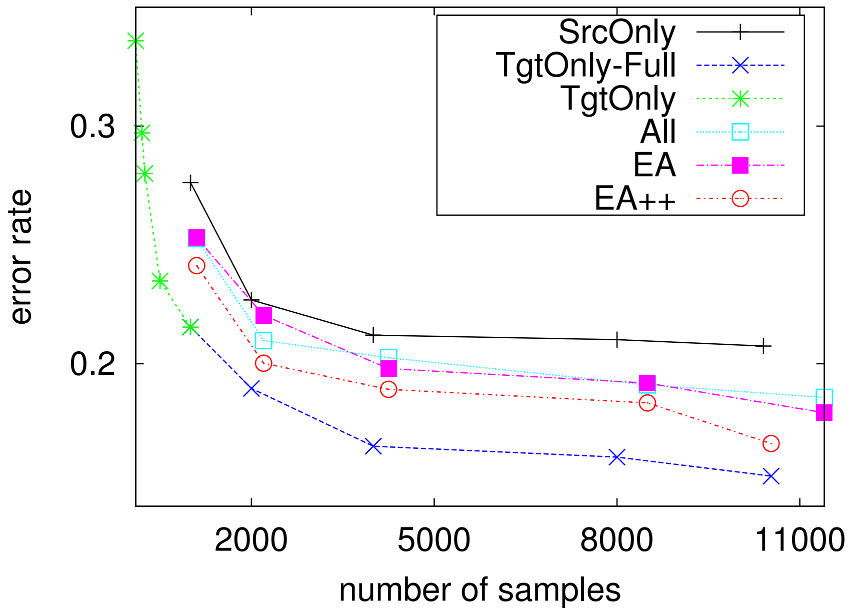
### 3.4.1 Results on sentiment classification task

We follow experimental setups similar to Daumé III (2007) but report our empirical results for the task of sentiment classification using the SENTIMENT data provided by Blitzer et al. (2007b). The task of sentiment classification is a binary classification task which corresponds to classifying a review as positive or negative for user reviews of eight product types (apparel, books, DVD, electronics, kitchen, music, video, and other) collected from *Amazon.com*. We quantify the domain divergences in terms of the $\mathscr{A}$-distance (Ben-David et al., 2006) which is computed (Rai et al., 2010) from finite samples of source and target domain using the *proxy* $\mathscr{A}$-distance (Ben-David et al., 2006). For our experiments, we consider the following domain-pairs: (a) *Dvd→Books* (*proxy* $\mathscr{A}$-distance=0.7616) and, (b) *Kitchen→Apparel* (*proxy* $\mathscr{A}$-distance=0.0459). As in Daumé III (2007), we use an averaged perceptron classifier from the Megam framework (implementation due to Daumé III (2004)) for all the aforementioned tasks. The training sample size varies from $1k$ to $16k$. In all cases, the amount of unlabeled target data is equal to the total amount of labeled source and target data.

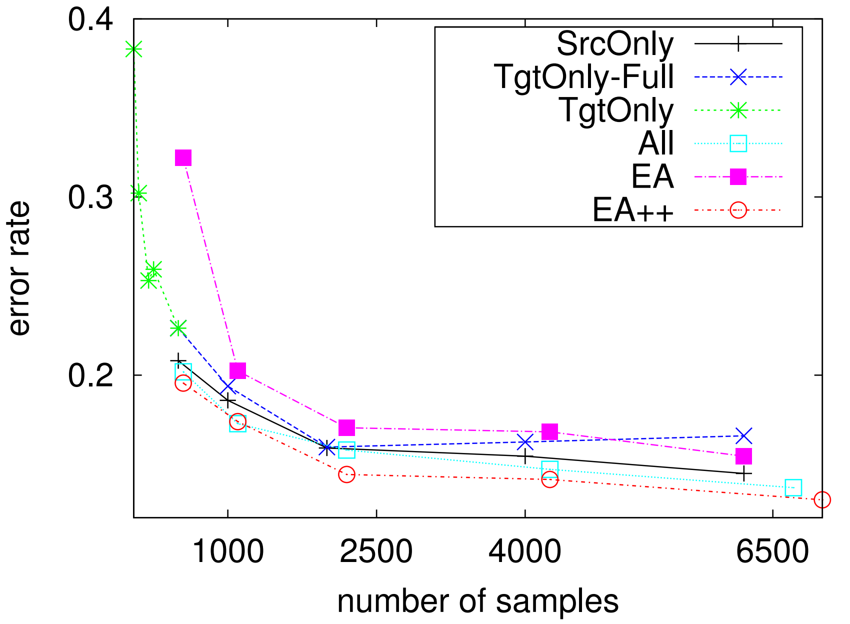
We compare the empirical performance of EA++ with the following baselines, namely, (a) SOURCEONLY (classifier trained on source labeled samples), (b) TARGETONLY-FULL (classifier trained on the same number of target labeled samples as the number of source labeled samples in SOURCEONLY), (c) TARGETONLY (classifier trained on small amount of target labeled samples, roughly one-tenth of the amount of source labeled samples in SOURCEONLY), (d) ALL (classifier trained on combined labeled samples of SOURCEONLY and TARGETONLY), (e) EA (classifier trained in *augmented feature space* on the same input training set as ALL), (f) EA++ (classifier trained in *augmented feature space* on the same input training set as EA and an equal amount of unlabeled *target* data). All these approaches were tested on the entire amount of available *target* test data.

Figure 3.3 presents the learning curves for (a) SOURCEONLY, (b) TARGETONLY-FULL, (c) TARGETONLY, (d) ALL, (e) EA, and (f) EA++ (EA with unlabeled data). The x-axis represents the number of training samples on which the predictor has been trained. At this point, we note that the number of training samples vary depending on the particular approach being used. For SOURCEONLY, TARGETONLY-FULL and TARGETONLY, it is just the corresponding number of labeled source or target samples, respectively. For ALL and EA, it is the summation of labeled source and target samples. For EA++, the $x$-value plotted denotes the amount of unlabeled target data used (in addition to an equal amount of source+target labeled data, as in ALL or EA). We plot this number for EA++, just to compare its improvement over EA when using an additional (and equal) amount of unlabeled target data. This accounts for the different $x$ values plotted for the different curves. In all cases, the y-axis denotes the error rate.

As can be seen, for both the cases, EA++ outperforms EASYADAPT. For *Dvd→Books*, the domains are far apart as denoted by a high *proxy* $\mathscr{A}$-distance. Hence, TARGETONLY-FULL achieves the best performance and EA++ almost catches up for large amounts of training data. For different number of sample points, EA++ gives relative improvements in the range of $4.36\% - 9.14\%$, as compared to EA. The domains KITCHEN and APPAREL can be considered to be reasonably close due to their low domain divergence. Hence, this domain pair is more amenable for domain adaptation as is demonstrated by the fact that the other approaches (SOURCEONLY, TARGETONLY, ALL) perform better or at least as good as TARGETONLY-FULL. However, as earlier, EA++ once again outperforms all



(a)



(b)

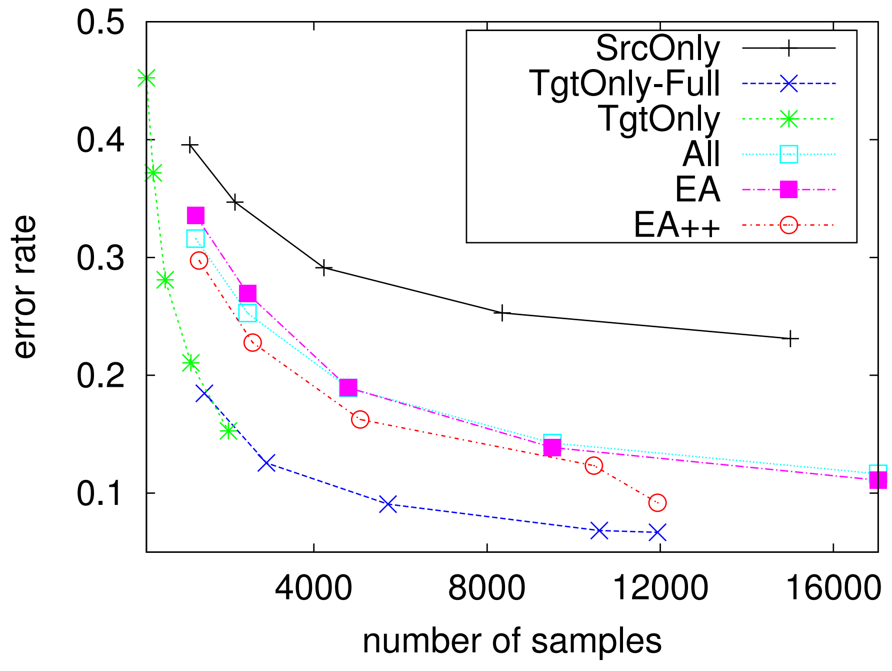
**Figure 3.3**. Test accuracy of SOURCEONLY, TARGETONLY-FULL, TARGETONLY, ALL, EA, EA++ (with unlabeled data) for, (a) *Dvd→Books* (*proxy $\mathscr{A}$*-distance=0.7616), (b) *Kitchen→Apparel* (*proxy $\mathscr{A}$*-distance=0.0459) these approaches including TARGETONLY-FULL. Due to the closeness of the two domains, additional unlabeled data in EA++ helps it in outperforming TARGETONLY-FULL. At this point, we also note that EA performs poorly for some cases, which corroborates with prior experimental results (Daumé III, 2007). For this dataset, EA++ yields relative improvements in the range of $14.08\% - 39.29\%$ over EA for different number of sample points experimented with. Similar trends were observed for other tasks and datasets (refer to Figure 3 of Daumé III et al. (2010)).

### 3.4.2 Results on sequence labeling tasks

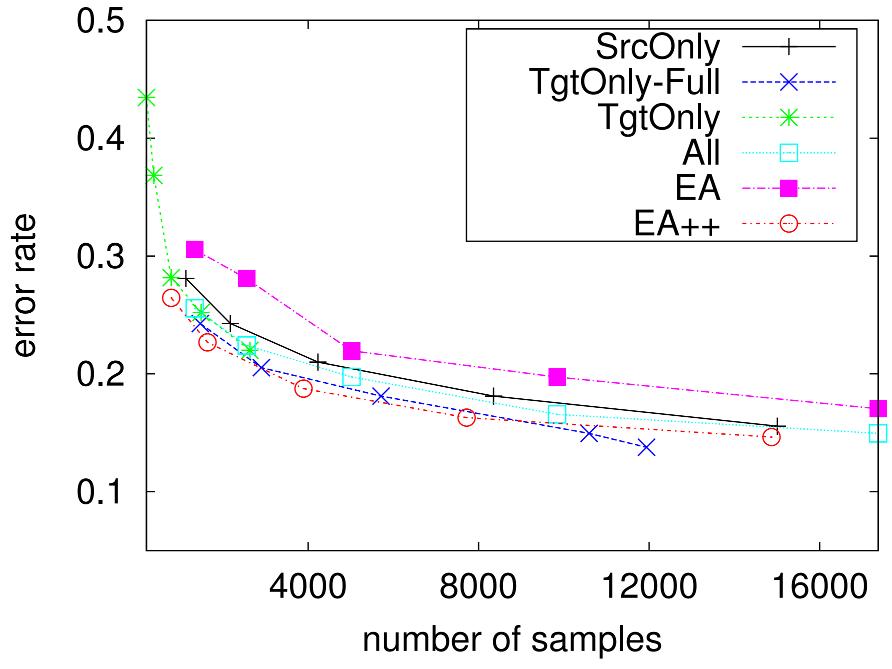
In this section, we demonstrate the empirical performance of EA++ on some additional tasks and datasets. We use the same tasks and datasets as in Daumé III (2007) and perform two sequence labelling tasks (a) named-entity-recognition (NER), and (b) part-of-speech-tagging (POS) on (a) PubMed-POS, and (b) Treebank-Brown.

- **PubMed-POS.** Introduced by Blitzer et al. (2006), this dataset consists of two domains. The WSJ portion of the Penn Treebank serves as the source domain and the PubMed abstracts serve as the target domain. The task is to perform part-of-speech tagging on unlabeled PubMed abstracts with a classifier trained on labeled WSJ and PubMed data (see Figure 3.4(a)).
- **Treebank-Brown.** Treebank-Chunk data consists of the following domains: the standard WSJ domain (the same data as for CoNLL 2000), the ATIS switchboard domain and the Brown corpus. The Brown corpus consists of data combined from six subdomains. Treebank-Chunk is a shallow parsing task based on the data from the Penn Treebank. Treebank-Brown is identical to the Treebank-Chunk task. However, in Treebank-Brown we consider all of the Brown corpus to be a single domain (see Figure 3.4(b)).

All datasets use roughly the same feature set which are lexical information (words, stems, capitalization, prefixes and suffixes), membership on gazetteers, etc. As earlier, we use an averaged perceptron classifier from the Megam framework (implementation due to (Daumé III, 2004)) for all the aforementioned tasks. The training sample size varies from $1k$ to $16k$ and in all cases the amount of unlabeled target data was the same as total amount of labeled source+target data.



(a)



(b)

**Figure 3.4**. Test accuracy of SOURCEONLY, TARGETONLY-FULL, TARGETONLY, ALL, EA, EA++ (with unlabeled data) for, (a) *PubMed-POS*, (b) *Treebank-Brown*

As earlier, we compare the empirical performance of EA++ with other aforementioned baselines. Figure 3.4(a) presents the learning curves for (a) SOURCEONLY, (b) TARGETONLY-FULL, (c) TARGETONLY, (d) ALL, (e) EA, and (f) EA++ (EA with unlabeled data). The x-axis represents the number of training samples on which the predictor has been trained. As mentioned earlier, this number varies for the different approaches. The y-axis denotes the error rate due to each setting. As can be seen, the labeled and unlabeled case start together, but with increase in number of samples their gap increases with the unlabeled case resulting in much lower error as compared to the labeled case. Similar trends were observed in other data sets as can be seen in Figure 3.4(b). It should be noted that EA performs worse than all other approaches for Treebank-Brown, which is a result consistent with the findings in Daumé III (2007). As noted in Daumé III (2007), these are mostly the tasks in which SOURCEONLY outperforms TARGETONLY, which implies that source and target domains may not be that different. Due to similarity of the domains, large amount of source data outperforms a small amount of target data and blowing-up the feature space may not help much. EA++ still manages to outperform all other approaches (except TARGETONLY-FULL in a few cases) due to the large amount of unlabeled data used in it.

## 3.5 Summary

We proposed a semisupervised extension to an existing domain adaptation technique (EA). Our approach, EA++, leveraged unlabeled data to improve the performance of EA. With this extension, EA++ applies to both *fully supervised* and *semisupervised* domain adaptation settings. We have formulated EA and EA++ in terms of coregularization, an idea that originated in the context of multiview learning (Rosenberg and Bartlett, 2007, Sindhwani and Rosenberg, 2008). Our proposed formulation also bears resemblance to existing work (Blum and Mitchell, 1998) in multiview semisupervised (SSL) literature which has been studied extensively in Balcan and Blum (2005, 2010), Sridharan and Kakade (2008). The difference being, while in *multiview SSL* one would try to make the different hypotheses learned from different views agree on unlabeled data, in *semisupervised domain adaptation* the aim is to make the different hypotheses learned from different distributions agree on unlabeled data. Using our formulation, we have presented theoretical analysis of the superior performance of EA++ as compared to EA. Our empirical results further confirm the theoretical findings. EA++ can also be extended to the multiple source settings.

# CHAPTER 4

# ONLINE TRANSFER

In the realm of transfer learning, multitask learning (Caruana, 1997, Heskes, 2000) refers to the setting when the learner has access to data from multiple related learning tasks. The goal is to *jointly* learn the *related tasks* so as to improve generalization across all tasks. This is especially important when there is a scarcity of labeled data for one or more tasks. In this work, we consider an online multitask learning setting with *linear classifiers*. In our setting, the learner receives examples from $K$ different tasks (in an interleaved fashion), and learns the $K$ weight vectors as well as a $K \times K$ task-relatedness matrix, simultaneously.

A precise characterization of task relatedness is of extreme importance as it facilitates sharing of relevant information across the multiple related tasks. In the batch setting, one can enforce task relatedness via *structural assumptions* on the weight vectors of the tasks; for example, a shared prior distribution (Heskes, 2000), cluster assumption (Xue et al., 2007b), subspace assumption (Evgeniou et al., 2005, Rai and Daumé III, 2010), task hierarchies (Daumé III, 2009), adopting a Gaussian process framework (Bonilla et al., 2007), and so on. An alternative (Cavallanti et al., 2008) is to *explicitly* encode the task relationships in a matrix which is assumed to be known beforehand. However, an *a priori* assumption on the nature or extent of relatedness can often be restrictive. Furthermore, in the online setting, intertask relatedness could potentially vary over time making it even more difficult to be elicited. A favorable choice is to learn the task relationships automatically from the data. However, in a truly online setting where the weight vectors are constantly changing with each incoming example, even this can be quite difficult to achieve (as we discuss later in section 4.2.2). Therefore, we need to devise ways for *online learning* of task relationships, adaptively from the data.

In this work, we propose a framework which allows simultaneous learning of the weight vectors of multiple tasks as well as the task relationship matrix in an *online* setting. In particular, the problem of online learning the task relationship matrix can be framed (Tsuda et al., 2005) as a Bregman divergence minimization problem for positive definite matrices (which is true since the task relationship matrix is defined as a task covariance matrix in Eq. (4.5); also, see Eq. (6) of Zhang and Yeung (2010)). One of the implicit reasons to learn the task relationship matrix, is to employ intertask similarity to quantify the informativeness of an incoming sample that belongs to a particular task. In subsequent sections, we show how the learned task-relationship matrix can be exploited to select the most informative examples in an online multitask active learning scenario.

Our work assumes the setup of Abernethy et al. (2007), Cavallanti et al. (2008) where instances (for different tasks) arrive one-at-a-time, and the sequence of examples and the corresponding task index (the task which an incoming example belongs to) is chosen adversarially. In the next section, we briefly describe this setting referring to the prior work that assumes a fixed task relationship matrix. Thereafter, we present our proposed approaches for online multitask learning with adaptive task relationships.

Multitask learning has received considerable attention in machine learning literature. Most of the existing work primarily differ in their assumptions of task relatedness. In this section, we refer to a small subset of the existing literature that relates to *online* multitask learning.

The online multitask learning problem was first addressed in Dekel et al. (2006). The authors assume a very general setting where the tasks were related by a global loss function and the goal was to reduce the cumulative loss (for all tasks involved) over rounds of the online algorithm. The hope was that the nature of the global loss function would dictate the error correction mechanism of the algorithm and a family of algorithms was proposed for a wide variety of loss functions. We contend that while combining losses via global loss functions is a good way to formulate cost function, it does not leverage the task relationship information from the available data.

On a similar but somewhat different note, Abernethy et al. (2007) and Agarwal et al. (2008) consider an alternate formulation of online multitask learning under the traditional expert advice model. In their regret-minimization framework, the notion of *task relatedness* was captured in terms of experts with the hope that experts which perform well on one task should also do well on other related tasks. The goal was to find a small subset of experts which perform well throughout the learning process. This, in a way, is analogous to finding a low-dimensional common representation for the multiple related tasks (Evgeniou et al., 2005, Rai and Daumé III, 2010). Our setting, on the other hand, is conceptually simpler and much easier to implement in practice. Another work (Lugosi et al., 2009) along similar lines extended the notion of experts to the set of decisions the forecaster is allowed to take. As earlier, the idea is to impose task relatedness by constraining the different tasks to choose their decision from a small subset.

Apart from minimizing the cumulative loss and regrets, reducing mistake bounds for the online multitask learning has been considered in Cavallanti et al. (2008). Our work is based on this setting and we will discuss it in detail in section 4.1. However, we note that in contrast to our approach, Cavallanti et al. (2008) assumes a fixed task relationship matrix.

## 4.1 Background

We start with the perceptron based online multitask learning setting described in Cavallanti et al. (2008), henceforth referred to as CMTL. In their setting, the learner proceeds in rounds by observing a sequence of examples, each belonging to some task from a predefined set of $K$ tasks. The goal of the learner is to learn $K$ perceptron weight vectors, one for each task. In round $t$, the learner receives a pair $(x_t, i_t)$, where $x_t \in \mathbb{R}^d$ is the example and $i_t \in \{1,\ldots,K\}$ is the corresponding task-id. The learner outputs a binary prediction $\hat{y}_t \in \{-1,1\}$ and then receives the true label $y_t \in \{-1,1\}$ for this example. The observed task sequence is adversarial. We follow the notation of (Cavallanti et al., 2008) and represent the incoming example at round $t$ as a compound vector $\phi_t = (0,\ldots,0,x_{i_t},0,\ldots,0) \in \mathbb{R}^{Kd}$. Similarly, the weights of $K$ perceptrons are stored in a compound weight vector $w_s^T = (w_{1,s}^T,\ldots,w_{K,s}^T) \in \mathbb{R}^{Kd}$, where $w_{j,s} \in \mathbb{R}^d\ \forall j \in \{1,\ldots,K\}$, and $s$ denotes the number of updates so far.

In CMTL's proposed multitask perceptron, the $K$ weight vectors are updated *simultaneously* using rules that are derived from a predefined (fixed) task relationship matrix which they call the *interaction matrix* (defined below). We note that in this work we use the terms "task relationship matrix" and "interaction matrix" interchangeably. The entries of the interaction matrix define the learning rates ($\gamma$) to be used in the updates rules for each of the $K$ perceptron weights. Using the following fixed task *interaction matrix*,

$$A^{-1} = \frac{1}{K+1}\begin{bmatrix} 2 & 1 & \dots & 1 \\ 1 & 2 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 2 \end{bmatrix}$$

the update rules become:

$$w_s = w_{s-1} + y_t (A \otimes I_d)^{-1} \phi_t \tag{4.1}$$

where $\otimes$ denotes the $Kd \times Kd$ Kronecker product defined as:

$$A \otimes I_d = \begin{pmatrix} a_{11}I_d & \cdots & a_{1K}I_d \\ \dots & \dots & \dots \\ a_{K1}I_d & \cdots & a_{KK}I_d \end{pmatrix}$$

For individual tasks $j$, Eq. (4.1) reduces to:

$$w_{j,s} = w_{j,s-1} + y_t A^{-1}_{j,i_t} x_t \tag{4.2}$$

From the above $K \times K$ *interaction matrix* ($A^{-1}$), it follows that for $j = i_t, \gamma = \frac{2}{K+1}$, whereas for tasks $j \neq i_t$, $\gamma = \frac{1}{K+1}$, where $\gamma$ is the learning rate of the weight vectors. This update scheme is reasonable since it basically does a fixed, constant update for the current task $i_t$ but at the same time also does "half-updates" for the remaining $K-1$ tasks, since they are expected to be related to the current task.

Following Cesa-Bianchi and Lugosi (2006), the CMTL algorithm can be seen as optimizing the following regularized loss function:

$$\underset{w \in \mathbb{R}^{Kd}}{\arg\min} \left[ \frac{1}{2} w^T (A \otimes I_d) w + \sum_{1}^{t} l_t(w) \right] \tag{4.3}$$

where $l_t(w) = [1 - y_t w^T \phi_t]_+$ denotes the hinge loss of the weight vector $w$ at time $t$. The $Kd \times Kd$ matrix $(A \otimes I_d)$ in the first term above coregularizes the compound weight vector $w$ so as to bring the individual task weight vectors closer to each other. When $A$ is the $K \times K$ identity matrix, CMTL degenerates to $K$ Independent Perceptron Learning (IPL).

## 4.2 Online Task Relationship Learning

The CMTL approach assumes a fixed task interaction matrix which seems restrictive in many respects. First, one does not usually know the task relationships *a priori*. Second, the fixed task interaction matrix of CMTL assumes that all the tasks are positively correlated, which can again be an unreasonable assumption for many real-world multitask datasets that may consist of unrelated, or possibly even noisy or negatively correlated tasks. Therefore, a fixed interaction matrix may not always be the right choice since it may vary over time, especially, with an adversary. At this point, we note that the CMTL can conceivably accommodate negative correlation between tasks by hand-specifying negative weights in the task interaction matrix. However, this constitutes *a priori* assumptions on task relations, whereas the main thesis of our work is to learn these relationships from the data.

In this work, we propose to learn the task interaction matrix *adaptively* from the data, thereby letting the data itself dictate what the task relationships should look like instead of fixing them *a priori*. Since the success of learning the $K$ perceptron weight vectors hinges crucially on the task interaction matrix, the hope is that an adaptively learned task interaction matrix would lead to improved estimates of the weight vectors of all the tasks.

Following Crammer et al. (2006), we formulate our goal as an optimization problem in the online learning setting, as shown below. Formally, at round $t+1$, we solve the following:

$$\underset{w \in \mathbb{R}^{Kd}, A \succ 0}{\arg\min} \left[ D_w(w||w_s) + D_A(A||A_s) + \sum_1^t l_t(w) \right] \tag{4.4}$$

where $w_t$ and $A_t$ are the weight vector and the interaction matrix at the previous round $t$, and $D_w(.||.)$ and $D_A(.||.))$ denote Bregman divergences. The above cost function is inspired by the classical cost function formulations of online algorithms where the update of the weight vector balances between "conservativeness" and "correctiveness" (Cesa-Bianchi and Lugosi, 2006). It is easy to see that if we use the Mahalanobis divergence for $D_w(.||.)$, Eq. (4.4) reduces to the CMTL objective function of Eq. (4.3) (modulo the extra $D_A(.||.)$ term). However, our setting is different as follows: (1) the matrix $A$ is no longer a fixed matrix, and (2) we add a matrix regularization penalty (discussed later) over $A$ such that it stays close to the previous estimate of the interaction matrix akin to a *conservative update* strategy (recall that we have an online setting). Our proposed formulation yields the following objective function to be solved at each round of online learning:

$$\underset{w \in \mathbb{R}^{Kd}, A \succ 0}{\arg\min} \left[ \frac{1}{2} w^T A_{\otimes} w + D_A(A||A_t) + \sum_1^t l_t(w) \right] \tag{4.5}$$

where $A_{\otimes} = A \otimes I_d$. The optimization problem in Eq. (4.5) is defined jointly over both $w$

and $A$. It can be solved in an alternating fashion by solving for $w$ given $A$, and then solving for $A$ given $w$.

Our objective function is generic and the $D_A(.||.)$ term allows substituting any suitable divergence defined over positive definite matrices. We first define the general form of matrix divergence between two positive definite matrices:

$$D_\phi(X,Y) = \phi(X) - \phi(Y) + tr((X-Y)f(Y)^T)$$

where $X,Y$ are $n \times n$ matrices and $f(Y) = \nabla_Y \phi(Y)$. In addition, $\phi : S^n \to \mathbb{R}$ is a strictly convex, differentiable functions and $tr$ denotes the matrix trace.

In this work, we consider the following matrix divergences by substituting the appropriate function for $\phi$, as shown below:

1. **LogDet divergence:** When $\phi(X) = \phi_{LD}(X) = -\log|X|$, we obtain the LogDet divergence between two positive definite matrices $X$ and $Y$ defined as: $D_{\phi_{LD}}(X,Y) = tr(XY^{-1}) - \log|XY^{-1}| - n$.

2. **von-Neumann divergence:** When $\phi(X) = \phi_{VN}(X) = tr(X\log X - X)$, we obtain the von-Neumann divergence between two positive definite matrices $X$ and $Y$ defined as: $D_{\phi_{VN}}(X,Y) = tr(X\log X - Y\log Y - X + Y)$.

We show that the aforementioned divergence functions permit *online* update schemes for our task interaction matrix $A$. Furthermore, these divergence functions also ensure that our updates for $A$ preserve (Kulis et al., 2009, Tsuda et al., 2005) positive definiteness and unit trace.

### 4.2.1 Alternating optimization

We adopt an alternating optimization scheme to solve for $w$ and $A$. We undergo a small change in notation and note that $w$ and $A$ are updated only when a prediction mistake occurs. We denote the update index by $s$ and the rounds of the online algorithm by $t$, $(s \leq t)$. Fixing $A$ to $A_{s-1}$, it is easy to see that our updates for $w$ are exactly of the same form as the CMTL update rule defined in Eq. (4.2):

$$\begin{aligned} w_s &= w_{s-1} + y_t(A_{s-1} \otimes I_d)^{-1}\phi_t \\ w_{j,s} &= w_{j,s-1} + y_t A^{-1}_{s-1,(j,i_t)} x_t \end{aligned} \tag{4.6}$$

where $A^{-1}_{s-1,(j,i_t)}$ denotes the inverse of the $(j,i_t)^{th}$ element of $A_{s-1}$. Having solved for $w_s$, we treat it as fixed and solve for $A$. We consider both the matrix divergences mentioned earlier and derive the general expression for the update rules. We use the fact that $w_s^T(A \otimes I_d)w_s = tr(W_s A W_s^T)$, where $W_s$ is a $d \times K$ matrix obtained by column-wise reshaping the $Kd \times 1$ vector $w_s$. The $K$ columns of $W_s$ represent weight vectors of the $K$ tasks. With $w_s$ (and thus $W_s$) fixed, our objective function reduces to:

$$\underset{A \succ 0}{\arg\min} \left[ \frac{1}{2} tr(W_{s-1} A W_{s-1}^T) + D_A(A \| A_{s-1}) \right] \tag{4.7}$$

For both the cases, following (Tsuda et al., 2005), the update rule can be written as:

$$A_s = \arg \min_{A \succ 0} \left[ D_\phi(A, A_{s-1}) + \eta \frac{1}{2} tr(W_{s-1} A W_{s-1}^T) \right] \tag{4.8}$$

which has the solution:

$$A_s = f^{-1}\left( f(A_{s-1}) - \eta \, \mathbf{sym}(\nabla_A \frac{1}{2} tr(W_{s-1} A W_{s-1}^T)) \right) \tag{4.9}$$

where $f(A) = \nabla_A \phi(A)$, $f^{-1}$ is the inverse function of $f$, $\mathbf{sym}(X) = (X + X^T)/2$ and $\eta$ is the learning rate of the interaction matrix $A$. Next, we consider the specific cases when $\phi = \phi_{LD}$ (LogDet divergence) and $\phi = \phi_{VN}$ (von-Neumann divergence).

**LogDet divergence:** For the LogDet matrix divergence, $f(A) = \nabla_A \phi_{LD}(A) = -A^{-1}$ and $f^{-1}(B) = -B^{-1}$, which reduces Eq. (4.9) to the following update rule:

$$A_s = \left( A_{s-1}^{-1} + \eta \, \mathbf{sym}(W_{s-1}^T W_{s-1}) \right)^{-1} \tag{4.10}$$

It is easy to see that the above update equation maintains the positive definiteness of $A_s$. We refer to the LogDet matrix divergence based online algorithm for $A$ as OMTLLOG.

**von-Neumann divergence:** For the von-Neumann matrix divergence, $f(A) = \nabla_A \phi_{VN}(A) = \log(A)$ and $f^{-1}(B) = \exp(B)$, for which the update rule of Eq. (4.9) reduces to:

$$A_s = \exp\left( \log A_{s-1} - \eta \, \mathbf{sym}(W_{s-1}^T W_{s-1}) \right) \tag{4.11}$$

where exp and log are matrix exponential and matrix logarithm, respectively. Since $A_{s-1}$ is real symmetric, $\log A_{s-1}$ is also real symmetric. Hence, the term

$$\left( \log A_{s-1} - \eta \, \mathbf{sym}(W_{s-1}^T W_{s-1}) \right)$$

in Eq. (4.11) is a symmetric matrix and the "exp" operation maps this back into a symmetric positive definite matrix. Thus, the above update equation maintains the symmetric positive definiteness of $A_s$. We refer to the algorithm based on this online update rule for $A$ as OMTLVON.

It can be seen that the very nature of the derived equations (Eq. (4.6), Eq. (4.10) and Eq. (4.11)) suggests an online learning setting such that both $w$ and $A$ can be updated in an incremental fashion (refer to Algorithm 1).

In addition to the LogDet and von-Neumann divergences based update rules for $A$, we also propose using the covariance of task weight vectors as an alternate strategy. The intuition for a covariance-based update scheme stems from the observation that the covariance of task weight vectors is a natural way to estimate the intertask relationships. In fact, most of the literature on Gaussian Process based multitask learning (Bonilla et al., 2007, Daumé III, 2009) assume a Gaussian Process prior on the space of functions being learned and use the Gaussian Process covariance function to model task relatedness. This motivates us to use the task covariance matrix to model intertask relationships and we use a task covariance based update in our online multitask scenario. We refer to it as OMTLCOV which has the following update rule:

$$A_s = \mathrm{cov}(W_{s-1}) \tag{4.12}$$

where "cov" denotes a standard covariance operation over a matrix.

Finally, we consider a recent work (Zhang and Yeung, 2010) which showed that in the batch setting, the *optimal* task relationship matrix can be expressed as $A = \frac{(W^T W)^{\frac{1}{2}}}{tr((W^T W)^{\frac{1}{2}})}$ where $W$ is a $d \times K$ matrix whose $K$ columns consist of the weight vectors of each of the $K$ tasks. Note that the batch approach first estimates *all* $K$ weight vectors, before computing $A$, and the process is repeated in an alternating fashion until convergence. In contrast, the online setting updates the weight vector of one task at a time and has to update $A$ immediately after that. We nevertheless compare with this approach by updating $A$ everytime the weight vector of some task gets updated. We call it BATCHOPT and treat it as *one of our baselines*. BATCHOPT uses the following update rule:

$$A_s = \frac{(W_{s-1}^T W_{s-1})^{\frac{1}{2}}}{tr((W_{s-1}^T W_{s-1})^{\frac{1}{2}})} \tag{4.13}$$

**Algorithm 1** Online Task Relationship Learning

```
1: Input: Examples from $K$ tasks, Number of rounds
2: Output: $w$ and a positive definite $K \times K$ matrix $A$, learned after $T$ rounds;
3: Initialization: $A = \frac{1}{K} \times I_d$; $w_0 = 0$;
4: for $t = 1$ to $T$ do
5:    receive the pair $(x_t, i_t)$, $x_t \in \mathbb{R}^d$;
6:    construct $\phi_t \in \mathbb{R}^{Kd}$ from $x_t$;
7:    predict label $\hat{y}_t = SGN(w_{s-1}^T \phi_t) \in \{-1, +1\}$;
8:    receive true label $y_t \in \{-1, +1\}$;
9:    if $(y_t \neq \hat{y}_t)$ then
10:      /* update $w_s$ and $A_s$ */
11:      for $j = 1$ to $K$ do
12:         $w_{j,s} = w_{j,s-1} + y_t A_{s-1,(j,i_t)}^{-1} x_t$;
13:      end for
14:      if $t \geq epoch$ then
15:         update $A_s$ [Eq. (4.10) – Eq. (4.13)];
16:      end if
17:      $s \leftarrow s + 1$;
18:   end if
19: end for
```

### 4.2.2 Practical considerations

During the initial few rounds, the weight vectors $w$ are not well formed and since the updates of $A$ depend on $w$, poor initial estimates of $w$ may lead to poor estimates of $A$, which in turn could worsen the estimates of weights $w$ as they depend on $A$. To account for this, we wait for a number of rounds (a *priming duration* which we also refer to as epoch) before turning on the updates for $A$, and until then update the weight vectors $w$ as if we were learning $K$ independent perceptrons (i.e., by using $A = \frac{1}{K} \times I_d$ initially). Once the priming duration is over, we turn on the updates of $A$. We follow the same guideline for our approaches as well as the other baselines that use a task relationship matrix. Our procedure is summarized in Algorithm 1.

### 4.2.3 Computational efficiency

CMTL updates only weight vectors, whereas BATCHOPT, OMTLCOV, OMTLLOG and OMTLVON additionally update task interaction matrices as well. Hence, CMTL is always faster as compared to the other approaches.

BATCHOPT computes matrix multiplications ($O(K^3)$), whereas OMTLCOV computes matrix covariances ($O(K^2)$). Our approaches, OMTLLOG and OMTLVON, use operations such as inverse, exponentiation and logarithms of $K \times K$ matrices which can be expensive, especially when the number of tasks $K$ is large. However, these operations can be expedited using Singular Value Decomposition (SVD) routines for the matrix $A$, i.e., $A = VDV^T$ where $D$ is a diagonal matrix consisting of the singular values. Then these operations boil down to computing the same for the diagonal matrices which have $O(K)$ complexity. For example, the matrix exponentiation can be done as $\exp(A) = V \exp(D) V^T$. The SVD step can be performed using efficient eigen-decomposition algorithms such as the randomized SVD algorithm (Liberty et al., 2007).

## 4.3 An Active Learning Extension

Active Learning in a multitask setting (batch/online) is considered a difficult problem and little prior work exists in this realm. What complicates active learning in a multitask setting is that one needs to evaluate the informativeness of an example across several tasks before deciding whether or not to query its label.

In this work, we show that our online multitask learning framework can be easily extended to an active learning setting that takes into account the task relatedness. A naïve active learning strategy in an online setting is to use the margin biased randomized sampling (Cesa-Bianchi et al., 2006) for active learning. More specifically, the approach proposed in Cesa-Bianchi et al. (2006) uses a sampling probability term $p = b/(b + |r_{i_t}|)$ to decide whether to query the label of an incoming example belonging to the task $i_t$, where $r_{i_t}$ is the signed margin of this example on the hypothesis being learned. The parameter $b$ is set to a fixed value and dictates the level of aggressiveness of the sampling process. However, this approach does not exploit the task relatedness in the presence of multiple tasks.

We propose to use the task relationship matrix $A$ of pairwise task similarity coefficients to set the sampling parameter $b$. For an incoming example belonging to the task $i_t$, we set $b = \sum_j |A_{i_t,j}|$ which is nothing but the sum of the absolute values of the $i_t^{th}$ row (or column) of the matrix $A$. Thus $b$ denotes the sum of similarities of task $i_t$ with all other tasks. It is easy to see that the expression for $b$ would take a large value (meaning more aggressive sampling) if the tasks are highly correlated, whereas $b$ will have a small value (moderately aggressive sampling) if the tasks are not that highly related.

## 4.4 Experiments

In this section, we evaluate our online task relationship learning approaches by comparing them against a number of baselines, and on several datasets. The results have been averaged over 20 runs for random permutations of the training data order and standard deviations are also reported.

### 4.4.1 Setup

In this section, we describe the datasets used and the methods compared.

**4.4.1.1 Datasets.** We report our results on one synthetic (*Synthetic*), and three real world (*20newsgroups*, *Sentiment* and *Spam*) datasets. *Synthetic* is an artificial dataset which has been generated as follows. First, we construct three weight vectors $w_1$, $w_2$, $w_3 \in \mathbb{R}^{10}$ with $w_1 = -w_2$, and $w_3$ being uncorrelated with the other two. Then we generate three binary classification datasets, each consisting of a sample of 100 data points. Each dataset comprises a learning task. We mix these three datasets with examples in random task order and split the data into 200 training examples and 100 test examples. *20newsgroups*, constructed as in Raina et al. (2006) contains a total of 11269 training and 7505 test examples for 10 tasks. *Sentiment* dataset (Blitzer et al., 2007b) consists of user reviews of 8 classification tasks on 8 data types (apparel, books, DVD, electronics, kitchen, music, video, and other) from Amazon.com. Each sentiment classification task is a binary classification which corresponds to classifying a review as positive or negative. *Spam* (Crammer et al., 2009) consists of 3000 test and 4000 training examples constructed from email messages of 3 different users (each user is a task).

**4.4.1.2 Methods.** We compare prediction accuracy, number of mistakes and (for the active learning variants) number of labels queried for STL, IPL, CMTL (Cavallanti et al., 2008), BATCHOPT, OMTLCOV, OMTLLOG, OMTLVON (summarized in Table 4.1).

### 4.4.2 Task relationships learned

To demonstrate that our proposed algorithms can discover the task relationships reliably, we experiment with *Synthetic* which has known task relationships. Table 4.2 shows the task (weight vector) correlation matrices learned by CMTL, OMTLLOG and OMTLVON

**Table 4.1**. Description of methods being compared.

| **Method** | Description |
|---|---|
| STL | *pooling* based single task perceptron |
| IPL | *K* independent perceptrons (CMTL with identity interaction matrix) |
| CMTL | online perceptron (Cavallanti et al., 2008) with fixed interaction matrix |
| BATCHOPT | online multitask perceptron with *batch optimal* update |
| OMTLCOV | online multitask perceptron with *covariance* based update |
| OMTLLOG | online multitask perceptron with *LogDet divergence* based update |
| OMTLVON | online multitask perceptron with *von-Neumann divergence* based update |

**Table 4.2**. Task correlation of *Synthetic* for CMTL, OMTLLOG and OMTLVON with epoch = 0.5 (single run with random data order). **ID** denotes the task ID.

| **Method** | **ID** | 1 | 2 | 3 |
|---|---|---|---|---|
| | 1 | 1.0000 | -0.2030 | 0.5217 |
| CMTL | 2 | -0.2030 | 1.0000 | 0.1371 |
| | 3 | 0.5217 | 0.1371 | 1.0000 |
| | 1 | 1.0000 | -0.9059 | 0.0003 |
| OMTLLOG | 2 | -0.9059 | 1.0000 | 0.1225 |
| | 3 | 0.0003 | 0.1225 | 1.0000 |
| | 1 | 1.0000 | -0.8171 | 0.0322 |
| OMTLVON | 2 | -0.8171 | 1.0000 | 0.1295 |
| | 3 | 0.0322 | 0.1295 | 1.0000 |

on *Synthetic* which consists of 3 tasks. As can be seen, both OMTLLOG and OMTLVON are able to capture the negative correlations between $w_1$ and $w_2$, and the uncorrelatedness of $w_3$ with the other two weight vectors. On the other hand, since the approach of Cavallanti et al. (2008) is biased towards enforcing positive correlations, it falsely concludes a significant correlation of $w_3$ with $w_1$ and $w_2$. At the same time, for CMTL, $w_1$ and $w_2$ appear less negatively correlated than they actually are. We also note that the task correlations learned by OMTLCOV and BATCHOPT were off from the truth by a reasonable amount.

### 4.4.3 Results

We now report accuracy, number of mistakes and labels queried with active learning.

**4.4.3.1 Accuracy.** We report the prediction accuracies of our update rules for the datasets *20newsgroups*, *Sentiment* and *Spam*. As discussed earlier (refer to section 4.2.2), the various update schemes need to decide when to start updating the task relationship matrix $A$. It is not advisable to update $A$ until the weight vectors are well-formed. As mentioned earlier in section 4.2.2, we wait until a duration called the priming phase (denoted by epoch), which is decided based on the fraction of datapoints we want to see in the stream, before turning on the update for $A$. During this phase, $A$ is set to an identity matrix (i.e., independent tasks). Once we get past the epoch point, we switch to the incremental updates of $A$. Table 4.3 presents the results on *20newsgroups*, *Sentiment* and *Spam* data for $epoch = 0.5$. OMTLLOG performs the best for *20newsgroups* and *Sentiment* and OMTLCOV is the best for *Spam*. In addition, OMTLVON outperforms the baseline accuracy for all the datasets.

Figure 4.1 demonstrates the variation in prediction accuracy with increase in epoch values. As can be seen, an increase in epoch value leads to a gradual improvement in prediction accuracy. However, we cannot have a very high value of epoch, that will amount to waiting too long, leading to learning $K$ independent perceptrons for most of the duration. This might not be able to completely utilize the relatedness among the tasks in the weight update equations. This fact is reflected for *20newsgroups* around epoch = 0.8, after which the accuracies of OMTLCOV and OMTLLOG drop down to that of the IPL accuracy. For *Sentiment* and *Spam*, this inflection point was observed around epoch = 0.7 and epoch = 0.8, respectively.

**4.4.3.2 Number of mistakes.** We present the number of mistakes of all algorithms in Table 4.4 for $epoch = 0.5$. Except for *Spam*, OMTLLOG has the lowest number of mistakes and OMTLCOV and OMTLLOG convincingly outperform CMTL. These empirical results imply that the theoretical mistake bounds of the proposed update rules should be better than CMTL. However, the data-dependent adaptive nature of the interaction matrix

**Table 4.3**. Accuracy for *full training data* (epoch = 0.5).

| **Method** | **Accuracy (Standard Deviation)** | | |
|---|---|---|---|
| | *20newsgroups* | *Sentiment* | *Spam* |
| STL | 56.94(±3.32) | 66.31(±2.14) | 76.45(±1.56) |
| IPL | 75.20(±2.35) | 67.24(±1.40) | 91.02(±0.77) |
| CMTL | 73.14(±2.35) | 67.38(±1.82) | 90.17(±0.66) |
| BATCHOPT | 75.78(±2.22) | 67.59(±1.40) | 91.10(±0.80) |
| OMTLCOV | 80.84(±0.70) | 70.49(±0.53) | **92.17(±0.52)** |
| OMTLLOG | **81.83(±0.46)** | **73.49(±0.53)** | 91.35(±1.12) |
| OMTLVON | 76.51(±1.54) | 67.60(±0.83) | 91.05(±1.05) |

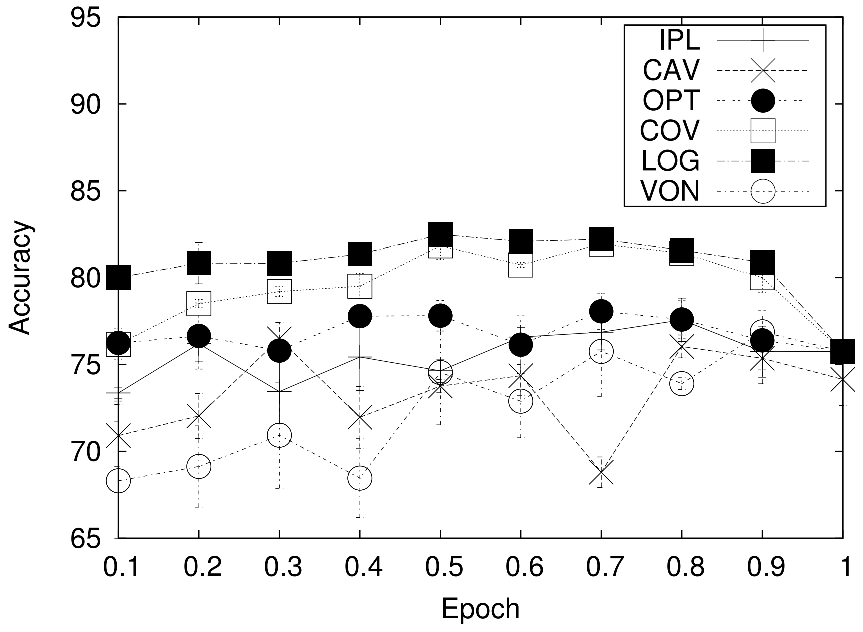


**Figure 4.1**. Accuracy vs. epoch on *20newsgroups*.

**Table 4.4**. Number of mistakes with epoch = 0.5 for *full training data*.

| **Method** | **Number of mistakes** | | |
|---|---|---|---|
| | *20newsgroups* | *Sentiment* | *Spam* |
| STL | 4818 | 25273 | 742 |
| IPL | 3002 | 24317 | 348 |
| CMTL | 3246 | 24212 | 389 |
| BATCHOPT | 3008 | 24371 | 347 |
| OMTLCOV | 2696 | 22980 | **337** |
| OMTLLOG | **2674** | **22023** | 347 |
| OMTLVON | 3105 | 24474 | 380 |

renders the theoretical analysis difficult and we defer it to future work.

**4.4.3.3 With active learning.** The accuracy and number of labels queried of our active learning variants for all the approaches are shown in Table 4.5. The left half of the table presents prediction accuracies and the right half compares the number of labels requested. As mentioned in section 4.3, we use the task interaction matrix to set the sampling parameter for the active learning variants of OMTLCOV, OMTLVON, OMTL-LOG, whereas the baselines use a fixed label sampling parameter as in Cesa-Bianchi et al. (2006). When compared to Table 4.3, it can be seen that the accuracies are similar for

**Table 4.5**. Accuracy and labels queried with epoch $= 0.5$ for *full training data* with *active learning* variants.

| **Method** | **Accuracy (Standard Deviation)** **Labels requested (% reduction)** | | |
|---|---|---|---|
| | *20newsgroups* | *Sentiment* | *Spam* |
| STL | 57.87($\pm$2.18) | 67.67($\pm$2.63) | 76.82($\pm$1.90) |
| | 7334 (35%) | 44224 (39.6%) | 1827 (39.1%) |
| IPL | 75.28($\pm$1.92) | 68.80($\pm$1.06) | 90.98($\pm$0.52) |
| | 7265 (35.5%) | 44437 (39.3%) | 1917 (36.1%) |
| CMTL | 73.79($\pm$2.52) | 68.17($\pm$1.42) | 89.96($\pm$0.75) |
| | 10171 (9.75%) | 63810 (12.84%) | 2276 (24.13%) |
| BATCHOPT | 74.42($\pm$2.18) | 68.18($\pm$1.82) | 90.93($\pm$0.59) |
| | 6956 (38.3%) | 52577 (28.18%) | 1898 (36.73%) |
| OMTLCOV | 79.78($\pm$0.46) | **71.33($\pm$0.68)** | **90.72($\pm$0.87)** |
| | **4784 (57.55%)** | 42112 (42.48%) | 1347 (55.1%) |
| OMTLLOG | **80.50($\pm$0.53)** | 71.16($\pm$0.60) | 90.32($\pm$0.85) |
| | 5966 (47.06%) | **24162 (67%)** | **1288 (57.06%)** |
| OMTLVON | 75.53($\pm$2.99) | 67.63($\pm$2.23) | 89.14($\pm$1.66) |
| | 6336 (43.75%) | 54854 (25.07%) | 1583 (47.23%) |

passive and active versions of all the approaches compared. However, the number of labels requested in all the active cases are substantially lower than the corresponding passive versions. Moreover, for both *20newsgroups* and *Sentiment*, the number of labels queried by OMTLCOV and OMTLLOG are substantially lower than that of CMTL. Thus, the active learning variants result in a substantial reduction in the number of labels queried without noticeable degradation in prediction accuracy.

### 4.4.4 Remarks

For all cases, the proposed update rules of OMTLCOV and OMTLLOG outperform all other approaches compared and are substantially better than the fixed interaction matrix based CMTL. All active learning variants reduce the number of labels queried with the reduction for the proposed update rules being substantial ($\sim$ ($\mathbf{42}\% - \mathbf{58}\%$) for OMTLCOV and $\sim$ ($\mathbf{47}\% - \mathbf{67}\%$) for OMTLLOG). This confirms that the use of *an adaptive interaction matrix* benefits the multitask learning process in the online setting and is also an useful tool to devise active learning strategies. It is worth noting that BATCHOPT, while optimal in the batch setting, does not give the best results in the online setting and in most cases performs barely better than IPL. Thus, the poor performance of both CMTL and BATCHOPT highlights the need to devise adaptive multitask relationship learning strategies for the online setting.

Figure 4.1 emphasizes the importance of choosing a good value for epoch which varies based on the dataset. One straightforward approach would be to compute the variance of the different weight vectors and wait until the variance has settled for all. However, it is difficult to know when the variance has settled down and requires nonparametric statistical tests which are computationally prohibitive and do not fit into the computationally efficient paradigm of online learning. Our work resorts to threshold based decisions but a favorable choice would be to learn the epoch value from the data.

We experimented with multiple passes over data where we use IPL in pass 1 and then switch to the respective update rules for all subsequent passes. At the end of each pass, the interaction matrix (to be used in the following pass) is updated based on the weight vectors learnt in that pass. We noticed that the multipass results do not improve much over the single pass results. Also, the time required for the multiple passes is substantially more than that required by the single pass approaches.

The von-Neumann update rule is numerically unstable and we compute matrix exponential using spectral decomposition, as suggested in Tsuda et al. (2005). However, the spectral decomposition based technique is also sometimes unstable which results in poor performance and high variance, as demonstrated in our results. We did not experiment with Schur decomposition based matrix exponential which might yield better results.

## 4.5 Summary and Future Directions

We have explored an online setting for learning task relationships. Our proposed approach constructs an adaptive *interaction matrix* which quantifies the relatedness among the multiple tasks and *also* uses this matrix to update the related tasks. We have presented simple update rules based on different Bregman divergence measures and showed how the task interaction matrix can be used to select the label sampling parameter in an online active learning setting, given multiple related learning tasks.

An alternate active learning scenario is to perceive labels for all examples, but the task or domain information is revealed only for some of the examples. Our proposed framework can be extended for such scenarios by simultaneously doing online active learning on $(x, i_t)$ and $([x,y], i_t)$ pairs for the *multidomain* and *multitask* cases, respectively. Note that the multidomain case *does not require* the labels $y$ to distinguish between domains since the assumption is that $p(x)$ is different for different domains. However, the multitask case *requires* the labels since $p(x)$ stays the same for all tasks but $p(x,y)$ changes.

Our work highlights the challenges posed by the joint learning of task weight vectors and the task relationship matrix in the online setting; the major hurdle being the decision on how long to wait until the individual weight vectors of all the tasks are stable enough to be used for computing the task interaction matrix. Our work proposed predefined wait periods that seem to work well in practice. However, it is imperative that we clearly understand what factors determine the confidence of weight vectors and whether it is possible to learn the switch over point from the data. As already mentioned, use of nonparametric statistical tests seems to be an overkill and is fundamentally against the *computationally efficient* nature of online learning. At present, we do not have a good answer to this question which provides an interesting direction for future work.

Our empirical results demonstrate fewer number of mistakes (and improved label complexities for the active learning extension) when compared to other baselines. However, it is not theoretically apparent whether our proposed approach would yield better mistake bounds than the CMTL approach. What complicates the analysis is that our task interaction matrix is adaptive, unlike that of Cavallanti et al. (2008) which assumes a fixed interaction matrix. We believe this to be an interesting direction for future work.

# CHAPTER 5

# ACTIVE TRANSFER

Active learning in a domain adaptation setting has received little attention so far and, to the best of our knowledge, there exists no prior work that presents a principled framework to harness domain adaptation for active learning. One interesting setting was proposed in Chan and Ng (2007) where the authors apply active learning for word sense disambiguation in a domain adaptation setting. In addition, they also improve vanilla active learning when combined with domain adaptation. However, their approach does not use the notions of domain separator and hybrid oracle. Moreover, unlike our approach, their method only works in a batch setting.

Active learning in an online setting has been discussed in Dasgupta et al. (2009) and Cesa-Bianchi et al. (2006). The work of Dasgupta et al. (2009) assumes input data points uniformly distributed over the surface of an unit sphere. However, we cannot make such distributional assumptions for domain adaptation. As mentioned earlier, Cesa-Bianchi et al. (2006) provide worst-case analysis which is independent of any input data distribution. However, none of these explicitly consider the case of domain adaptation. Nonetheless, the framework of Cesa-Bianchi et al. (2006) folds nicely into our proposed Active Learning Domain Adaptation (ALDA) framework. Monteleoni and Kääriäinen (2007) present extensive empirical results to compare the performance of the two aforementioned approaches. However, all these settings are different from ours in that these works consider only active learning in an online setting without leveraging interdomain information.

A combination of transfer learning with active learning has been presented in (Shi et al., 2008). One drawback of their approach is the requirement of an initial pool of labeled target domain data which helps train the in-domain classifier. Without this in-domain classifier, no transfer learning is possible in their setting.

We consider the *supervised* domain adaptation setting (Jiang, 2008) where we have a large amount of labeled data from some source domain, a large amount of unlabeled data from a target domain, and *additionally*, a small budget for acquiring labels in the target domain. As earlier, we once again note that *supervised domain adaptation* contains labeled data in both *source* and *target*, whereas *unsupervised domain adaptation* contains labeled data only in *source*, and *semisupervised domain adaptation* contains labeled data in *source* and both labeled and unlabeled data in *target*. We show how, apart from leveraging information in the usual domain adaptation sense, the information from the source domain is *further* leveraged to selectively query for labels in the target domain (instead of choosing them randomly, as is the common practice). We achieve this by first training the best possible classifier in the source without using target labels, for instance, either by simply training a supervised classifier on the source labeled data, or by using some unsupervised adaptation technique using the unlabeled target data as well. Then, we use this learned hypothesis in various ways to leverage the source domain information when we are additionally given some fixed budget for acquiring some extra *labeled* target data (i.e., the active learning setting (Settles, 2009)).

Our proposed framework is based on three key components. The first component is *unsupervised* domain adaptation (i.e., without target labeled data). The goal of this step is to suitably adapt the source data representation such that it makes the marginal distributions of both source and target distributions look similar. This enables training any traditional supervised classifier for the target domain using the adapted representation of the source data. The second and the third components improve this classifier even further by using active learning to selectively acquire the labels of target examples, given a budget on the target labels. Moreover, these components leverage the source domain information as well. Specifically, the second step employs a *domain separator hypothesis* that rules out querying labels of those target examples that appear "similar" to examples from the source domain. The domain separator hypothesis is a classifier that distinguishes between source and target domain examples and *is learned using only unlabeled examples from the two domains*. The third component is a hybrid oracle which consists of two oracles: one that provides labels for free but is imperfect (there could be noise), and one expensive (but "perfect") oracle used in the standard active learning settings. The source classifier acts as the free oracle which, although not perfect, can provide correct labels for most of the examples queried (essentially, the ones that appear "source" like).

The proposed ALDA framework is sufficiently general to allow varied choices of domain adaptation and active learning modules. In addition, ALDA applies to both batch (section 5.1) as well as online settings (section 5.3). In this work, we present empirical results (section 5.4) for specific choices of the domain adaptation and the active learning schemes. For both batch and online settings, we empirically demonstrate that the proposed approach leads to significant improvement in prediction accuracies for a given target label budget when compared to other baselines. Moreover, for the online setting, apart from showing empirically better performance, we also show that our approach results in smaller mistake bounds under suitable notions of domain separation. We provide intuitive arguments for smaller label complexity in the target domain when compared to the standard active learning where we do not have access to data from a related distribution.

## 5.1 ALDA: Active Learning Domain Adapted

In this section, we propose a principled approach towards active learning in a target domain by leveraging information from a related source domain. In our setting, we are given a small budget for acquiring labels in a target domain, which makes it imperative to use active learning in the target domain. However, our goal is to *additionally* leverage the domain relatedness by exploiting whatever information we might already have from the source domain. At a high level, our proposed approach aims to answer the following questions:

1. given source information, which samples in the *target* are the most informative (in an *active sense*)?
2. among the *informative target samples*, can we use source information to *infer labels* of a few *informative target samples*, such that the actual number of target labels queried (from an oracle) is reduced even further?

In the following, we provide answers to the above questions. We begin by introducing some notations and presenting an overview of the ALDA framework.

### 5.1.1 Preliminaries

Let $\mathscr{X} \subset \mathbb{R}^d$ denote the instance space and $\mathscr{Y} = \{-1, +1\}$ denote the label space. Let $\mathscr{D}_S(x,y)$ and $\mathscr{D}_T(x,y)$ be the joint source and target distributions, respectively. We have a set of source labeled examples $L_S(\sim \mathscr{D}_S(x,y))$ and a set of source unlabeled examples $U_S(\sim \mathscr{D}_S(x))$. Additionally, we also have a set of target unlabeled instances $U_t(\sim \mathscr{D}_t(x))$, from which we *actively* acquire labels. Furthermore, $w_{src}$ denotes a classifier learned from the source labeled data and $w_{ds}$ denotes the *domain separator* hypothesis. Finally, let $\phi$ represent an unsupervised domain adaptation algorithm that outputs a classifier $u_\phi$. Note that learning $u_\phi$ *does not require* labeled target examples.

Figure 5.1 shows our basic setup for ALDA. The Active Learning (AL) module is a combination of the submodules Uncertainty Sampler (US) (that is initialized using the $u_\phi$ classifier from the unsupervised domain adaptation phase) and Domain Separator (DS) (that uses the $w_{ds}$ classifier). In addition, the setup employs a *hybrid oracle* which is a combination of a free oracle $\mathscr{O}_f$ and an expensive oracle $\mathscr{O}_c$. The free oracle $\mathscr{O}_f$ is nothing but the classifier ($w_{src}$) learned using the source labeled samples $L_S$. At each step, the learner *actively* selects an informative target sample and gets it labeled by an *appropriate* oracle. This continues in an iterative (for the batch setting) or online fashion until some

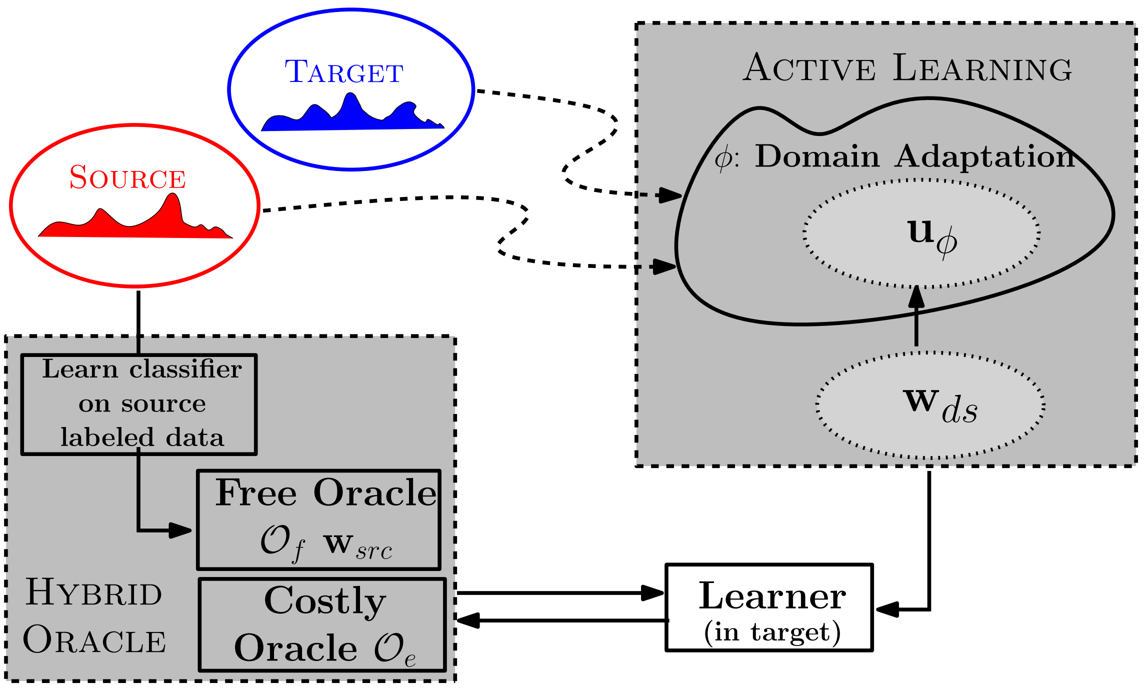

TARGET
SOURCE
ACTIVE LEARNING
$\phi$: Domain Adaptation
$\mathbf{u}_\phi$
$\mathbf{w}_{ds}$
Learn classifier on source labeled data
Free Oracle $\mathcal{O}_f$ $\mathbf{w}_{src}$
Costly Oracle $\mathcal{O}_e$
HYBRID ORACLE
Learner (in target)

**Figure 5.1**. An illustration of the proposed ALDA framework. Domain adaptation can be performed using any black-box unsupervised domain adaptation approach (e.g., (Blitzer et al., 2006, Sugiyama et al., 2007)). The active learning block can be any batch or online active learner.

stopping criterion is met (say, for example, reached prescribed accuracy or exhausted label budget). Next we describe each of these individual modules in more detail.

### 5.1.2 Initializing the uncertainty sampler

The first phase of ALDA learns an *unsupervised* domain adapted classifier $u_\phi$ which uses labeled source data, and unlabeled source and target data. Note that this phase does not use any labeled target data, hence the name, unsupervised. There are a number of ways to learn the classifier $u_\phi$. In this work, we take the approach (Sugiyama et al., 2007) that is based on estimating the *importance ratio* between the source and the target distribution, without actually estimating these distributions. The source domain examples, with their corresponding importance weights, can then be used to train any classifier which is now readily adapted for the target domain (of course, this can potentially still be improved, given extra labeled target data). Note that the unsupervised domain adaptation step can be performed in a number of other ways as well; for example, Kernel Mean Matching (KMM) can be performed by matching the source and target distributions in some Reproducing Kernel Hilbert Space (RKHS) and computing the importance weights of source domain examples (Huang et al., 2007). Another approach (especially for NLP problems), could be to use Structural Correspondence Learning (SCL) to identify invariant ("pivot") features between source and target, and use these features for unsupervised domain adaptation (Blitzer et al., 2006). The unsupervised domain adapted classifier $u_\phi$ serves as the initializing classifier for the subsequent active learning phase of our approach.

## 5.2 Leveraging Domain Divergence

It turns out that, in addition to using the source domain information to initialize our active learner in the target domain, we can further leverage the domain relatedness information to improve the active learning phase in the target.

In this section, we propose the *domain separator* that further leverages the relatedness of source and target domains while performing active learning in the target. Assuming the source and target domains to be related, our proposed technique exploits this relatedness to, upfront, rule out acquiring labels of those *target domain examples* that "appear" to be close to the source domain.

As an example, Figure 5.2 shows a typical domain separator hypothesis (denoted by

$w_{ds}$) that separates the SOURCE and TARGET examples. We note that similar source and target examples are expected to have the same labels since only the marginal distribution of examples changes between the source and target examples (i.e., $\mathscr{D}_S(x) \neq \mathscr{D}_t(x)$), whereas the conditional distribution of labels (given the examples) stays the same (i.e., $\mathscr{D}_S(y|x) = \mathscr{D}_t(y|x)$). Observe that if the source and target distributions are far apart, then the two domains can be perfectly classified by this separator. However, if the domains are similar, it is expected that there will be a reasonable overlap and therefore, some of the target (or source) domain examples might lie on the source (or target) side (encircled instances in Figure 5.2) and hence, will be misclassified by the domain separator hypothesis. Acquiring labels for such target domain examples (that lie on the source side) is not really needed since the initial hypothesis (refer $u_\phi$ in Figure 5.1) of ALDA would already have taken into account such examples. Therefore, such target examples can be effectively ignored from being queried. Thus, the domain separator hypothesis, which can be learned using *only* source and target *unlabeled* examples, provides a novel way of performing active sampling in domain adaptation settings.

The domain separator hypothesis avoids querying the labels of all those target examples that lie on the source side of the domain separator and hence, are misclassified by it. This

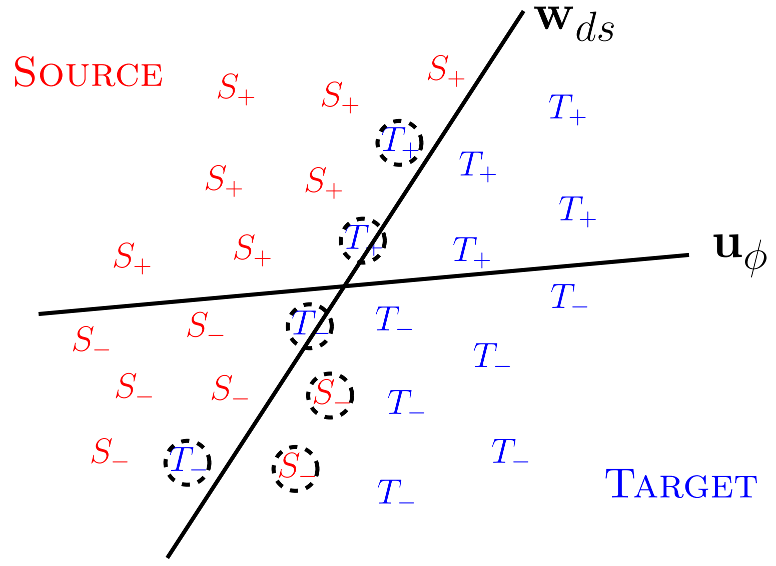


**Figure 5.2**. An illustrative diagram showing the domain separator hypothesis $w_{ds}$ separating source data from target data and the classifier $u_\phi$ learned using the unsupervised domain adapted source classifier.

number, in turn, depends on the domain divergence between the source and target domains. For reasonably similar domain pairs, the domain divergence is expected to be small which implies that a large number of target examples lies on the source side. We can formalize the label complexity reduction due to the domain separator hypothesis. As earlier, let $\mathscr{D}_s$ and $\mathscr{D}_t$ denote the source and target joint distributions, and let $p_{\mathscr{D}_s}(x)$ and $p_{\mathscr{D}_t}(x)$ be probabilities of an instance $x$ belonging to the source and the target, respectively, in the unlabeled pool used to train the domain separator hypothesis. Let $\Delta$ denote the Mahalanobis distance between the source and target distributions. The Bayes error rate (Tumer and Ghosh, 1996) of the domain separator hypothesis is: $E_{bayes} \leq \frac{2p_{\mathscr{D}_s}(x)p_{\mathscr{D}_t}(x)}{1+p_{\mathscr{D}_s}(x)p_{\mathscr{D}_t}(x)\Delta}$. Thus, the label complexity reduction due to the domain separator hypothesis is proportional to the number of target examples misclassified by the domain separator hypothesis. This is again, proportional to the Bayes error rate which in turn, is inversely related to the distance between the two domains.

### 5.2.1 Hybrid oracle

ALDA additionally exploits the source domain information by using the source learned hypothesis (see, $w_{src}$ in HYBRID of Figure 5.1) as an oracle that provides labels for free. We denote this oracle by $\mathscr{O}_f$. Accordingly with the *covariate shift* assumption in domain adaptation, only the marginal distribution changes across domains whereas the conditional distribution remains fixed. If some target example appears to be close to the source domain then it is reasonable to assume that the prediction of the source classifer (which depends on the source conditional distribution) on that target sample should be close to the prediction of a *good* target classifier on that target sample. This explains the use of the source learned classifier as a free oracle for the target domain examples. Moreover, as in the standard active learning setting, we also have an expensive oracle $\mathscr{O}_c$. This leads to a hybrid setting which utilizes one of these two oracles for each actively sampled target example. The hybrid oracle starts with a domain adapted source initialized classifier ($u_\phi$ in US of Figure 5.1) and uses the domain separator hypothesis ($w_{ds}$ in DS of Figure 5.1) to assess which of the uncertain target examples lie on the source side and, for all such examples, it queries the labels from the free oracle $\mathscr{O}_f$. For the remaining uncertain examples that lie on the target side, the hybrid approach queries the expensive oracle $\mathscr{O}_c$. Although the oracle $\mathcal{O}_f$ is not perfect, the hope is that it can still provide correct labels for most of the target examples.

Algorithm 2 presents the final algorithm that combines all aforementioned schemes. This algorithm operates in a batch setting and we call it B-ALDA (for Batch-ALDA). As mentioned earlier (ref. section 5.1.2), the importance ratio in line two of Algorithm 2 can be obtained by the techniques SCL (Blitzer et al., 2006), KMM (Huang et al., 2007), etc.

## 5.3 Online ALDA

In B-ALDA, the active learning module, at each iteration, chooses the data point that lies closest to the decision boundary. However, this approach is prohibitively slow for large or even moderately sized datasets. Hence, we propose Online ALDA (O-ALDA) which performs active learning in an online fashion and for each example decides whether or not to query its label. As in standard active learning, this query decision must be biased by the *informativeness* of the example.

To extend ALDA to the online setting, we adopt the label query strategy proposed in (Cesa-Bianchi et al., 2006). However, we note that our framework is sufficiently general

**Algorithm 2** B-ALDA

```
1: Input: $L_S = \{x_S, y\}$; $U_S$; $U_t$; maxCost (label budget $K$ and/or desired accuracy $\varepsilon$);
2: Output: $v$ (target classifier);
3: cost := 0;
4: $S := \tilde{L}_S$ (importance weighted $L_S$ learned using $L_S, U_S$ and $U_t$);
5: $u_\Phi$ := learn a domain adapted source classifier using $S$;
6: $w_{ds}$ := learn a classifier using the data $\{U_S, +1\}$ and $\{U_t, -1\}$;
7: $w_{src}$ := learn a domain adapted source classifier using $L_S$;
8: while (cost < maxCost) do
9:    $\bar{x}_t$ := US($u_\Phi$,$U_t$);          /* choose most informative target point */

10:   $\hat{y}_{ds}$ := DS($w_{ds}$,$\bar{x}_t$);         /* compute source resemblance */
11:   if ($\hat{y}_{ds} == +1$) then
12:      $y_t = \mathcal{O}_f(w_{src}, \bar{x}_t)$;       /* query the free oracle */
13:   else if ($\hat{y}_{ds} == -1$) then
14:      $y_t = \mathcal{O}_c(\bar{x}_t)$;              /* query the costly oracle */
15:      cost ← cost + 1;
16:   end if
17:   $S = S \cup \{\bar{x}_t, y_t\}$;
18:   retrain $u_\Phi$ using $S$;
19: end while
```

and allows integration with other *active online sampling strategies*. The sampling scheme in (Cesa-Bianchi et al., 2006) proceeds in rounds and at round $i$ queries the label of the example $x^i$ with probability $\frac{b}{b+|r^i|}$, where $|r^i|$ is the *confidence* (in terms of margin) of the current weight vector on $x^i$. Parameter $b$ quantifies how aggressively the labels are being queried. A large value of $b$ implies that, in expectation, a large number of labels will be queried (aggressive sampling), whereas a small value would lead to a small number of examples being queried (conservative sampling). For each label queried, the algorithm updates the current weight vector if the label was predicted incorrectly. It is easy to see that the total number of labels queried by this algorithm is $\sum_{i=1}^{T} \mathbb{E}[\frac{b}{b+|r^i|}]$, where $T$ is the total number of rounds. At this point we note that the preprocessing stage of O-ALDA assumes the existence of some (maybe a small amount) of target *unlabeled* data that can be utilized to construct the common representation. The online active learning in the target starts after this preprocessing phase when O-ALDA selectively queries the labels of the target data points that arrive in some random order.

Algorithm 3 presents the online variant of ALDA which we refer to as O-ALDA (for Online-ALDA). As shown in Theorem 5.3.1, our proposed O-ALDA yields provable guarantees on mistake bounds and label complexity.

**Theorem 5.3.1** *Let $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathbb{R} \times \{-1, +1\})^T$ be any sequence of examples and $\mathcal{U}\mathcal{P}_T$ the (random) set of update trials for the algorithm (i.e., the set of trials $i \leq T$ such that $\hat{y}^i \neq y^i$ and $Z^i = 1$). Let $v_0$ be the weight vector with which the base target classifier is initialized and $r^i$ be the margin of O-ALDA on example $x^i$. Then the expected number of mistakes made by the algorithm is upper bounded by*

$$\inf_{\gamma>0} \inf_{v^* \in \mathbb{R}^D} \left( \frac{(2b+1)}{2b} \mathbb{E}\left[ \sum_{i \in \mathcal{U}\mathcal{P}_T} \frac{1}{\gamma} D\gamma(v^*; (\hat{x}^i, y^i)) \right] + \frac{(2b+1)^2}{8b} \frac{||v^* - v_0||^2}{\gamma^2} \right)$$

*The expected number of labels queried by the algorithm is equal to $\sum_{i=1}^{T} \mathbb{E}[\frac{b}{b+|r^i|}]$.*

In the above theorem, $\gamma$ refers to some margin greater than zero such that the cumulative hinge loss of the optimal target hypothesis $v^*$ on $S$ is given by $\sum_1^T D\gamma(v^*; (x^i, y^i))$, where $D\gamma(v^*; (x^i, y^i)) = max\{0, \gamma - y^i v^{*T} x^i\}$ is the hinge-loss on example $i$. In next section, we discuss the above theorem and provide a proof sketch for the mistake bound and the label complexity of O-ALDA. In addition, we discuss the conditions on $v_0$ that lead to improved

**Algorithm 3** O-ALDA

```
1: **Input:** $b > 0$; $L_S = \{x_S, y\}$; $U_S$; $U_t$; maxCost (label budget $K$/desired accuracy $\varepsilon$);
2: **Output:** $v$ (target classifier);
3: cost := 0;
4: $u_\Phi$ := learn a domain adapted source classifier using $L_S, U_S$ and $U_t$;
5: $w_{ds}$ := learn a classifier using the data $\{U_S, +1\}$ and $\{U_t, -1\}$;
6: $w_{src}$ := learn a domain adapted source classifier using $L_S$;
7: while ( ($i <= T$) & (cost < maxCost) ) do
8:   $r^i$ := US($u_\Phi, x_t^i$);              /* compute margin of $i^{th}$ target point */
9:   $\hat{y}_{ds}^i$ := DS($w_{ds}, x_t^i$);            /* compute source resemblance */
10:  sample $Z^i \sim Bernoulli(\frac{b}{b+|r^i|})$;
11:  if ($Z^i == 1$) then
12:    if ($\hat{y}_{ds}^i == +1$) then
13:      $y_t^i = \mathcal{O}_f(w_{src}, x_t^i)$;          /* query the free oracle */
14:    else if ($\hat{y}_{ds}^i == -1$) then
15:      $y_t^i = \mathcal{O}_c(x_t^i)$;               /* query the costly oracle */
16:      cost $\leftarrow$ cost + 1;
17:    end if
18:    if ($y_t^i \neq u_\Phi^T x_t^i$) then
19:      update $u_\Phi$ using online update rule (such, as perceptron);
20:    end if
21:  end if
22: end while
```

mistake bounds in domain adaptation settings as compared to the case where there is no access to data from a related source domain.

### 5.3.1 Mistake bounds

Our basic algorithm ALDA is similar to the streamed active learning of Cesa-Bianchi et al. (2006) and our theoretical analysis follows suit. We use Theorem 1 of Cesa-Bianchi et al. (2006) and claim that a sensible initialization (whenever such information is available) leads to tighter mistake bounds and label complexity in the target domain. For completeness, we repeat Theorem 1 from Cesa-Bianchi et al. (2006) applicable to Active Learning Zero Initialized (ALZI).

**Theorem 5.3.2** *(Cesa-Bianchi et al., 2006, Theorem 1) Let S and $\mathcal{U}\mathcal{P}_T$ be defined as earlier. Then the expected number of mistakes made by the algorithm is upper bounded by*

$$\inf_{\gamma>0} \inf_{v^* \in \mathbb{R}^D} \left( \frac{(2b+1)}{2b} \mathbb{E}\left[ \sum_{t \in \mathscr{U}\mathscr{P}_T} \frac{1}{\gamma} D\gamma(v^*; (\hat{x}_t, y_t)) \right] + \frac{(2b+1)^2}{8b} \frac{||v^*||^2}{\gamma^2} \right) \quad (5.1)$$

*The expected number of labels queried by the algorithm is equal to* $\sum_{t=1}^T \mathbb{E}[\frac{b}{b+|r_t|}]$.

In the above theorem, $\gamma$ refers to some margin greater than zero such that the cumulative hinge loss of the optimal target hypothesis $v^*$ on $S$ is given by $\sum_1^T D\gamma(v^*; (x_t, y_t))$, where $D\gamma(v^*; (x_t, y_t)) = max\{0, \gamma - y_t {v^*}^T x_t\}$ is the hinge-loss on example $t$.

In the ALZI setting, the learner starts with a zero initialized hypothesis. However in ALDA, as depicted in Figure 5.1, we start with a nonzero hypothesis ($u_{da}$) in the TARGET. The following theorem (applicable to ALDA) shows that the mistake bound and label complexity of ALDA is better than ALZI.

**Theorem 5.3.3** *Let S,* $\mathscr{U}\mathscr{P}_T$ *and* $v_0$ *be defined as earlier, and* $r_t'$ *be the margin of* ALDA *on example* $x_t$. *Then the expected number of mistakes made by the algorithm is upper bounded by*

$$\inf_{\gamma>0} \inf_{v^* \in \mathbb{R}^D} \left( \frac{(2b+1)}{2b} \mathbb{E}\left[ \sum_{t \in \mathscr{U}\mathscr{P}_T} \frac{1}{\gamma} D\gamma(v^*; (\hat{x}_t, y_t)) \right] + \frac{(2b+1)^2}{8b} \frac{||v^* - v_0||^2}{\gamma^2} \right) \quad (5.2)$$

*The expected number of labels queried by the algorithm is equal to* $\sum_{t=1}^T \mathbb{E}[\frac{b}{b+|r_t'|}]$.

**Proof.** Proceeding in a manner similar to the proof of Theorem 1 of Cesa-Bianchi et al. (2006), it can be seen that almost all terms in the final expression for the mistake bound cancel out by the telescopic argument. The term that remains is $||v^* - v_0||^2$. The proof follows. ■

It is easy to see that Theorem 5.2 reduces to Theorem 5.1 if we set $v_0 = 0$. We note that, the first term in the mistake bounds of Theorem 5.1 and Theorem 5.2 is the cumulative hinge loss of the *optimal* target classifier. This term will be the same irrespective of the initialization used. So the difference in the mistake bounds of Theorem 5.1 and Theorem 5.2 is due to the second term, which in our case is smaller provided $\theta \leq cos^{-1}\left(\frac{||v_0||}{2||v^*||}\right)$, where $\theta$ is the angle between the initializing hypothesis $v_0$ and the target hypothesis $v^*$. Without loss of generality, assuming the norm of $v_0$ and $v^*$ stays fixed (which is true since both the *initial* and the *optimal* hypotheses remain unchanged during learning in target domain), as the value of $\theta$ decreases, it causes $||v^* - v_0||^2$ to decrease, leading to reduced mistake bounds in our case (Theorem 5.2). Thus, in our framework, $\theta$ incorporates the notion of the domain separation that influences the mistake bounds. For small values of $\theta$, the source and target domains have high proximity such that the *initial target* hypothesis $v_0$ lies reasonably close to the *optimal target* hypothesis $v^*$. As a result, in such cases, ALDA is expected to make a smaller number of mistakes to get to the optimal hypothesis.

### 5.3.2 Label complexity

ALDA is initialized with a *nonzero hypothesis* $v_0 = u_{da}$ learned using data from a related source domain. Hence, the sequence of hypotheses ALDA produces, will in expectation, have higher confidences margins $|r'_t|$ as compared that of ALZI which is based on a *zero initialized hypothesis* $v_0 = 0$. Therefore, at each step the sampling probability of ALDA given by $\frac{b}{b+|r'_t|}$ will also be smaller, which will lead to a smaller number of queried labels since it is nothing but $\sum_{t=1}^{T} \mathbb{E}[\frac{b}{b+|r'_t|}]$.

Now, we present an intuitive argument for the lower label complexity of O-ALDA as compared to single task online active settings. O-ALDA is initialized with a *nonzero hypothesis* $v_0 = w_{src}$ learned using data from a related source domain. Hence, the sequence of hypotheses O-ALDA produces, will in expectation, have higher confidences margins $|\bar{r}^i|$ as compared to some *zero initialized hypothesis*. Therefore, at each step the sampling probability of O-ALDA given by $\frac{b}{b+|\bar{r}^i|}$ will also be smaller, which will lead to a smaller number of queried labels since it is nothing but $\sum_{i=1}^{T} \mathbb{E}[\frac{b}{b+|\bar{r}^i|}]$.

## 5.4 Experiments

In this section, we demonstrate the empirical performance of our algorithms and compare them with a number of baselines.

### 5.4.1 Setup

In this section we describe our datasets and the methods compared.

**5.4.1.1 Datasets.** We present results for *Sentiment* and *Landmine* datasets. The *Sentiment* dataset consists of user reviews of eight product types (apparel, books, DVD, electronics, kitchen, music, video, and other) from Amazon.com. The sentiment classification task for this dataset is binary classification which corresponds to classifying a review as positive or negative. The dataset consists of several domain pairs with varying $\mathscr{A}$-distances, akin to a sense described in Ben-David et al. (2006). Table 5.1 shows some of the domain pairs used in our experiments and their corresponding domain divergences in terms of the $\mathscr{A}$-distance (Ben-David et al., 2006).

To compute the $\mathscr{A}$-distance from finite samples of source and target domain, we use a surrogate to the true $\mathscr{A}$-distance (the *proxy* $\mathscr{A}$-distance) in a manner similar to Ben-David et al. (2006). First, we train a linear classifier to separate the SOURCE domain from the TARGET domain using only unlabeled examples from both. The average per-instance hinge-loss of this classifier subtracted from 1 serves as our estimate of the *proxy* $\mathscr{A}$-distance. A score of 1 means perfectly separable distributions, whereas a score of 0 means that the two distributions are essentially the same. The amount of useful information that can be leveraged from the other domain would depend on how similar the two domains are. To this end, we therefore choose two datasets from the sentiment data: one with a domain-pair that is reasonably close (*Kitchen*→*Apparel*), and another with a domain-pair that is reasonably far apart (*DVD*→*Books*).

Our second dataset (*Landmine*) is the real landmine detection data (Xue et al., 2007b) which consists of 29 datasets. The datasets 1 to 10 are collected at foliated regions, whereas the datasets 20 to 24 are collected from bare earth or desert regions. We combined datasets $1-5$ as our source domain and treat dataset 24 as the target domain.

**5.4.1.2 Methods.** Table 5.2 summarizes the methods used with a brief description of each. Among the first three (ID, SDA, FEDA), FEDA (Daumé III, 2007) is a state-of-the-art *supervised* domain adaptation method but assumes *passively* acquired labels. The first three methods (ID, SDA, FEDA) acquire labels *passively*. The last five (ALZI,

**Table 5.1**. Proxy $\mathscr{A}$-distances between some domain pairs in the sentiment data

| **Source** | **Target** | **$\mathscr{A}$-distance** |
|---|---|---|
| DVD (D) | BOOKS (B) | 0.7616 |
| DVD (D) | MUSIC (M) | 0.7314 |
| BOOKS (B) | APPAREL (A) | 0.5970 |
| DVD (D) | APPAREL (A) | 0.5778 |
| ELECTRONICS (E) | APPAREL (A) | 0.1717 |
| KITCHEN (K) | APPAREL (A) | 0.0459 |

**Table 5.2**. Description of the methods compared

| **Method** | **Summary** | **Active ?** |
|---|---|---|
| ID | In-domain data | No |
| SDA | Unsupervised domain adaptation followed by *passively* chosen labeled target data | No |
| FEDA | Frustratingly easy domain adaptation (Daumé III, 2007) | No |
| ALZI | Active learning zero initialized | Yes |
| ALRI | Active learning random initialized (with fixed label budget) | Yes |
| **ALSI** | Active learning source (hypothesis) initialized | Yes |
| **B-ALDA** | Batch active learning domain adapted | Yes |
| **O-ALDA** | Online active learning domain adapted | Yes |

ALRI, ALSI, B-ALDA and O-ALDA) methods in Table 5.2 acquire labels in an active fashion. As the description denotes, ALZI and ALRI start active learning in TARGET with a zero initialized and randomly initialized hypothesis, respectively. It is also important to distinguish between ALSI and ALDA (which jointly denotes both B-ALDA and O-ALDA). While both are products of our proposed ALDA framework, ALSI uses an unmodified source classifier learned only from SOURCE labeled data as the initializer, whereas ALDA (i.e., both B-ALDA and O-ALDA) uses an *unsupervised* domain-adaptation technique (i.e., without using labeled target data) to learn the initializer.

In our experiments, we use the instance reweighting approach (Sugiyama et al., 2007) to construct the unsupervised domain adaptated classifier $u_\phi$. However, we note that this step can also be performed using any other unsupervised domain adaptation technique such as Structural Correspondence Learning (SCL) (Blitzer et al., 2006) and Kernel Mean Matching (KMM) (Huang et al., 2007).

We compare all the approaches based on classification accuracies achieved for a fixed unlabeled pool of target examples with varying label budgets. For B-ALDA, we use a margin based classifier (SVM), whereas for O-ALDA we use vanilla perceptron as the base classifier. All online experiments have been averaged over multiple runs with respect to random data order permutations.

### 5.4.2 B-ALDA results

We present results for B-ALDA using a fixed target unlabeled pool and varying target label budgets. Since, domain adaptation is required only when there are small amounts of labeled data in the target, we limit our target label budget to values that are much smaller than the size of the unlabeled target data pool. In addition, due to long running times of our batch ALDA (owing to repeated retraining), we report results on relatively smaller target pool sizes. The B-ALDA results are presented for a unlabeled target pool size of 2500 data points.

**5.4.2.1 Sentiment classification.** Table 5.3 and Table 5.4 present the results for the domain pairs *DVD→Books* and *Kitchen→Apparel*. For these domain pairs, both ALSI and B-ALDA substantially outperform all other baselines. For the distant source-target pair (*DVD→Books*), ALSI performs very well for a small number of target labels (say, 100 and 200). As the number of target labels increases, B-ALDA consistently improves with increasing number of target labels and finally outperforms ALSI. When the source-target pairs are reasonably close (*Kitchen→Apparel*), both ALSI and B-ALDA have similar prediction accuracies which are in turn are much higher that the baseline accuracies.

**5.4.2.2 Landmine detection.** The *Landmine* dataset has a high class imbalance (only about 5% positive examples), so we report Area Under Curve (AUC) scores instead of accuracies. We compare our algorithms with other baselines in terms of the AUC score on the entire pool of target data (the pool size was 300; rest of the examples in dataset 24 were treated as test data). As shown in Table 5.5, our approaches perform better than the other baselines with the domain separator based B-ALDA doing the best (in terms of AUC scores).

**Table 5.3**. Classification accuracies and number of labels requested for *DVD→Books*. Results are averaged over 10 runs. Note: ID, SDA and FEDA are given labels of all examples in the target pool.

| **Method** | **Target Label Budget** | | | | |
|---|---|---|---|---|---|
| | **100** | **200** | **300** | **400** | **500** |
| | **Acc** | **Acc** | **Acc** | **Acc** | **Acc** |
| ID | 50.83 | 57.86 | 62.42 | 55.69 | 62.68 |
| SDA | 62.18 | 62.78 | 55.75 | 52.45 | 50.49 |
| FEDA | 63.92 | 64.27 | 64.88 | 65.94 | 66.19 |
| ALZI | 54.40 | 54.36 | 54.33 | 54.33 | 54.33 |
| ALRI | 54.99 | 59.42 | 61.28 | 65.81 | 65.52 |
| **ALSI** | **63.75** | **66.26** | **68.73** | 63.10 | 62.08 |
| **B-ALDA** | 63.40 | 65.17 | 67.84 | **68.61** | **68.51** |
| **Acc:** Accuracy | | | | | |

**Table 5.4**. Classification accuracies and number of labels requested for *Kitchen→Apparel*. Results are averaged over 10 runs. Note: ID, sDA and FEDA are given labels of all examples in the target pool.

| **Method** | **Target Label Budget** | | | | |
|---|---|---|---|---|---|
| | **100** | **200** | **300** | **400** | **500** |
| | **Acc** | **Acc** | **Acc** | **Acc** | **Acc** |
| ID | 48.40 | 43.44 | 44.92 | 48.40 | 49.77 |
| sDA | 52.78 | 55.41 | 57.37 | 53.60 | 46.37 |
| FEDA | 70.47 | 69.97 | 70.06 | 71.83 | 69.96 |
| ALZI | 54.56 | 54.50 | 54.44 | 54.44 | 54.44 |
| ALRI | 64.97 | 66.86 | 69.01 | 70.40 | 71.06 |
| **ALSI** | **74.91** | 70.58 | **72.97** | **72.34** | 72.29 |
| **B-ALDA** | 71.30 | **70.90** | 71.19 | 71.73 | **73.07** |
| **Acc:** Accuracy | | | | | |

**Table 5.5**. AUC scores (**AUC**) and labels requested (**Lab**) for the *Landmine* dataset.

| **Method** | **Target Budget (300)** |
|---|---|
| | AUC |
| ID | 0.59 |
| sDA | 0.60 |
| FEDA | 0.56 |
| ALZI | 0.59 |
| ALRI | 0.53 |
| **ALSI** | 0.63 |
| **B-ALDA** | **0.65** |

We do not report any label complexity result for B-ALDA as the nature of the algorithm is such that it iterates until the entire label budget is exhausted. Hence, in all the results presented above in Table 5.3, Table 5.4 and Table 5.5, the number of labels used is equal to the target label budget provided.

### 5.4.3 O-ALDA results

One of the goals to propose an online variant for ALDA is to make the proposed approach scale efficiently for larger target pool sizes because batch mode ALDA requires repeated retraining. On the other hand, an online active learner is an efficient alternative because it allows incremental update of the learner for each new selected data point. In this section, we present results for O-ALDA and demonstrate the scalability of the ALDA framework to larger target pool sizes. The results for O-ALDA use the entire target unlabeled pool ($\sim$ 7000 for *Sentiment* data). As a result, the label budget allocated is also much larger as compared to B-ALDA. We note that ID and SDA and FEDA have been made online by the use of the perceptron classifier. In addition, the same online active strategy as O-ALDA has been used for ALZI, ALRI and ALSI.

**5.4.3.1 Sentiment classification.** The results are shown in Table 5.6 and Table 5.7. As the results indicate, on both datasets, our approaches (ALSI and ALDA) perform consistently better than the baseline approaches (Table 5.2) which also include one of the state-of-the-art supervised domain adaptation algorithms (FEDA). We note that ALDA outperforms ALSI for *Kitchen→Apparel* as compared to *DVD→Books*. When the domains are far (*DVD→Books*), the performance of ALDA depends on the underlying domain adaptation technique. However, when the domains are close (*Kitchen→Apparel*), ALDA performs better than ALSI. This behavior suggests that the performance gains achieved by these variants are significant when the source and target domains are *reasonably close*.

**5.4.3.2 Landmine detection.** Similar to B-ALDA results, in this case also we used the entire pool of 300 target data points. The rest of the examples in dataset 24 were treated as test data. As earlier, our approaches perform better than the other baselines with the domain separator based O-ALDA demonstrating a slightly better AUC score and slightly lesser label complexity as compared to online ALSI. Table 5.8 presents the AUC scores and the label complexities of the various methods.

### 5.4.4 Remarks

For all datasets considered, both batch and online versions of ALDA demonstrate substantial improvement of prediction accuracy for *Sentiment* data ($\sim$ ($\mathbf{0.4\%-5.09\%}$)). This improvement is particularly high when the domains are reasonably similar (for example, *Kitchen→Apparel* in Table 5.4 and Table 5.7). In addition, the *Landmine* data reports AUC scores, not accuracies, and 1% increase in AUC score implies substantial improvement.

For *Sentiment* and *Landmine* datasets, both ALSI and ALDA (i.e., B-ALDA and O-ALDA) demonstrate improvement in *prediction accuracy for a fixed label budget* when compared to other baselines. Apart from the results for *DVD→Books* in the batch setting (Table 5.3), the prediction accuracies obtained by ALSI and ALDA in all other cases are comparable. However, to get a better sense of the robustness of these two approaches,

**Table 5.6**. Classification accuracies (**Acc**), standard deviations (**Std**) and number of labels requested for *DVD→Books*. Results are averaged over 20 runs (w.r.t. different permutations of the training data). Note: ID, sDA and FEDA are given labels of all examples in the target pool.

| **Met-hod** | **Target Label Budget** | | | | |
|---|---|---|---|---|---|
| | **1000** | **2000** | **3000** | **4000** | **5000** |
| | **Acc(±Std)** | **Acc(±Std)** | **Acc(±Std)** | **Acc(±Std)** | **Acc(±Std)** |
| ID | 65.94(±3.40) | 66.66(±3.01) | 67.00(±2.40) | 65.72(±3.98) | 66.25(±3.18) |
| sDA | 66.17(±2.57) | 66.45(±2.88) | 65.31(±3.13) | 66.33(±3.51) | 66.22(±3.05) |
| FEDA | 67.31(±3.36) | 68.47(±3.15) | 68.37(±2.72) | 66.95(3.11) | 67.13(±3.16) |
| ALZI | 66.24(±3.16) | 66.72(±3.30) | 63.97(±4.82) | 66.28(±3.61) | 66.36(±2.82) |
| ALRI | 51.79(±4.36) | 53.12(±4.65) | 55.01(±4.20) | 57.56(±4.18) | 58.57(±2.44) |
| **ALSI** | **68.22(±2.17)** | **69.65(±1.20)** | **69.95(±1.55)** | 70.54(±1.42) | **70.97(±0.97)** |
| **O-ALDA** | 67.64(±2.35) | 68.89(±1.37) | 69.49(±1.63) | **70.55(1.15)** | 70.65(±0.94) |

**Table 5.7**. Classification accuracies (**Acc**), standard deviations (**Std**) and number of labels requested for *Kitchen→Apparel*. Results are averaged over 20 runs (w.r.t. different permutations of the training data). Note: ID, sDA and FEDA are given labels of all examples in the target pool.

| **Met-hod** | **Target Label Budget** | | | | |
|---|---|---|---|---|---|
| | **1000** | **2000** | **3000** | **4000** | **5000** |
| | **Acc(±Std)** | **Acc(±Std)** | **Acc(±Std)** | **Acc(±Std)** | **Acc(±Std)** |
| ID | 69.64(±3.14) | 69.61(±3.17) | 69.36(±3.14) | 69.77(±3.58) | 70.77(±3.05) |
| sDA | 69.70(±2.57) | 70.48(±3.42) | 70.29(±2.56) | 70.86(±3.16) | 70.71(±3.65) |
| FEDA | 70.05(±2.47) | 69.34(±3.50) | 71.22(±3.00) | 71.67(±2.59) | 70.80(±3.89) |
| ALZI | 70.09(±3.74) | 69.96(±3.27) | 68.6 (±3.94) | 70.06(±2.84) | 69.75(±3.26) |
| ALRI | 52.13(±5.44) | 56.83(±5.36) | 58.09(±4.09) | 59.82(±4.16) | 62.03(±2.52) |
| **ALSI** | 73.82(±1.47) | **74.45(±1.27)** | 75.11(±0.98) | 75.35(±1.30) | **75.58(±0.85)** |
| **O-ALDA** | **73.93(±1.84)** | 74.18(±1.85) | **75.13(±1.18)** | **75.88(±1.32)** | **75.58(±0.97)** |

we compare the number of mistakes made by the online variants of these two approaches during the training phase. Table 5.9 presents the results for *Sentiment* data. As can be seen, in almost all case the number of mistakes made by O-ALDA is much less (almost half in many cases) than online ALSI. Hence, irrespective of the nearness or farness of the source-target domain pairs, ALDA is a better choice compared to ALSI.

## 5.5 Summary

In this work, we have considered a domain adaptation setting, and presented a framework that helps leverage interdomain information transfer while performing active learning

**Table 5.8**. AUC scores (**AUC**), standard deviation (**Std**) and labels requested (**Lab**) for the *Landmine* dataset. Results are averaged over 20 runs.

| **Method** | **Target Budget (300)** AUC±Std (Lab) |
|---|---|
| ID | 0.57±0.03 (-) |
| sDA | 0.60±0.02 (-) |
| FEDA | 0.52±0.04 (-) |
| ALZI | 0.61±0.02 (284) |
| ALRI | 0.56±0.05 (229) |
| **ALSI** | 0.65±0.02 (244) |
| **O-ALDA** | **0.67±0.03 (241)** |

**Table 5.9**. Number of mistakes made by ALSI and O-ALDA for *Sentiment* data.

| | **Target Label Budget** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1000** | **2000** | **3000** | **4000** | **5000** | **1000** | **2000** | **3000** | **4000** | **5000** |
| | **Number of Mistakes** | | | | | | | | | |
| **Method** | *DVD→Books* | | | | | *Kitchen→Apparel* | | | | |
| ALSI | **369** | **739** | 1117 | 1460 | 1816 | 245 | 532 | 810 | 1097 | 1088 |
| O-ALDA | 384 | 741 | **1000** | **1012** | **1004** | **232** | **478** | **549** | **551** | **556** |

in the target. Both the batch and online versions of the proposed ALDA empirically demonstrate the benefits of domain transfer for active learning.

At present, ALDA is oblivious to the feature set used and as such, does not depend on domain knowledge and feature selection. It takes all features into consideration. Nonetheless, it is possible that in the feature space not all features contribute equally while transferring information from source to target and without a priori information about the source and target domains, it is difficult to assess which features might maximally benefit the transfer of parameters from source to target. However, if prior domain knowledge about the target is available from related source domains, then one can potentially leverage active learning to selectively choose *only* those features that transfer maximum information between the two domains.

An alternative approach to leverage feature information for ALDA is to perform active learning on features. There exists work in active learning that queries labels for features (Druck et al., 2009) and, in some cases, queries labels for both instances and features in tandem (Raghavan et al., 2006). We note that this is different from the above where active learning can essentially be used as a tool for feature selection. In this case, active strategies query labels that exploit both instance and feature informativeness (for e.g., in NLP, consider querying labels for rare words which serve as informative features in the target domain). It would be interesting to extend the proposed ALDA to perform active domain transfer by querying labels of both instances and features.

# PART II

# DISTRIBUTED LEARNING

# CHAPTER 6

# A NEW MODEL FOR DISTRIBUTED LEARNING

Distributed learning (Bekkerman et al., 2011) is the study of machine learning on data distributed across multiple locations. Examples of this setting include data gathered from sensor networks, or from data centers located across the world, or even from different cores on a multicore architecture. In all cases, the challenge lies in solving learning problems with minimal *communication* overhead between nodes; learning algorithms cannot afford to ship all data to a central server, and must use limited communication efficiently to perform the desired tasks. In this chapter, we introduce a framework for studying distributed classification that treats internode communication as a limited resource, and present a number of algorithms for this problem that uses internode interaction to reduce communication. Our main technique is the use of carefully chosen data and classifier descriptors that convey the most useful information about one node to another; in that respect, our work makes use of (in spirit) the *active learning* paradigm (Settles, 2009).

For distributed classification, the dominant strategy (Lazarevic and Obradovic, 2001, Mann et al., 2009, McDonald et al., 2010, Predd et al., 2006) is to design local classifiers that work well on individual nodes. These classifiers are then communicated to a central server, and then aggregation strategies like voting, averaging, or even boosting are used to compute a global classifier. These approaches, while designed to improve communication, do not study communication as a resource to be used sparingly, and ignore the fact that interactions between nodes might reduce communication even further by allowing them to learn from each others' data.

Existing work in distributed learning mainly focuses on either inferring an accurate global classifier from multiple distributed subclassifiers learned individually (at respective nodes) or on improving the efficiency of the overall learning protocol. The first line of work consists of techniques like *parameter mixing* (Mann et al., 2009, McDonald et al., 2010) or *averaging* (Collins, 2002) and classifier *voting* (Bauer and Kohavi, 1999). Parameter mixing (or averaging (Collins, 2002)), which has been primarily proposed for maximum entropy (MaxEnt) models (Mann et al., 2009) and structured perceptrons (Collins, 2002, McDonald et al., 2010), have shown to admit convergence results but lack any bounds on the communication. Indeed, parameter-mixing for structured perceptrons uses an iterative strategy that require a large amount of communication. Additionally, we have shown that if the different classifiers are only allowed to train on mutually exclusive data subsets then there exists specific examples (under the adversarial model) where voting will *always* yield suboptimal results. We have presented such examples in section 6.4. Thus, these approaches do admit convergence results but lack any bounds on the communication. The goal of the second line of work is to make distributed algorithms scale to large datasets. Many of these works (Chu et al., 2007a, Teo et al., 2010) focus on MapReduce. Zinkevich et al. (2010) proposed a MapReduce based improved parallel stochastic gradient descent and more recently Servedio and Long (2011) improved the time complexity of $\gamma$-margin parallel algorithms from $\Omega(1/\gamma^2)$ to $O(1/\gamma)$. Dekel et al. (2010b) averaged over minibatches of accumulated gradients to improve regret bounds for distributed online settings. Duchi et al. (2010) and Agarwal and Duchi (2011) considered optimization in distributed settings but their convergence analysis applied to specific cases of subgradient and stochastic gradient descent algorithms.

Surprisingly, communication in learning has not been studied directly as a resource to be used sparingly. As we show in this chapter, intelligent interaction between nodes, communicating key data subsets not just its classification, can greatly reduce the necessary communication over existing approaches. On large distributed systems, communication has become a major bottleneck for many real-world problems; it accounts for a large percentage of total energy costs, and is the main reason that MapReduce algorithms are designed to minimize rounds (of communication). This strongly motivates the need to incorporate the study of this aspect of an algorithm directly, as presented and modeled in this chapter.

Independently of this work, research by Balcan et al. (2012) study a very similar model. They also consider adversarially distributed data among $k$ parties and attempt to learn on the adversarially distributed data while minimizing the total communication between the parties. Similar to this work, the work of Balcan et al. (2012) presents both agnostic and nonagnostic results for generic settings, and shows improvements over sampling bounds in several specific settings including the $d$-dimensional linear classifier problem we consider here (also drawing inspiration from boosting). In addition, their work provides total communication bounds for decision lists and for proper and nonproper learning of parity functions. They also extend the model so as to preserve differential and distributional privacy while conserving total communication, as a resource, during the learning process.

Our overall contribution, in this work, is to model communication minimization (in distributed classification) as an active probing problem. We start in section 6.2.1.2 by showing that, within our proposed framework, the one-way communication problem can be solved trivially under i.i.d. assumptions (ref. section 6.2.1.2). Hence, in this work, most of our effort is focused on *adversarial distributions*. In all subsequent cases, we first help build intuition by discussing a two-party protocol and thereafter extend the two-party results to the $k$-party case. In section 6.2.1.3 we show that, for *one-way* communication, it is possible to learn optimal global classifiers *exactly* (i.e., with 0-error) for thresholds (in $\mathbb{R}^1$), intervals (in $\mathbb{R}^1$) and axis-aligned rectangles (in $\mathbb{R}^d$) with only a *constant* amount of communication. For the case of linear separators, we prove an $\Omega(1/\varepsilon)$ lower bound (ref. Appendix B.1). Thereafter in section 6.2.2, we present our *two-way, two-party* communication protocol ITERATIVESUPPORT which learns an $\varepsilon$-error classifier (under adversarial distributions) using *only* $O(\log 1/\varepsilon)$ communication – *an exponential improvement over the one-way case!* A $O(d^2 \log 1/\varepsilon)$ protocol based on multiplicative-weight-update for learning in arbitrary dimension follows. Next, in section 6.3, we use the results of section 6.2.2 to obtain an $O(k^2 \log 1/\varepsilon)$ bound for $k$-parties using two-way communication in 2-dimensions and $O(kd^2 \log 1/\varepsilon)$ bound using a boosting-based algorithm in higher dimensions. In section 6.4, we present empirical results that demonstrate the correctness and convergence of the linear separator algorithms and also compare its performance with a few other baselines.

Table 6.1 summarizes the results obtained with references to appropriate sections of this chapter. All our results pertain to the noiseless setting which assumes the existence of a classifier that perfectly separates the data. Finally, for cases when it is difficult to a priori ascertain the presence of noise, we present one-way communication lower bounds for learning in our model (ref. Appendix B.2).

**Table 6.1**. Summary of results obtained for different hypotheses classes under an *adversarial* model with one-way and two-way communications. All results are for the *noiseless* setting. $\nu$ denotes the VC-dimension for the family of classifiers.

| **Hypothesis Class** | **Dimen-sions** | **Error** | **Communication Complexity** | | **Reference** |
|---|---|---|---|---|---|
| | | | **Two-party** | **k-party** | |
| | | | One-way protocols | | |
| generic | $d$ | $\varepsilon$ | $O(\nu/\varepsilon \log \nu/\varepsilon)$ | $O(k(\nu/\varepsilon)\log \nu/\varepsilon)$ | Th 6.2.2 & 6.3.1 |
| thresholds | 1 | 0 | 2 | $2k$ | Lem 6.2.1 & 6.3.2 |
| aa-rectangles | $d$ | 0 | $4d$ | $4dk$ | Th 6.2.3 & 6.3.2 |
| hyperplanes | $d$ | $\varepsilon$ | $\Omega(1/\varepsilon)$ | $\Omega(k/\varepsilon)$ | Th 6.2.4 & 6.2.5 |
| | | | Two-way protocols | | |
| hyperplanes | 2 | $\varepsilon$ | $O(\log 1/\varepsilon)$ | $O(k^2 \log 1/\varepsilon)$ | Th 6.2.6 & 6.3.5 |
| hyperplanes | $d$ | $\varepsilon$ | $O(d^2 \log 1/\varepsilon)$ | $O(kd^2 \log 1/\varepsilon)$ | Th 6.2.7 & 6.3.7 |

## 6.1 Proposed Communication-efficient Model

There are many aspects to formalizing the problem of learning classifiers with limited communication, including discussion of the data sources (i.i.d. or adversarial), data quality (noiseless or noisy), communication models (one-way, two-way or k-way) and classifier models (linear, nonlinear, mixtures). In this work, we focus on a simple core model that illustrates both the challenges and the benefits of focusing on the communication bottleneck.

In our model, we first consider one-way and two-way communication between *two* parties Alice (say, $A$) and Bob (say, $B$) that receive *noiseless* data sets $D_A$ and $D_B$ that result from partitioning a larger data set $D = D_A \cup D_B$. Thereafter, we consider one-way and two-way communication between $k$ parties $P_1, P_2, \ldots, P_k$ that receive noiseless data sets $D_1, D_2, \ldots, D_k$ partitioned from $D = \bigcup_{i=1}^{k} D_i$. In either case, the partitioning may be done randomly, but might also be adversarial: indeed, a number of recent discussions (Cesa-Bianchi et al., 2009, Dekel et al., 2010a, Hsu and Langford, 2011, Laskov and Lippmann, 2010) highlight the need to consider adversarial data in learning scenarios.

In our model, the nodes learn together (via communication), a classifier $h_k$ ($h_{AB}$ for two nodes $A$ and $B$) from a family of classifiers such as linear classifiers. Let $h^*$ denote the optimal classifier that can be learned on $D$. Let $E_D(h)$ denote the number of points misclassified by some classifier $h$ on $D$. We say that $h_k$ has *$\varepsilon$-approximation error* ($\varepsilon$-error for short) on $D$ if

$$E_D(h_k) - E_D(h^*) \leq \varepsilon|D|$$

*The goal is for $h_k$ to have at most $\varepsilon$-error ($0 < \varepsilon < 1$) while minimizing internode communication.*

In our proposed model, we phrase the learning task in terms of training error, rather than generalization. This is motivated by numerous results that indicate that low training error combined with limits on the hypothesis class used lead to good generalization bounds (Kearns and Vazirani, 1994).

We assume that there are $k$ parties $P_1, P_2, \ldots P_k$. Each party $P_i$ possesses a dataset $D_i$ that no other party has access to, and each $D_i$ may have both positive and negative examples. The goal is to classify the full dataset $D = \cup_i D_i$ correctly. We assume that there exists a perfect classifier $h^*$ from a family of classifiers $\mathcal{H}$ with associated range space $(D, \mathcal{H})$ and bounded VC-dimension $v$. We are willing to allow $\varepsilon$-classification error on $D$ so that up to $\varepsilon|D|$ points in total are misclassified.

Each *word* of data (e.g., a single point or vector in $\mathbb{R}^d$ counts as $O(d)$ words) passed between any pair of parties is counted towards the total communication; this measure in words allows us to examine the cost of extending to $d$-dimensions, and allows us to consider communication in forms other than example points, but does not hinder us with precision issues required when counting bits. For instance, a protocol that broadcasts a message of $M$ words (say $M/d$ points in $\mathbb{R}^d$) from one node to the other $k-1$ players costs $O(kM)$ communication. The goal is to design a protocol with as little communication as possible. We assume an *adversarial* model of data distribution; in this setting we prepare for the worst, and allow some *adversary* to determine which player gets which subset of $D$.

## 6.2 Two-party Protocols

### 6.2.1 One-way communication

**6.2.1.1 Sampling bounds.** Given $D$ and a family of classifiers with bounded VC-dimension $v$, a random sample from $D$ of size

$$s_{\varepsilon,v} = O(\min\{(v/\varepsilon)\log(v/\varepsilon), v/\varepsilon^2\}) \tag{6.1}$$

has at most $\varepsilon$-classification error on $D$ with constant probability, as long as there exists a perfect classifier. Throughout this paper we will assume that a perfect classifier exists. This constant probability of success can be amplified to $1-\delta$ with an extra $O(\log(1/\delta))$ factor of samples.

**6.2.1.2 Random partitioning.** We first consider the case when the data is partitioned randomly among nodes. Specifically, each node $i$ can view its data $D_i$ as being drawn from the same distribution $D \subset \mathbb{R}^d$. That is, all datasets $D_i$ are identically distributed. We can now apply learning theory results for any family of classifiers $\mathcal{H}$ with bounded VC-dimension $\nu$. Any classifier $h_S \in \mathcal{H}$ which perfectly separates a random sample $S$ of $s_{\varepsilon,\nu} = O(\min\{(\nu/\varepsilon)\log(\nu/\varepsilon), \nu/\varepsilon^2\})$ samples from $D$ has at most, $\varepsilon$-classification error on $D$ with constant probability (Anthony and Bartlett, 2009). Thus, each $D_i$ can be viewed as such a sample $S$ and if $D_i$ is large enough, with no communication a node can return a classifier with small error as long as there exists a perfect classifier. Throughout this work we will assume that a perfect classifier exists.

**Theorem 6.2.1** *Let $\{D_1,\ldots,D_k\}$ randomly partition $D \subset \mathbb{R}^d$. In the noiseless setting a node i can produce a classifier from $(\mathbb{R}^d,\mathcal{H})$ (with VC-dimension $\nu$) with at most $\varepsilon$-error for $\varepsilon = O((\nu/|D_i|)\log(\nu|D_i|))$, with constant probability.*

This constant probability of success can be amplified to $1-\delta$ with an extra $O(\log(1/\delta))$ factor of samples.

A similar result (with slightly worse dependence on the $D_i$) can be obtained for the noisy setting. These results indicate that the $k$-party (and hence also two-party) setting is trivial to solve if we assume random partitioning of $D$. Thus, for the remainder of the chapter we focus on protocols for adversarially partitioned data.

**6.2.1.3 Adversarial partitioning.** We now turn to data adversarially partitioned between two nodes $A$ and $B$, as disjoint sets $D_A$ and $D_B$, respectively. For the hypothesis classes discussed in this section, *one-way protocols* where only $A$ sends data to $B$ suffices for $B$ to learn an $\varepsilon$-error classifier. Consider first a generic setting, with $D \subset \mathbb{R}^d$ and family of hypothesis $\mathcal{H} \subset 2^D$ so $(\mathbb{R}^d,\mathcal{H})$ has VC-dimension $\nu$.

**Theorem 6.2.2** *Assume there exists a 0-error classifier $h^* \in \mathcal{H}$ on $D$ where $(D,\mathcal{H})$ has VC-dimension $\nu$. Then $A$ sending $s_\varepsilon = O((\nu/\varepsilon)\log(\nu/\varepsilon))$ words ($S_A \subset D_A$) to $B$ allows $B$ to, with constant probability, produce an $\varepsilon$-error classifier $h \in \mathcal{H}$.*

**Proof.** The classifier returned by $B$ will have 0 error on $D_B \cup S_A$; thus it only has error on $D_A$. Since $S_A$ is an $\varepsilon$-net of $D_A$ with constant probability, then it has at most $\varepsilon$-error on $D_A$ and hence, at most $\varepsilon$-error on $D_A \cup D_B = D$. ■

A similar result with $s_\varepsilon = O(\nu/\varepsilon^2)$ applies to the noisy setting. An important technical contribution of this work is to show that in many cases we can improve upon these general results.

**6.2.1.4 Results for basic geometric hypotheses families.** Here, we present communication bounds for the class simple geometric families.

First we describe how to find a threshold $t \in \mathcal{T} \subset \mathbb{R}$ such that all points $p \in D$ with $p < t$ are positive and with $p > t$ are negative. $A$ sends to $B$ a set $S_A$ consisting of two points in $D_A$: its largest positive point $p^+$ and its smallest negative point $p^-$. Then $B$ returns a 0-error classifier on $D_B \cup S_A$.

**Lemma 6.2.1** *In $O(1)$ words one-way communication we can find a 0-error classifier in $(D, \mathcal{T})$.*

**Proof.** The optimal classifier $t \in \mathcal{T}$ must lie in the range $[p^+, p^-]$ otherwise, it would misclassify some point in $D_A$, breaking our noiseless assumption. Then any 0-error classifier on $D_B$ within this range has 0 error on $D$. ■

We can now apply Lemma 6.2.1 to get stronger bounds. In particular, this generalizes to the family $\mathcal{I}$ of intervals in $\mathbb{R}^1$. First $A$ finds $h_A$, its optimal classifier for $D_A$. This interval has two end points each of which lies in between a pair of a positive and a negative point (if there are no negative or no positive points, $A$ returns the empty set). These two pairs of points form a set $S_A$ that $A$ sends to $B$. $B$ now returns the classifier that optimally separates $D_B \cup S_A$, and if $S_A$ is empty then the interval classifier is as small as possible.

**Lemma 6.2.2** *In $O(1)$ words one-way communication we can find a 0-error classifier $h \in \mathcal{I}$.*

**Proof.** When $S_A$ is nonempty, this encodes two versions of Lemma 6.2.1. Assume without loss of generality that the positive points are contained in an interval with negative points lying outside the interval. Then we can pick any positive point $p$ from either set $D_A$ or $D_B$ and consider the points greater than $p$ in the first instance of Lemma 6.2.1 and points less than $p$ in the second instance. Invoking Lemma 6.2.1 proves this case. When $S_A$ is empty, and a perfect classifier exists, then the minimal separating interval on $D_B$ will not violate any points in $S_A$, and will have no error. ■

We now consider finding a 0-error classifier from the family $\mathcal{R}^d$ of all axis-aligned rectangles in $\mathbb{R}^d$. An axis-aligned rectangle $R \in \mathcal{R}^d$ can be defined by $d$-values in $\mathbb{R}^d$, a minimum and maximum value along each coordinate axis. Given a data set $P$, the *minimum axis-aligned rectangle* for $P$ is the smallest axis-aligned rectangle that contains all of $P$; that is, it has the smallest maximum coordinate possible along each coordinate axis and the largest minimum coordinate possible along each coordinate axis. These $2d$ terms can be optimized independently as long as $P$ is nonempty.

For a dataset $D_A$ we can define two minimum axis-align rectangles, $R_A^+$ and $R_A^-$ defined on the positive and negative points, respectively. If the positive or negative point set is empty, then each coordinate minimum and maximum is set to a special character $\emptyset$. Two such rectangles can be defined for $D_B$ and $D = D_{A\cup B}$ in the same way.

**Theorem 6.2.3** *A one-way protocol where A sends $R_A^+$ and $R_A^-$ to B is sufficient to find a* 0*-error classifier $h_{AB} \in \mathcal{R}^d$ in the noiseless setting. It requires $O(d)$ words of communication.*

**Proof.** The key observation is that the minimum axis-aligned rectangle that contains $R_A^+$ and $R_B^+$ is precisely $R_{A\cup B}^+$ (and symmetrically for negative points). Since the minimum and maximum for each coordinate axis is set independently, then we can optimize each using that value from $R_A^+$ and $R_B^+$. Thus, $B$ can compute this using points from $D_B$ and $R_A^+$.

First, consider the case where positive points are inside the classifier and negative points are outside. Since there exists a 0-error classifier $h^*$, then $R_{A\cup B}^+$ must be contained in that classifier, since no smaller classifier can contain all positive points. It follows by our assumption that $h^*$ and thus also $R_{A\cup B}^+$, contains no negative points and can be returned as our 0-error classifier $h_{AB}$. $B$ can determine if positive or negative points are inside by which of $R_{A\cup B}^+$ and $R_{A\cup B}^-$ is smaller. If $R_A^+$ or $R_A^-$ is $\emptyset$, then $R_{A\cup B}^+ = R_B^+$ or $R_{A\cup B}^- = R_B^-$, respectively. ■

**6.2.1.5 An $\Omega(1/\varepsilon)$ lower bound for linear separators in $\mathbb{R}^2$.** The positive results from simpler geometric concepts do not extend to hyperplanes. We prove the following lower bound in Appendix B.1.

**Theorem 6.2.4** *Using only one-way communication from A to B, it requires $\Omega(1/\varepsilon)$ words of communication to find an $\varepsilon$-error linear classifier in $\mathbb{R}^2$.*

Note that due to Theorem 6.2.1, this is tight up to a $\log(1/\varepsilon)$ factor for one-way communication.

We can extend this lower bound to the $k$-node one-way model of computation where we assume each node $P_i$ can only send data to $P_{i+1}$. In this case, we give node $A$'s input to $P_1$, and node $B$'s input to node $P_k$, and nodes $P_i$ for $i \in [2, k-1]$ have no data. Then each node $P_i$ is forced to send the $\Omega(1/\varepsilon)$ communication that $A$ wants to send to $B$ along the chain.

**Theorem 6.2.5** *Using only one-way communication among k-players in a chaining model, it requires $\Omega(k/\varepsilon)$ words of communication to find an $\varepsilon$-error linear classifier in $\mathbb{R}^2$.*

### 6.2.2 Median-based two-way protocol for linear separators in $\mathbb{R}^2$.

In this section, we present a two-party algorithm that uses two-way communication to learn an $\varepsilon$-optimal combined classifier $h_{AB}$. We prove an $O(\log(1/\varepsilon))$ bound on the communication required.

**6.2.2.1 Algorithm.** For ease of exposition, we first provide an overview of the algorithm. Thereafter we discuss the details and provide proofs to bound the communication.

Our algorithm proceeds in rounds. In each round both nodes send a constant number of points to the other. The goal is to limit the number of rounds to $O(\log(1/\varepsilon))$, resulting in a total communication complexity of $O(\log(1/\varepsilon))$. At the end of $O(\log(1/\varepsilon))$ rounds of communication, the algorithm yields a combined classifier $h_{AB}$ that has $\varepsilon$ error on $D$.

In order to bound the number of rounds, each node must maintain information about which points the *other node* might be classifying correctly, or not, at any stage of the algorithm. Specifically, suppose node $A$ is sent a classifier $h_B$ from node $B$ (learned on $D_B$ and hence has zero error on $D_B$) and this classifier misclassifies some points in $D_A$. We denote these points as the *Set of Disagreement* (*SOD*) where $SOD \subseteq D_A$. The remaining points in $D_A$ can be divided into the *Set of Total Agreement* (*SOTA*), which are the points on which classifiers from $A$ and $B$ will continue to agree on in the future, and the *Set of Luck* (*SOL*), which are points on which the two nodes currently agree, but might disagree later on. The set of disagreement and the set of luck together form the *Set of Uncertainty* $SOU = SOD \cup SOL$, representing all points that may or may not be classified incorrectly by $B$ in the future.

Our goal will be to show that the *SOU* decreases in cardinality by a constant factor in each round. Achieving this will guarantee that at the end of $\log(1/\varepsilon)$ rounds, the size of the *SOU* will be at most an $\varepsilon$-fraction of the total input. Since $|SOU| \geq |SOD|$, we obtain the desired $\varepsilon$-error classifier.

The simplest strategy would be for each node to build a max-margin classifier on all points it has seen thus far, and send the support points for this classifier to the other node. While this simple protocol might converge quickly in practice (we actually compare against it in section 6.4, it is called MAXMARG, and it often does), in principle this protocol may take a linear number of rounds to converge. Thus, our algorithm will choose non-max-margin support vectors, but we will show that by sending these points we can achieve provable error and communication trade-off bounds.

Let $\mathcal{P}_A^+$ and $\mathcal{P}_A^-$ denote polytopes that contain positive and negative points in $D_A$, respectively. Let $\mathcal{C}_A^+$ and $\mathcal{C}_A^-$ denote the convex hulls formed by the positive and negative *SOTA* in $D_A$ after the $i^{\text{th}}$ round, respectively. In general, when sets have a $+$ or $-$ superscript it will denote the restriction of that set to only positive or negative points, respectively. Often to simplify messy, but usually straightforward, technical details we will drop the superscript and refer to either or both sets simultaneously. We denote the *region of uncertainty* $\mathcal{U}_A$ as $\mathcal{P}_A \setminus \mathcal{C}_A$, and note $U_A = \mathcal{U}_A \cap D_A$.

In each round $A$ will send to $B$ a set $S_A \subset D_A$; these points imply a max-margin classifier $h_A$ on $S_A$ that has 0 error on $D_A$; see Figure 6.1. Then $B$ will either terminate with an $\varepsilon$-error classifier $h_B$, or symmetrically return a set of points $S_B \subset D_B$. This process is summarized in Algorithm 4.

Two aspects remain: determining if a player may exit the protocol with a $\varepsilon$-error classifier (early termination), and computing the support points in the function SUPPORT.

Note that in Algorithm 4, under certain *early-termination* conditions, player $B$ may

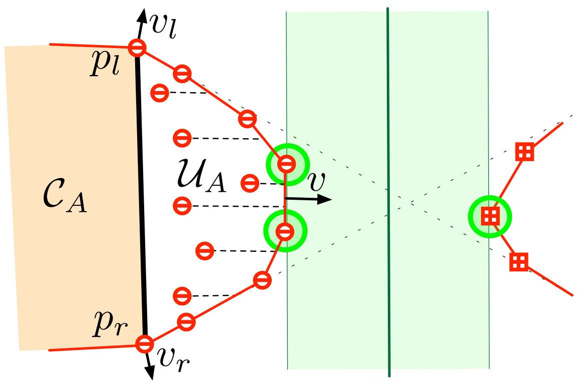


**Figure 6.1**. 3 support points chosen from $U_A$, and the family of 0-error classifiers for $A$ parallel to $h_A$.

**Algorithm 4** ITERATIVESUPPORTS

```
Input: $D_A$ and $D_B$
Output: $h_{AB}$ (classifier with $\varepsilon$-error on $D_A \cup D_B$)
$S_A :=$ SUPPORT$(D_A)$; send $S_A$ to $B$;
while (1) do
    ——— B's move ———
    compute error (err) using $h_A$ (from $S_A$) on $D_B$;
    if(err $\leq \varepsilon|D_B|$) then exit;
    $D_B = D_B \cup S_A$; $S_B :=$ SUPPORT$(D_B)$; send $S_B$ to $A$;
    ——— A's move ———
    compute error (err) using $h_B$ (from $S_B$) on $D_A$;
    if(err $\leq \varepsilon|D_A|$) then exit;
    $D_A = D_A \cup S_B$; $S_A :=$ SUPPORT$(D_A)$; send $S_A$ to $B$;
end while
```

terminate the protocol and return a valid classifier, even if $h_A$ has more than $\varepsilon$ error on $D_B$. Any classifier that is parallel to $h_A$ and is shifted less than the margin of the max-margin classifier also has 0 error on $D_A$. Thus, if any such classifier has at most $\varepsilon$-error on $D_B$, player $B$ can terminate the algorithm and return that classifier.

This early-termination observation is important because it allows $B$ to send to $A$ information regarding a 0-error classifier, with respect to $h_A$, and the points $S_A$ that define it. If $B$ cannot terminate, then either some point in $D_B$ must be completely misclassified by all separators within the margin, or some negative point in $D_B$ and some positive point in $D_B$ must both be in the margin and cannot be separated; see Figure 6.2. Either scenario implies that any $\varepsilon$-error classifier on $D_B$ must rotate in some direction (either clockwise or counter-clockwise) relative to $h_A$. This is important because it informs $A$ that all points on $\partial\mathcal{P}_A$ (the boundary of $\mathcal{P}_A$) in the clockwise (resp. counter-clockwise) direction from $S_A$ will never be misclassified by $B$ if $h_B$ rotates in the counterclockwise (resp. clockwise) direction from $h_A$, increasing the SOTA, and decreasing the SOU. This logic is formalized in Lemma 6.2.3.

If the set $S_A$ always has half of $U_A$ on either side, then this process will terminate in, at most, $O(\log(1/\varepsilon))$ rounds. However, it may have no points on one side, and always be forced to rotate towards the other side. Thus, the set $S_A$ is chosen judiciously to ensure that $|U_A|$ decreases by at least half each round.

What remains to describe is how $A$ chooses a set $S_A$, i.e., how to implement the subroutine SUPPORT in Algorithm 4. If the set $S_A$ always has half of $U_A$ on either side, then this process will terminate in, at most $O(\log(1/\varepsilon))$ rounds, via the consequences of no early-termination. However, if no points are on one side of $S_A$, and $B$'s response always forces $h_A$ to rotate towards the other side, then this cannot be assured. Thus, the set $S_A$ should be chosen judiciously to ensure that $|U_A|$ decreases by at least half each round.

We present two methods to choose $S_A$. This first does not have the half-on-either-side guarantee, but is a very simple heuristic, and which we show in section 6.4 often works quite well, even in higher dimensions. The second, is only slightly more complicated and is designed precisely to have this half-on-either-side guarantee. Both methods start by computing the region of uncertainty $\mathcal{U}_A$ and the set of its points $D_A$ which lie in that region $U_A$.

The first is called MAXMARG, and simply chooses the max-margin support points as $S_A$. These points may include points sent over in previous iterations from $B$ to $A$.

The second is called MEDIAN, and is summarized in Algorithm 5 (shown from $A$'s perspective). It projects all of $U_A$ onto $\partial\mathcal{P}_A$ (the boundary of $\mathcal{P}_A$); this creates a weight for each edge of $\partial\mathcal{P}_A$, defined by the number of points projected onto it. Then MEDIAN chooses the weighted median edge $E$. Finally, the orientation of $h_A$ is set parallel to edge $E$, and the corresponding support vectors are constructed.

**6.2.2.2 Analysis of ITERATIVESUPPORTS.** Now, we formally prove the number of rounds required by ITERATIVESUPPORTS to converge.

To simplify the exposition of the protocol, we start with a special case, where player

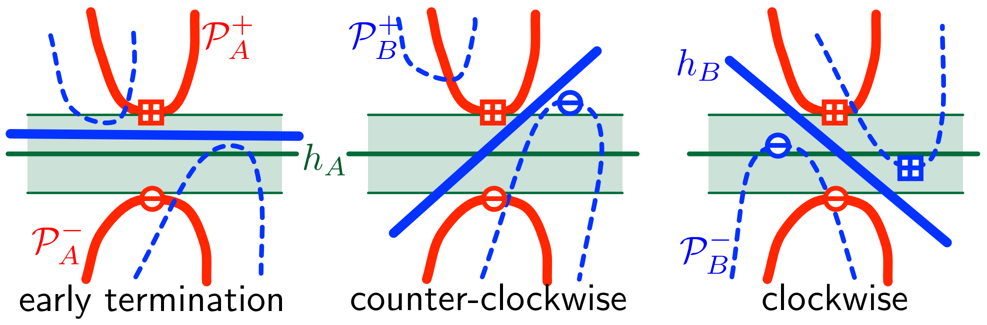


**Figure 6.2**. Cases for either early termination, or for the direction of the normal to the linear separator being forced counter-clockwise or clockwise.

**Algorithm 5** SUPPORT implemented as MEDIAN

```
1: Input: $D = D_A \cup \{S_B\}$
2: Output: $S_A$ (a set of support points)
3: project points in $U_A$ onto $\partial\mathcal{P}_A$;
4: $E$ := weighted median edge of $\partial\mathcal{P}_A$;
5: $h_A$ := classifier on $D$ parallel to edge $E$;
6: $S_A$ := support points of $h_A$;
```

$A$ must, through interaction with $B$, teach $B$ parameters of classifier that has at most $\varepsilon$ error on $D_A^-$, as well as some (but not all) negative examples in $D_B$. This case captures the bulk of the technical development of the overall protocol. In section 6.2.2.3 we will then describe how to extend the protocol to (a) ensure at most $\varepsilon$ error on both positive and negative examples in $D_A$, and (b) be symmetric: have at most $\varepsilon$ error on $D_A \cup D_B$.

We will describe the protocol from the point of view of player $A$. Each round of communication will start with $A$ computing a classifier from its current state, and sending support points for this classifier to $B$. $B$ then performs some computation, and either terminates returning an $\varepsilon$-error classifier, or returns a single bit of information to $A$. $A$ updates its internal state, completing the round.

At any stage, $A$ maintains an interval of directions $(v_l, v_r) \subset \mathbb{S}^1$ where by convention, we go clockwise from $v_l$ to $v_r$. This interval represents $A$'s current bound on the possible directions normal to an $\varepsilon$-optimal classifier based on all conversation with $B$ up to this point. $A$ also maintains $\mathcal{C}_A$ (recall that $\mathcal{C}_A$ is the convex hull of the *SOTA*) as well as the set of points $U_A$ that form the *SOU*. By Lemma 6.2.4, we know that $\mathcal{P}_A = \mathcal{C}_A \cup \mathcal{U}_A$, and therefore, there exist a pair of points $\{p_l, p_r\}$ on $\mathcal{P}_A$ whose supporting line segment separates $\mathcal{C}_A$ and $\mathcal{U}_A$. $A$ maintains this pair as well; in fact, $v_l$ and $v_r$ represent outward normals to $\mathcal{P}_A$ at $p_l$ and $p_r$.

*(1) A's move:* $A$ projects all points in $U_A$ onto the boundary of $\mathcal{P}_A$, denoted $\partial\mathcal{P}_A$, (the projection is orthogonal to the edge through $\{p_l, p_r\}$). Each edge in $\partial\mathcal{P}_A$ is weighted by how many points are projected to it (with boundary points being assigned arbitrarily to one of the two incident edges). We select the two points on the boundary of edge $e$, which is the weighted median, and place these points in a set $S$. The normal direction to $e$ is $v$, and the extreme positive point in $D_A$ along direction $-v$ is also placed in $S$. Now the classifier $h_A$ is the max-margin separator of $S$, has 0 error on $D_A$, and is parallel to $e$. Then $A$ sends $(v_l, v_r, v, S)$ to $B$.

*(2) B's move:* $B$ receives $(v_l, v_r, v, S)$ from $A$. It then determines whether there exists a classifier $h_B$ with normal $v$ within the margin defined by $S$ that correctly classifies all but an $\varepsilon$-fraction of points in $B$. If so, $B$ sends $(h_B, 0)$ to $A$ and terminates, returning $h_B$.

Suppose that such a classifier does not exist. Then by Lemma 6.2.3, any 0-error classifier for $D_B$ must have a normal either in the interval $(v_l, v)$ or $(v, v_r)$. If the former, $B$ returns $(+1)$ to $A$, else it returns $(-1)$.

*(3) A's update:* If $A$ receives $(h, 0)$ from $B$, the protocol has terminated, returning $h$. If $A$ receives $(+1)$, it then updates its interval of directions to be $(v_l, v)$ and sets the support pair separating $\mathcal{C}_A$ and $\mathcal{U}_A$ to $(p_l, p)$. Similarly, if it receives $(-1)$, it updates the interval of directions to $(v, v_r)$ and sets the support pair to $(p, p_r)$. In both cases, it adds $p$ to $C_A$, updating $\mathcal{C}_A$ accordingly.

In this section we provide structural results about $\mathcal{C}_A$ and prove Lemma 6.2.3 and Lemma 6.2.4. The first challenge is to reason about the set of total agreement – what points can not be misclassified. Then we can argue that $SOTA = C_A \cap D_A$. We use two technical tools, the convex hull and a pivoting argument. Let $W = \bigcup_i S_i$ be the union of all $S_i$ sent in round $i$ from $A$ to $B$.

- **Convex Hull.** Let $\mathcal{K}^- = \mathcal{C}(W^-)$ be the convex hull of all the negative points sent by the protocol so far. No negative points $p \in \mathcal{P}_A^-$ can be misclassified if $p \in \mathcal{K}^-$. So, $\mathcal{K}^- \cap \mathcal{P}_A^- \subset \mathcal{C}_A^-$. The same rule holds for positive points.
- **Pivoting.** Consider any point $q \in \mathcal{P}_A^-$. If any edge from $q$ to any point $p \in \mathcal{K}^+$ intersects $\mathcal{K}^-$, then $q$ cannot be misclassified – otherwise a classifier which was correct on $p$

(and incorrect on $q$) would have to be incorrect on some negative point in $\mathcal{K}^-$. This identifies another part of $\mathcal{P}_A^-$ as being in $\mathcal{C}_A$, intuitively the region "behind" $\mathcal{K}^-$. Note that the early-termination rotation argument, along with this pivoting rule, each round excludes from $U$ all points on one of two sides of the support points in $S$.

These rules have been explained in Figure 6.3. We now have the tools to prove the two key structural lemmas needed for our protocol.

**Lemma 6.2.3** *Consider when $B$ does not terminate. If $B$ returns $(+1)$, then $A$ can update its range to $(v_l, v)$. If $B$ returns $(-1)$, then $A$ can update its range to $(v, v_r)$.*

**Proof.** When $B$ can not produce an $\varepsilon$-error separator parallel to $h_A$ and within the margin provided by $S$, that implies for any such classifier some points from $D_B$ must be misclassified. Furthermore, $B$ can present points $Y \subset D_B$ that along with $S$ violate any classifier orthogonal to $v$. Let $y, s \in Y \cup S$ be a negative and positive point, respectively, one of which any classifier orthogonal to $v$ will misclassify. Then any linear separator classifying $s$ and $y$ correctly must intersect the edge between $s$ and $y$, and thus, must rotate from direction $v$ clockwise or counter-clockwise. This excludes directions in either $(v_l, v)$ or $(v, v_r)$ and allows $B$ to return $(+1)$ or $(-1)$, accordingly. ■

**Lemma 6.2.4** *After $A$ has updated its state (step (3)), then $\mathcal{U}_A$ is convex.*

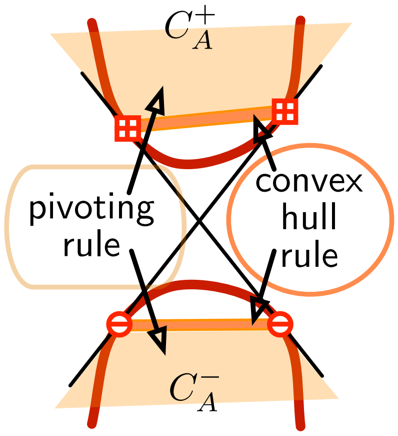


**Figure 6.3**. Illustration of convex hull and pivoting rule.

**Proof.** First consider the two negative points $\{p_l, p_r\}$. Using the convex hull rule, the edge $e_{12}$ between them is in $\mathcal{C}_A$; and because the points $\{p_l, p_r\}$ are defined as the extremal points for the range $(v_l, v_r)$ under the pivoting rule, everything "behind" them in $\mathcal{P}_A$ is also in $\mathcal{C}_A$. Thus, $\mathcal{C}_A$ is partitioned from $\mathcal{U}_A$ by the line passing through the edge $e_{12}$, implying that $\mathcal{U}_A$ is convex. ■

**6.2.2.3 Extending the basic protocol.** The simplified protocol above captures the spirit of *A*'s perspective of the algorithm on its negative points; but to show it converges, we need to extend these techniques to also handle positive points and to make it symmetric from *B*'s perspective.

In each round of the basic protocol $U_A^-$ reduces in cardinality by at least half. We now describe how to modify the protocol so that the entire set $U_A = U_A^- \cup U_A^+$ is reduced in cardinality by half. Recall that in step (1) of the basic protocol, *A* projects all points in $U_A^-$ to the boundary of $\mathcal{P}_A^-$ and determines an edge of the boundary that splits the set in half. In addition, now we project all points in $U_A^+$ to the boundary of $\mathcal{P}_A^+$ as well. We can consider the normal direction of each edge in $\partial\mathcal{P}_A^- \cap \mathcal{U}^-$ or in $\partial\mathcal{P}_A^+ \cap \mathcal{U}^+$ and map it to a point on $\mathbb{S}^1$.

We can now scan both sets of normal directions on $\mathbb{S}^1$ simultaneously by interleaving the order of directions from $\partial\mathcal{P}_A^- \cap \mathcal{U}^-$ with the antipodal directions from $\partial\mathcal{P}_A^+ \cap \mathcal{U}^+$. We again find the weighted median direction, corresponding to an edge, now among all negative and positive directions. The set $S_A$ now consists of the two points defining the median edge as well as the point incident upon the two edges with normal directions on either side of the antipodal direction of the median edge.

As before, this splits the regions of uncertainty into two convex regions on each polytope. The bit returned by *B* will guarantee that one region on each polytope will be eliminated, and by the above construction, this guarantees that we reduce the size of $U_A$ by a factor of two in each round. This has been explained in Figure 6.4.

**Lemma 6.2.5** *Over the course of a single round, the size of $U_A$ decreases by at least half.*

The basic protocol and its extension described above only reduce the *SOU* for *A*. Since *B* decides termination, it is possible that the error of the resulting classifier on *B* never reduces sufficiently. While we could run protocols in parallel for *A* and *B*, this could result

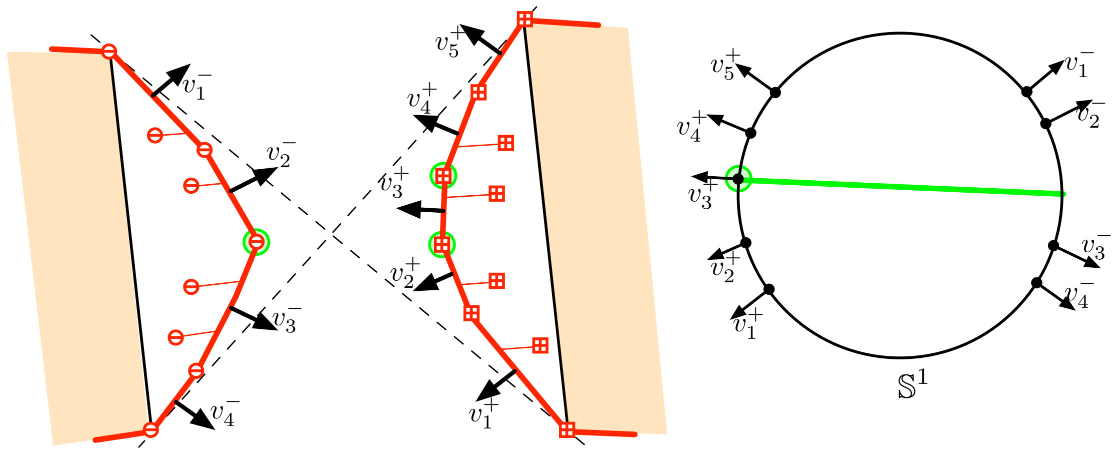


**Figure 6.4**. Extending the basic protocol.

in classifiers $h_A$ and $h_B$ that do not have $\varepsilon$-error on the *entire* data set $D_A \cup D_B$.

The solution is for $B$ to send more information back to $A$. Consider step (2) of the basic protocol. $B$ receives a support set $S_A$ from $A$, as well as the set of directions $v_l, v, v_r$ and determines which of the intervals $(v_l, v)$ and $(v, v_r)$ the direction of a 0-error classifier $h_B$ on $D_B$ must lie in. Now instead of merely sending back a bit, $B$ also sends back a support set $S_B$ corresponding to $h_B$, as well as its own directions $(v'_l, v'_r, v')$. $A$ now uses the support set $S_B$ to update its own *SOTA* and *SOU*, completing the round. Notice that now, $B$'s transmission to $A$ in step (2) of the protocol is identical to $A$'s transmission that initiates step (2)! Thus, all future separators proposed by $A$ or $B$ must correctly classify the same set of points in the full protocol transcript.

#### 6.2.2.4 Complexity analysis.

**Theorem 6.2.6** *The* 2*-player two-way protocol for linear separators always terminates in, at most,* $O(\log(1/\varepsilon))$ *rounds, using at most,* $O(\log(1/\varepsilon))$ *words of communication.*

**Proof.** By Lemma 6.2.5 we know that each round shrinks the region of uncertainty *SOU* by half of its current size for both $A$ and $B$; and we keep doing this until $|U_A| \leq \varepsilon|D_A|$ or $|U_B| \leq \varepsilon|D_B|$, then the early-termination condition must be reached. This can be achieved in $O(\log(1/\varepsilon))$ rounds. ∎

### 6.2.3 Boosting-based two-way protocol for linear separators in $\mathbb{R}^d$.

In this section we consider a randomized protocol, summarized in Algorithm 6, called WEIGHTEDSAMPLING. The Multiplicative Weight Update (MWU) routine is provided in Algorithm 7. In each round, $A$ sends a classifier $h_A$ to $B$ and $B$ responds back with a set of points $R_B$, constructed by sampling from a weighting on its points. After $T$ rounds (for $T = O(\log(1/\varepsilon))$), we will show that by voting on the result from the set of $T$ classifiers $h_A$ will misclassify at most $\varepsilon|D_B|$ points from $D_B$ while being perfect on $D_A$, and hence $\varepsilon|D_B| < \varepsilon|D_B \cup D_A| = \varepsilon|D|$, yielding a $\varepsilon$-optimal classifier as desired.

$R_B$ can construct its points in two ways: a random sample and a deterministic sample. We will focus on the randomized version since it is more practical, although it has slightly worse bounds in the two-party case. Then we will also mention and analyze the deterministic version.

**Algorithm 6** WEIGHTEDSAMPLING

```
Input: $D_A, D_B$, parameters: $0 < \varepsilon < 1$
Output: $h_{AB}$ (classifier with $\varepsilon$-error on $D_A \cup D_B$)
Init: $R_B = \{\}$; $w_i^0 = 1\ \forall x_i \in D_B$;
for t = 1 ... $T = 5\log_2(1/\varepsilon)$ do
    ——— A's move ———
    $D_A = D_A \cup R_B$;
    $h_A^t := Learn(D_A)$;
    send $h_A^t$ to $B$;
    ——— B's move ———
    $R_B$ := MWU ($D_B$, $h_A^t$, 0.75, 0.2);
    send $R_B$ to $A$;
end for
$h_{AB} = \mathsf{Majority}(h_A^1, h_A^2, \dots, h_A^T)$;
```

**Algorithm 7** MWU ($D_B$, $h_A^t$, $\rho$, $c$)

```
1: Input: $h_A^t, D_B$, parameters: $0 < \rho < 1, 0 < c < 1$
2: Output: $R_B$ (a set of $s_{c,d}$ points)
3: for all ($x_i \in D_B$) do
4:     if($h_A^t(x_i) \neq y_i$) then $w_i^{t+1} = w_i^t(1+\rho)$;
5:     if($h_A^t(x_i) == y_i$) then $w_i^{t+1} = w_i^t$;
6: end for
7: randomly sample $R_B$ from $D_B$ (according to $w^{t+1}$);
```

It remains to describe how $B$'s points are weighted and updated, which dictates how $B$ constructs the sample sent to $A$. Initially, they are all given a weight $w_1 = 1$. Then the reweighting strategy (described in Algorithm 7) is an instance of the multiplicative weight update framework; with each new classifier $h_A$ from $A$, party $B$ increases all weights of misclassified points by a $(1+\rho)$ factor, and does not change the weight for correctly classified points. We will show $\rho = 0.75$ is sufficient. Intuitively, this ensures that consistently misclassified points eventually get weighted high enough that they are very likely to be chosen as examples to be communicated in future rounds. The deterministic variant simply replaces Line 7 of Algorithm 7 with the weighted variant (Matousek, 1991) of the deterministic construction of $R_B$ (Chazelle, 2000); see details below.

Note that this is roughly similar in spirit to the heuristic protocol (Daumé III et al., 2012) that exchanged support points and was called ITERATIVESUPPORTS, which we will experimentally compare against. However, the protocol proposed here is less rigid, and as we will demonstrate next, this allows for a much less nuanced analysis.

Our analysis is based on the multiplicative weight update framework (and closely resembles boosting). First, we state a key structural lemma. Thereafter, we use this lemma to prove our main result.

A random sample $S_\varepsilon$ of size $s_{\varepsilon,d} = O(\min\{(d/\varepsilon)\log(d/\varepsilon), d/\varepsilon^2\})$ drawn over the entire dataset $D \subset \mathbb{R}^d$ is sufficient to learn a linear classifier with $\varepsilon$-classification error on all of $D$ with constant probability. This is based on sampling bounds mentioned earlier (see 6.1). There exist deterministic constructions for these samples $S_\varepsilon$ still of size $s_{\varepsilon,\nu}$ (Chazelle, 2000); although they provide at most $\varepsilon$-classification error with probability 1, they, in general, run in time exponential in $\nu$. Note that the VC-dimension of linear classifiers in $\mathbb{R}^d$ is $O(d)$, and these results still hold when the points are weighted and the sample is drawn (respectively constructed (Matousek, 1991)) and error measured with respect to this weighting distribution. Thus $B$ could send $s_{\varepsilon,d}$ points to $A$, and we would be done; but this is too expensive. We restate this result with a constant $c$, so that at most a $c$ fraction of the weights of points are misclassified (later we show that $c = 0.2$ is sufficient with our framework). Specifically, setting $\varepsilon = c$ and rephrasing the above results yields the following lemma.

**Lemma 6.2.6** *Let $B$ have a weighted set of points $D_B$ with weight function $w : D_B \to \mathbb{R}^+$.*

*For any constant $c > 0$, party $B$ can send a set $S_{c,d}$ of size $O(d)$ (where the constant depends on $c$) such that any linear classifier that correctly classifies all points in $S_{c,d}$ will misclassify points in $D_B$ with a total weight at most $c\sum_{x\in D_B} w(x)$. The set $S_{c,d}$ can be constructed deterministically, or a weighted random sample from $(D_B, w)$ succeeds with constant probability.*

We first state the bound using the deterministic construction of the set $S_{c,d}$, and then extend it to the more practical (from a runtime perspective) random sampling result, but with a slightly worse communication bound.

**Theorem 6.2.7** *The deterministic version of two-party two-way protocol* WEIGHTEDSAMPLING *for linear separators in $\mathbb{R}^d$ misclassifies at most, $\varepsilon|D|$ points after $T = O(\log(1/\varepsilon))$ rounds using $O(d^2 \log(1/\varepsilon))$ words of communication.*

**Proof.** At the start of each round $t$, let $\phi_t$ be the potential function given by the sum of weights of all points in that round. Initially, $\phi_1 = \sum_{x_i \in D_B} w_i = n$ since by definition, for each point $x_i \in D_B$ we have $w_i = 1$.

Then, in each round, $A$ constructs a classifier $h_A^t$ at $B$ to correctly classify the set of points that accounts for at least $1-c$ fraction of the total weight by Lemma 6.2.6. All other misclassified points are upweighted by $(1+\rho)$. Hence, for round $(t+1)$ we have $\phi^{t+1} \le \phi^t\left((1-c)+c(1+\rho)\right) = \phi^t\left(1+c\rho\right) = n\left(1+c\rho\right)^t$.

Let us consider the weight of the points in the set $S \subset D_B$ that have been misclassified by a majority of the $T$ classifiers (after the protocol ends). This implies every point in $S$ has been misclassified *at least* $T/2$ number of times and *at most*, $T$ number of times. So the minimum weight of points in $S$ is $(1+\rho)^{T/2}$ and the maximum weight is $(1+\rho)^T$.

Let $n_i$ be the number of points in $S$ that has weight $(1+\rho)^i$, where $i \in [T/2, T]$. The potential function value of $S$ after $T$ rounds is $\phi_S^T = \sum_{i=T/2}^{T} n_i (1+\rho)^i$. Our claim is that $\sum_{i=1}^{T} n_i = |S| \le \varepsilon n$. Each of these at most $|S|$ points have a weight of at least $(1+\rho)^{T/2}$. Hence we have

$$\phi_S^T = \sum_{i=T/2}^{T} n_i (1+\rho)^i \ge (1+\rho)^{T/2} \sum_{i=T/2}^{T} n_i = (1+\rho)^{T/2}|S|.$$

Relating these two inequalities we obtain the following,

$$|S|(1+\rho)^{T/2} \leq \phi_S^T \leq \phi^T = n(1+c\rho)^T .$$

Hence, (using $T = 5\log_2(1/\varepsilon)$)

$$|S| \leq n\left(\frac{(1+c\rho)}{(1+\rho)^{1/2}}\right)^T = n\left(\frac{(1+c\rho)}{(1+\rho)^{1/2}}\right)^{5\log_2(1/\varepsilon)} = n(1/\varepsilon)^{5\log_2\left(\frac{(1+c\rho)}{(1+\rho)^{1/2}}\right)} .$$

Setting $c = 0.2$ and $\rho = 0.75$ we get $5\log_2\left((1+c\rho)/(1+\rho)^{1/2}\right)) < -1$ and thus, $|S| < n(1/\varepsilon)^{-1} < \varepsilon n$, as desired since $\varepsilon < 1$. Thus each round uses $O(d)$ points, each requiring $d$ words of communication, yielding a total communication of $O(d^2 \log(1/\varepsilon))$.■

In order to use random sampling (as suggested in Algorithm 7), we need to address the probability of failure of our protocol. That is, more specifically the set $S_{c,d}$ in Lemma 6.2.6 is of size $O(d\log(1/\delta'))$ and a linear classifier that has no error on $S_{c,d}$ misclassifies points in $D_B$ with weight at most $c\sum_{x\in D_B} w(x)$, with probability at least $1-\delta'$.

However, we would like this probability of failure to be a constant $\delta$ over the entire course of the protocol. To guarantee this, we need the $c$-misclassification property to hold in each of $T$ rounds. Setting $\delta' = \delta/T$, and applying the union bound implies that then, the probability of failure at any point in the protocol is at most $\sum_{i=1}^T \delta' = \sum_{i=1}^T \delta/T = \delta$. This increases the communication cost of each round to $O(d^2\log(1/\delta')) = O(d^2\log(\log(1/\varepsilon)/\delta)) = O(d^2\log\log(1/\varepsilon))$ words, with a constant $\delta$ probability of failure. Hence, using random sampling as described in WEIGHTEDSAMPLING requires a total of $O(d^2\log(1/\varepsilon)\log\log(1/\varepsilon))$ words of communication. We formalize below.

**Theorem 6.2.8** *The randomized two-party two-way protocol* WEIGHTEDSAMPLING *for linear separators in* $\mathbb{R}^d$ *misclassifies at most* $\varepsilon|D|$ *points, with constant probability, after* $T = O(\log(1/\varepsilon))$ *rounds using* $O(d^2\log(1/\varepsilon)\log\log(1/\varepsilon))$ *words of communication.*

## 6.3 Multiparty Protocols

In the noiseless setting, extending from a two-party protocol to a $k$-party (where data is distributed to $k$ disjoint nodes) can be achieved by allowing an additional factor $k$ or $k^2$ communication, depending on the hypothesis class.

### 6.3.1 One-way communication

For $k$-players one-way protocols predetermine an ordering among players $P_1 < P_2 < \ldots < P_k$, and all communication goes from $P_i$ to $P_{i+1}$ for $i \in [1, k-1]$. In this section, we show that for $k$-players, $\varepsilon$-error classifiers can be achieved even with this restricted communication pattern. All discussed protocols can also be transformed into hierarchical one-way protocols that may have certain advantages in latency, or where all nodes just send information one-way to a predetermined coordinator node.

**6.3.1.1 Sampling results for $k$-players.** In sampling-based protocols, along the chain of players, player $P_i$ maintains a random sample $R_i$ of size $O((\nu/\varepsilon)\log(\nu/\varepsilon))$ from $\bigcup_{j=1}^{i} D_i$ and the total size $m_i = \sum_{j=1}^{i} |D_i|$. This can be easily achieved with reservoir sampling (Vitter, 1985). The final player $P_k$ computes and returns a 0-error classifier on $R_{k-1} \cup D_k$.

**Theorem 6.3.1** *Consider any family of hypothesis $(\mathbb{R}^d, \mathcal{A})$ that has VC-dimension $\nu$. Then there exists a one-way k-player protocol using $O(k(\nu/\varepsilon)\log(\nu/\varepsilon))$ total words of communication that achieves $\varepsilon$-error, with constant probability.*

**Proof.** The final set $R_{k-1}$ is an $\varepsilon$-net, so any 0-error classifier on $R_{k-1}$, is an $\varepsilon$-error classifier on $\bigcup_{j=1}^{k-1} D_i$. So, since the total number of points misclassified is at most $\sum_{j=1}^{k-1} \varepsilon |D_j| \leq \varepsilon |D|$, this achieves the proper error bound. The communication cost follows by definition of the protocol. ■

**6.3.1.2 0-Error protocols for $k$-players.** Any 0-error one-way protocol extends directly from 2-player to $k$-players. This requires that each player can send exactly the subset of the family of classifiers that permit 0 error to the next player in the sequence. This chain of players only refines this subset, so by our noiseless assumption that there exists some 0-error classifier, the final player can produce a classifier that has 0-error on all data.

**Theorem 6.3.2** *In the noiseless setting, any one-way two-player* 0*-error protocol of communication complexity C extended to a one-way k-player* 0*-error protocol with $O(Ck)$ words of communication.*

This implies that $k$-players can execute a one-way 0-error protocol for axis-aligned rectangles with $O(dk)$ communication. Classifiers from the families of thresholds and intervals follow as special case.

### 6.3.2 Two-way communication

When not restricted to one-way protocols, we assume all players take turns talking to each other in some preconceived or centrally organized fashion. This fits within standard techniques of organizing communication among many nodes that prevents transmission interference.

**6.3.2.1 Improved random sampling for $k$-players.** Our first contribution is an improved two-way $k$-player sampling-based protocol using *two-way* communication and the sampling result in (6.1). We designate party $P_1$ as a coordinator. $P_1$ gathers the size of each player's dataset $D_i$, simulates sampling from each player completely at random, and then reports back to each player the number of samples to be drawn by it, in $O(k)$ communication. Then, each other party $P_i$ selects $s_{\varepsilon,\nu}|D_i|/|D|$ random points (in expectation), and sends them to the coordinator. The union of this set satisfies the conditions of the result from (6.1) over $D = \cup_i D_i$ and yields the following result.

**Theorem 6.3.3** *For any hypothesis family with VC-dimension $\nu$ for points in $\mathbb{R}^d$, there exists a two-way k-player protocol using $O(kd + d\min\{(\nu/\varepsilon)\log(\nu/\varepsilon), \nu/\varepsilon^2\})$ total words of communication that achieves $\varepsilon$-classification error, with constant probability.*

Using two-way communication, this type of result can be made even more general. Consider the case where each $P_i$'s dataset arrives in a continuous stream; this is known as a *distributed data stream* (Cormode et al., 2008). Then applying the results of Cormode et al. (2010), we can continually maintain a sufficient random sample at the coordinator of size $s_\varepsilon$ (using a generalization of reservoir sampling) communicating $O((k + s_{\varepsilon,\nu})d\log|D|)$ words.

**Theorem 6.3.4** *Let each of k parties have a stream of data points $D_i$, where $D = \cup_i D_i$. For any hypothesis family with VC-dimension $\nu$ for points in $\mathbb{R}^d$, there exists a two-way k-player protocol using $O((k + \min\{(\nu/\varepsilon)\log(\nu/\varepsilon), \nu/\varepsilon^2\})\, d\log|D|)$ total words of communication that maintains $\varepsilon$-classification error, with constant probability.*

**6.3.2.2 An $O(k \log 1/\varepsilon)$ median based algorithm for linear separators in $\mathbb{R}^2$.** Next we consider linear separators in $\mathbb{R}^2$. We proceed in a series of epochs. In each epoch, each player takes one turn as coordinator. On its turn as coordinator, player $P_i$ plays one round of the 2-player protocol with each other player. That is, it sends out its proposed support points, and each other player responds with either early termination or an alternative set of support points, including at least one that "violates" the family of linear separators proposed by the coordinator. The protocol terminates if all noncoordinators agree to terminate early and their proposed family of linear separators all intersect. Note that even if all other players may want to terminate early, they might not agree on a single linear separator along the proposed direction; but by replying with a modified set of support points, they will designate a range, and the manner in which these ranges fail to intersect will indicate to the coordinator a "direction" to turn.

**Theorem 6.3.5** *In the noiseless setting, k-parties can find an $\varepsilon$-error classifier over halfspaces in $\mathbb{R}^2$ in $O(k^2 \log(1/\varepsilon))$ communication.*

**Proof.** Each epoch requires $O(k^2)$ communication; each of $k$ players uses a turn to communicate a constant number of bits with each of $k$ other players. We now just need to argue that the algorithm must terminate in, at most, $O(\log(1/\varepsilon))$ epochs.

We do so by showing that each player decreases its region of uncertainty by at least half for each turn it spends as coordinator, or it succeeds in finding a global separating half space and terminates. If any noncoordinator does not terminate early, it rules out at least half of the coordinator's points in the region of uncertainty since by Lemma 6.2.5, the coordinator's broadcasted support points represent the median of its uncertain points. If all noncoordinators agree on the proposed direction, and return a range of offsets that intersect, then the coordinator terminates the algorithm and can declare victory, since the sum of all error must be at most $\sum_i \varepsilon|D_i| \leq \varepsilon|D|$ in that range.

The difficult part is when all noncoordinators individually want to terminate early, but the range of acceptable offsets along the proposed normal direction of the linear separator do not globally intersect. This corresponds to the right-most picture in Figure 6.2 where the direction is forced clockwise or counter-clockwise because a negative point from one noncoordinator is "above" the positive point from a separate noncoordinator. The combination of these points thus allow the coordinator to prune half of its region of uncertainty just as if a single noncoordinator did not terminate early. ■

**6.3.2.3 An $O(kd \log 1/\varepsilon)$ boosting based algorithm for linear separators in $\mathbb{R}^d$.** In section 6.3.2.1 we described a simple protocol (Theorem 6.3.3) to learn a classifier with $\varepsilon$-error jointly among $k$ parties using $O(kd + d\min\{\nu/\varepsilon \log(\nu/\varepsilon), \nu/\varepsilon^2\})$ words of total communication. We now combine this with the two-party protocol from section 6.2.3 to obtain a $k$-player protocol for learning a joint classifier with error $\varepsilon$.

We fix an arbitrary node (say $P_1$) as the coordinator for the $k$-player protocol of Theorem 6.3.3. Then $P_1$ runs a version of the two-player protocol (from section 6.2.3) from $A$'s perspective and where players $P_2, \ldots, P_k$ serve jointly as the second player $B$. To do so, we follow the distributed sampling approach outlined in Theorem 6.3.3. Specifically, we fix a parameter $c$ (set $c = 0.2$). Each other node reports the total weight $w(D_i)$ of their data to $P_1$, who then reports back to each node what fraction of the total data $w(D_i)/w(D)$ they own. Then each player sends the coordinator a random sample of size $s_{c,d}w(D_i)/w(D)$. Recall that we require $s_{c,d} = O(d \log\log(1/\varepsilon))$ in this case to account for probability of failure over all rounds. The union of these sets at $P_1$ satisfies the sampling condition in Lemma 6.2.6 for $\cup_{i=2}^{k} D_i$. $P_1$ computes a classifier on the union of its data and this joint sample and all previous joint samples, and sends the resulting classifier back to all the nodes. Sending this classifier to each party requires $O(kd)$ words of communication. The process repeats for $T = \log_2(1/\varepsilon)$ rounds.

**Theorem 6.3.6** *The randomized k-party protocol for $\varepsilon$-error linear separators in $\mathbb{R}^d$ terminates in $T = O(\log(1/\varepsilon))$ rounds using $O((kd + d^2 \log\log(1/\varepsilon))\log(1/\varepsilon))$ words of communication, and has a constant probability of failure.*

The random sampling algorithm required a sample of size $O(d \log\log(1/\varepsilon))$. However, we can achieve a different communication trade-off using the deterministic construction where, in each round, each party $P_i$ communicates a deterministically constructed set $S_{c,i}$ of size $O(d)$. The coordinator $P_1$ computes a classifier that correctly classifies points from all of these sets having at most $cw(D_i)$ weight of points misclassified in each $D_i$. The error is at most $cw(D_i)$ on each dataset $D_i$ and so the error on all sets is at most $c\sum_{i=2}^{k} w(D_i) = cw(D)$. Again using $T = O(\log(1/\varepsilon))$ rounds, we can achieve the following result.

**Theorem 6.3.7** *The deterministic k-party protocol for $\varepsilon$-error linear separators in $\mathbb{R}^d$ terminates in $T = O(\log(1/\varepsilon))$ rounds using $O(kd^2 \log(1/\varepsilon))$ words of communication.*

## 6.4 Experiments

In this section, we present results to empirically demonstrate the correctness and convergence of ITERATIVESUPPORTS and WEIGHTEDSAMPLING. For ITERATIVESUPPORTS, we first show results using the subroutines MEDIAN and SUPPORT. However, as noted earlier, MEDIAN does not apply in higher dimensional settings. So when presenting results for WEIGHTEDSAMPLING, we compare MWU with SUPPORT only.

Each example point incurs a cost of $d+1$ ($d$ words to describe its position in $\mathbb{R}^d$ and 1 word to describe its sign). Similarly, each linear classifier requires $d+1$ words of communication ($d$ words to describe its direction and 1 word to describe its offset).

### 6.4.1 Results for median-based protocol ITERATIVESUPPORTS

In this case, we present on synthetic datasets only.

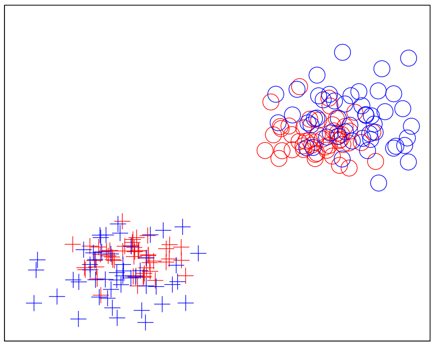
For the two-party results, we empirically compare the following methods:

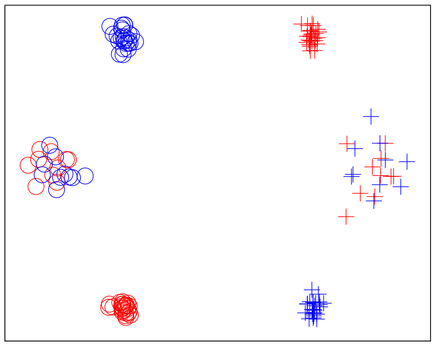
- NAIVE: a naive approach that sends all points in $A$ to $B$ and then learns at $B$,
- VOTING: a simple voting strategy that uses the majority voting rule to combine the predictions of $h_A$ and $h_B$ on $D = D_A \cup D_B$; ties are broken by choosing the label whose prediction has higher confidence,
- RANDEMP: $A$ sends a random sample (an $\varepsilon$-net $S_A$ of size $(d/\varepsilon)\log(d/\varepsilon)$) of $D_A$ to $B$ and $B$ learns on $D_B \cup S_A$,
- MAXMARG: ITERATIVESUPPORTS that selects informative points heuristically (ref. to section 6.2.2), and
- MEDIAN: ITERATIVESUPPORTS that selects informative points with convergence guarantees (ref. to section 6.2.2).

SVM (based on libSVM (Chang and Lin, 2011)) was used as the underlying classifier for all aforementioned approaches. In all cases, the errors are reported on the dataset $D$ with an $\varepsilon$ value of 0.05 (where applicable).

The above methods have been evaluated on three synthetically generated datasets (*Data1*, *Data2*, *Data3*). For all datasets, both $A$ and $B$ contain 500 data points each (250 positive and 250 negative). Figure 6.5 pictorially depicts the data.

Table 6.2 compares the accuracies and communication costs of the aforementioned methods for the dataset in 2-dimensions. For all datasets, MAXMARG and MEDIAN required the least amount of communication to learn an optimal classifier. For cases when it is easy to separate the positive from the negative samples (e.g., *Data1* and *Data2*), MAX-MARG converges faster than MEDIAN. However, *Data3* show that there exists difficult datasets where MEDIAN requires less communication than MAXMARG. This reinforces


(a) *Data1*


(b) *Data2*

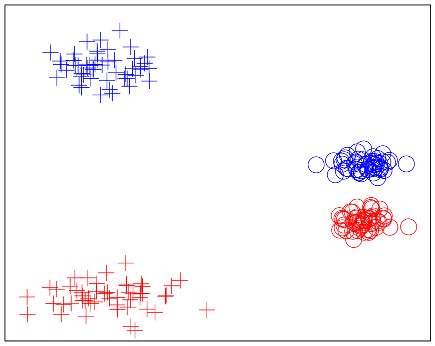

(c) *Data3*

**Figure 6.5**. Red represents $A$ and blue represents $B$. Positive and negative examples (for all datasets) are denoted by '$+$'s and '$\circ$'s, respectively.

our theoretical convergence claims for MEDIAN that hold for *any* input dataset. *Data3* in Table 6.2 shows that there exists cases when both VOTING and RANDEMP perform worse than MEDIAN and with a much higher communication overhead; for *Data3*, VOTING performs as bad as random guessing. Finally, neither VOTING nor MAXMARG provide any provable error guarantees.

Table 6.3 presents results for *Data1*, *Data2*, *Data3* extended to dimension $= 10$. As can be seen, our proposed heuristic MAXMARG outperforms all other baselines in terms communication cost while having comparable accuracies.

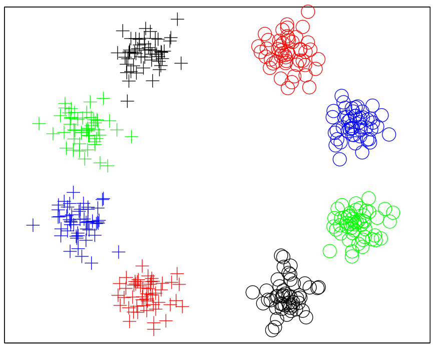
The aforementioned methods have been appropriately modified for the multiparty scenario. For NAIVE, VOTING and RANDEMP, a node is fixed as the *coordinator* and the remaining $(k-1)$ nodes send their information to the coordinator node which aggregates all the received information. For MAXMARG and MEDIAN, in each epoch, one of the $k$-players takes a turn to act as the *coordinator* and updates its state by receiving information from each of the remaining $(k-1)$ nodes. We experiment with a $k$ value of 4 (i.e., four nodes $A,B,C,D$). As earlier, for all datasets, each of $A$,$B$,$C$,$D$ contain 500 examples (250 positive and 250 negative). The datasets are shown in Figure 6.6.

**Table 6.2**. Accuracy (Acc) and communication cost (Cost) of different methods for *two-dimensional* noiseless datasets.

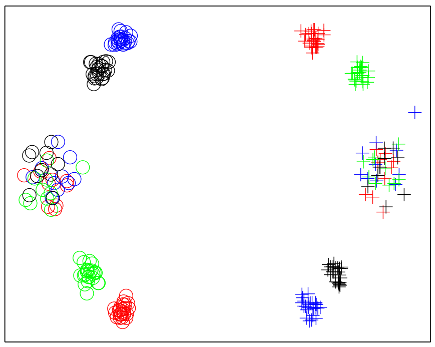
| **Method** | ***Data1*** | | ***Data2*** | | ***Data3*** | |
|---|---|---|---|---|---|---|
| | **Acc** | **Cost** | **Acc** | **Cost** | **Acc** | **Cost** |
| NAIVE | 100% | 500 | 100% | 500 | 100% | 500 |
| VOTING | 100% | 500 | 100% | 500 | 50% | 500 |
| RANDEMP | 100% | 65 | 100% | 65 | 99.62% | 65 |
| **MAXMARG** | **100%** | **4** | **100%** | **4** | **100%** | **12** |
| **MEDIAN** | **100%** | **6** | **100%** | **6** | **100%** | **10** |

**Table 6.3**. Accuracy (Acc) and communication cost (Cost) of different methods for *high-dimensional* noiseless datasets.

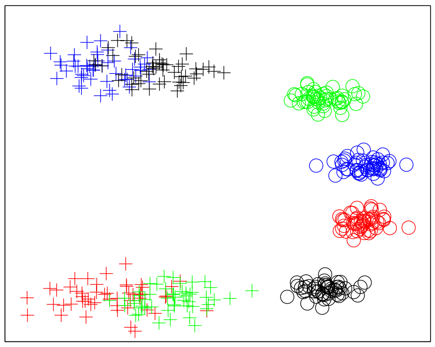
| **Method** | ***Data1*** | | ***Data2*** | | ***Data3*** | |
|---|---|---|---|---|---|---|
| | **Acc** | **Cost** | **Acc** | **Cost** | **Acc** | **Cost** |
| NAIVE | 100% | 500 | 100% | 500 | 100% | 500 |
| VOTING | 100% | 500 | 100% | 500 | 81.8% | 500 |
| RANDEMP | 100% | 100 | 100% | 100 | 99.1% | 100 |
| **MAXMARG** | **100%** | **4** | **100%** | **4** | **98.27%** | **40** |



(a) *Data1*



(b) *Data2*



(c) *Data3*

**Figure 6.6**. Red represents $A$, blue represents $B$, green represents $C$ and black represents $D$. Positive and negative examples (for all datasets) are denoted by '$+$'s and '$\circ$'s, respectively.

As shown in Table 6.4, for the $k$-party case, ITERATIVESUPPORTS substantially outperforms the baselines on all datasets. As earlier, for the difficult dataset *Data3*, MEDIAN incurs less communication cost as compared to MAXMARG. We observed that for *Data1* and *Data2*, both MAXMARG and MEDIAN require the same number of iterations to converge. However, the cost for MEDIAN is higher due to its quadratic dependency on $k$.

### 6.4.2 Results for boosting-based protocol WEIGHTEDSAMPLING

In this section we compare WEIGHTEDSAMPLING with the following baselines for 2-party and $k$-party protocols.

- NAIVE: sends all data from $(k-1)$ nodes to a coordinator node and then learns at the coordinator.
- VOTING: trains classifiers at each individual node and sends over the $(k-1)$ classifiers to a coordinator node. For any datapoint, the coordinator node predicts the label by taking a vote over all $k$ classifiers.
- RAND: each of the $(k-1)$ nodes sends a random sample of size $s_{\varepsilon,d}$ to a coordinator node and then a classifier is learned at the coordinator node using all of its own data and the samples received.
- RANDEMP: cheaper version of RAND that uses a random sample of size $9d$ from each party each round; this value was chosen to make this baseline technique as favorable as possible.
- MAXMARG: ITERATIVESUPPORTS that selects informative points heuristically (Daumé III et al., 2012). We do not compare with MEDIAN (Daumé III et al., 2012) as it is not applicable beyond two dimensions.
- MWU: WEIGHTEDSAMPLING that randomly samples points based on the distribution of the weights and runs for $5\log(1/\varepsilon)$ number of rounds (ref. section 6.2.3).
- MWUEMP: a cheaper version of MWU which is terminated early if the training error has reached $\varepsilon|D|$.

For all these methods, SVM (from libSVM (Chang and Lin, 2011) library), with a linear kernel, was used as the underlying classifier. We report training accuracy and communication cost. The training accuracy is computed over the combined dataset $D$ with an $\varepsilon$ value of 0.05 (where applicable). The communication cost (in words) of all methods are reported as

**Table 6.4**. Accuracy (Acc) and communication cost (Cost) of different methods for *two-dimensional* noiseless datasets.

| **Method** | ***Data1*** | | ***Data2*** | | ***Data3*** | |
|---|---|---|---|---|---|---|
| | **Acc** | **Cost** | **Acc** | **Cost** | **Acc** | **Cost** |
| NAIVE | 100% | 1500 | 100% | 1500 | 100% | 1500 |
| VOTING | 98.75% | 1500 | 100% | 1500 | 50% | 1500 |
| RANDEMP | 100% | 195 | 100% | 195 | 99.76% | 195 |
| **MAXMARG** | **97.61%** | **14** | **100%** | **2** | **97.38%** | **38** |
| **MEDIAN** | **99.0%** | **36** | **100%** | **6** | **98.75%** | **29** |

ratios with reference to MWUEMP as the base method. All numbers reported are averaged over 10 runs of the experiments; standard deviations are reported where appropriate. For MWU and MWUEMP, we use $\rho = 0.75$.

Note that given our cost computation, for some datasets the cost of RAND, RANDEMP and MWU can exceed the cost of NAIVE (see, for example, *Cancer*).

Six datasets, three each for two-party and four-party case, have been generated synthetically from a mixture of Gaussians. Each Gaussian has been carefully seeded to generate different data partitions. For *Synthetic1*, *Synthetic2*, *Synthetic4*, *Synthetic5*, each node contains 5000 data points (2500 positive and 2500 negative), whereas for *Synthetic3* and *Synthetic6*, each node contains 8500 data points (4250 positive and 4250 negative) and all of these datapoints lie in 50 dimensions. Additionally, we investigate the performance of our protocols on real-world datasets. We use *Cancer* and *Mushroom* from the LibSVM data repository (Chang and Lin, 2011) as these datasets are linearly or almost linearly separable. This shows that although our protocols were designed for noiseless data they work well on noisy datasets, too. However, when applied on noisy data, we do not guarantee the accuracy bounds that were claimed for noiseless datasets.

In Tables 6.5-6.6, we highlight (in bold) the protocol that performs the best. By best we mean that the method has the cheapest communication cost as well an accuracy that is more than $(1-\varepsilon)$ times the optimal, i.e., 95% for $\varepsilon = 0.05$. As will be frequently seen for VOTING, the communication cost is the cheapest but the accuracy is far from the desired $\varepsilon$-error specified, and in such circumstances we do not deem VOTING as the best method.

Table 6.5 compares the performance metrics of the aforementioned protocols for *two*-parties. As can be seen, VOTING performs the best for *Synthetic1* and RANDEMP performs the best for *Synthetic2*. For *Synthetic3*, MWUEMP requires the least amount of communication to learn an $\varepsilon$-optimal distributed classifier. Note that, for *Synthetic2* and *Synthetic3*, both VOTING and MAXMARG fail to produce an $\varepsilon$-optimal ($\varepsilon = 0.05$) classifier. MAXMARG exhibits this behavior despite incurring a communication cost that is as high as NAIVE (i.e., the accumulated cost of the support points become the same as the cost of NAIVE, at which point we stop the algorithm).

In Table 6.5, most of the two-party results carry over to the multiparty case. VOTING is the best for *Synthetic4*, whereas MWUEMP is the best for *Synthetic5* and *Synthetic6*. As

**Table 6.5**. Mean accuracy (Acc) and communication cost (Cost) required by *two-party* and *four-party* protocols for synthetic datasets.

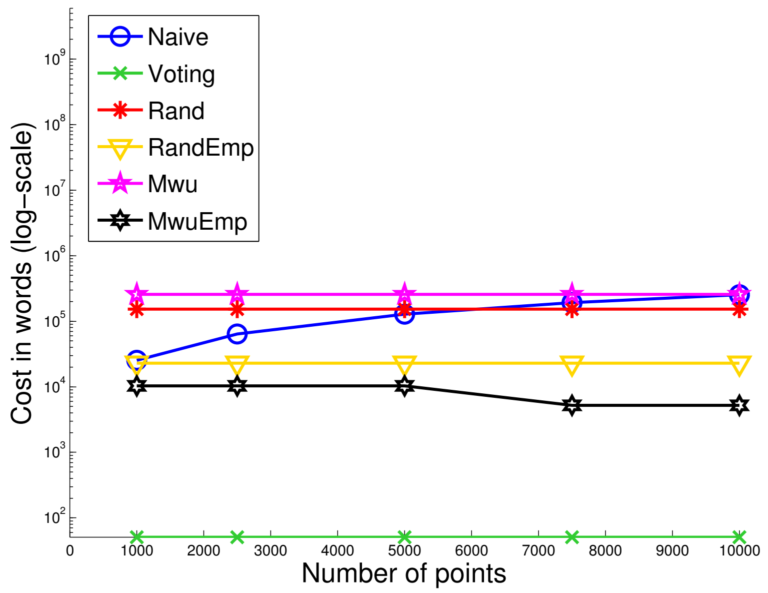
| | ***Synthetic1*** | | ***Synthetic2*** | | ***Synthetic3*** | |
|---|---|---|---|---|---|---|
| | **Acc** | **Cost** | **Acc** | **Cost** | **Acc** | **Cost** |
| | two-party | | | | | |
| NAIVE | 99.23 (0.0) | 49.0 | 97.91 (0.0) | 6.18 | 97.39 (0.0) | 19.1 |
| VOTING | **95.00 (0.0)** | **0.01** | 60.64 (0.0) | 0.01 | 74.55 (0.0) | 0.01 |
| RAND | 99.02 (0.0) | 29.4 | 97.72 (0.0) | 3.71 | 97.16 (0.0) | 6.74 |
| RANDEMP | 96.64 (0.1) | 4.41 | **95.13 (0.1)** | **0.56** | 96.03 (0.1) | 1.01 |
| MAXMARG | 96.39 (0.0) | 4.26 | 93.76 (0.0) | 6.18 | 73.62 (0.0) | 19.1 |
| MWU | 98.66 (0.1) | 49.5 | 97.59 (0.1) | 6.24 | 97.11 (0.1) | 11.3 |
| MWUEMP | 95.00 (0.0) | 1.00 | 95.17 (0.1) | 1.00 | **95.25 (0.2)** | **1.00** |
| | four-party | | | | | |
| NAIVE | 99.26 (0.0) | 100 | 97.97 (0.0) | 12.7 | 97.47 (0.0) | 54.8 |
| VOTING | **95.00 (0.0)** | **0.01** | 65.83 (0.0) | 0.01 | 75.52 (0.0) | 0.01 |
| RAND | 99.18 (0.0) | 60.0 | 97.83 (0.0) | 7.63 | 97.39 (0.0) | 19.4 |
| RANDEMP | 97.33 (0.1) | 9.00 | 96.61 (0.1) | 1.15 | 96.67 (0.1) | 2.90 |
| MAXMARG | 95.95 (0.0) | 0.82 | 93.94 (0.0) | 15.2 | 75.05 (0.0) | 80.2 |
| MWU | 98.03 (0.2) | 34.8 | 97.30 (0.1) | 4.45 | 96.87 (0.1) | 11.2 |
| MWUEMP | 95.11 (0.3) | 1.00 | **95.11 (0.2)** | **1.00** | **95.45 (0.2)** | **1.00** |

earlier, both VOTING and MAXMARG do not yield 0.05-optimal classifiers for *Synthetic5* and *Synthetic6*.

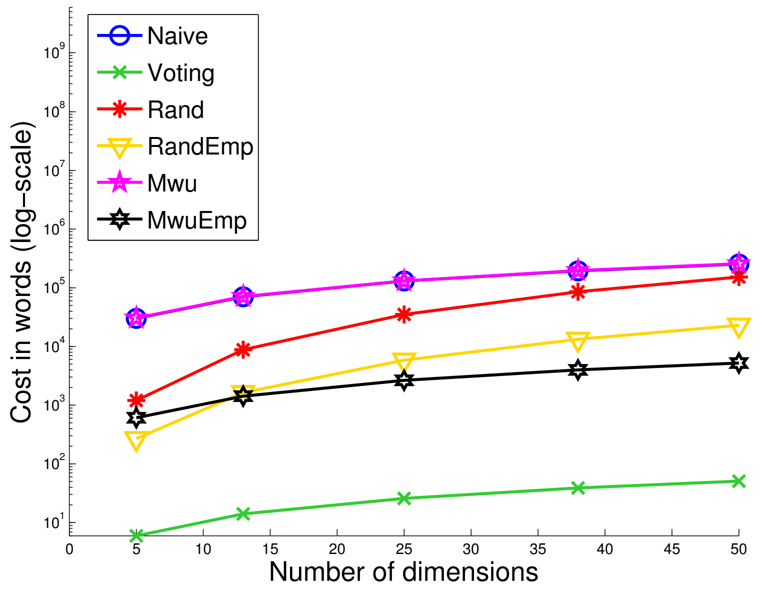
Figure 6.7 (for two-party using *Synthetic1*) shows the communication costs (in *log-scale*) with variations in the number of data points per node and the dimension of the data. Note that we do not report the numbers for MAXMARG since MAXMARG takes a long time to finish. However, for *Synthetic1* the numbers for MAXMARG are similar to those of RANDEMP and so their traces are similar. Note that in Figure 6.7, the cost of NAIVE increases as the number of dimensions increase. This is because the cost is multiplied by a factor of $(d+1)$, when expressed in words.

Table 6.6 presents results for two- and four-party protocols using real-world datasets. Other than the two-party case for *Mushroom*, VOTING performs best in all other cases. However, note that VOTING does not yield a 0.05-optimal distributed classifier for *Mushroom* using two-party protocol.

The results for communication cost (in *log-scale*) versus data size and communication cost (in *log-scale*) versus dimensionality are provided in Figure 6.8 for two-party protocol using the *Mushroom* dataset. MWUEMP (denoted by the black line) is comparable to

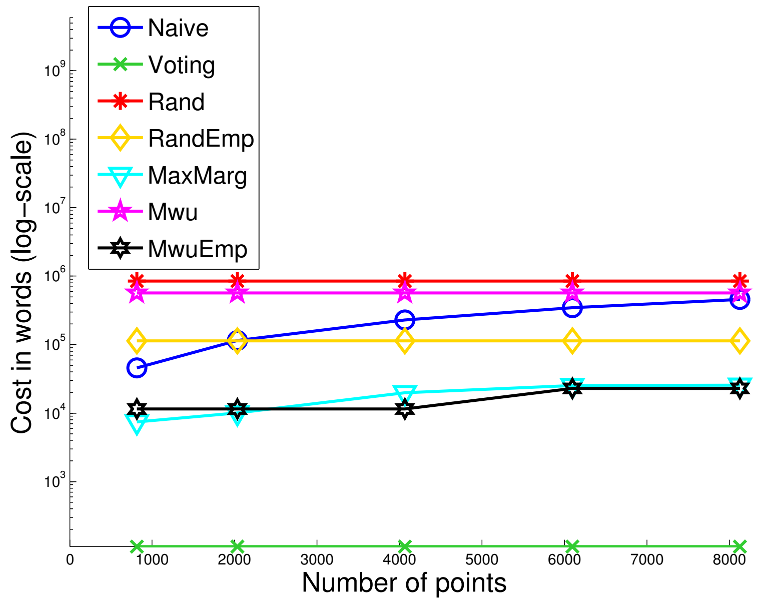


(a) Communication cost vs size

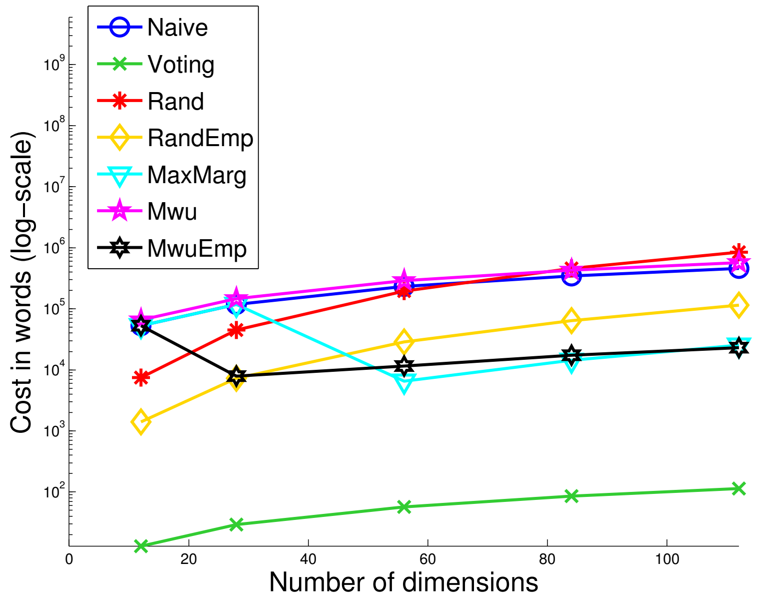


(b) Communication cost vs dimension

**Figure 6.7**. Communication cost vs size and dimensionality for *Synthetic1* with two-party protocol.



(a) Communication cost vs size



(b) Communication cost vs dimension

**Figure 6.8**. Communication cost vs size and dimensionality for *Mushroom* with two-party protocol.

**Table 6.6**. Results for all protocols using *Cancer* ($|D| = 683$, $d = 10$) and *Mushroom* ($|D| = 8124$, $d = 112$). The standard deviation of the accuracies over multiple runs are insignificant and hence have been ignored in this table.

| | ***Cancer*** | | ***Mushroom*** | | ***Cancer*** | | ***Mushroom*** | |
|---|---|---|---|---|---|---|---|---|
| | **Acc** | **Cost** | **Acc** | **Cost** | **Acc** | **Cost** | **Acc** | **Cost** |
| | two-party | | | | four-party | | | |
| NAIVE | 97.07 | 3.34 | 100.00 | 20.01 | 97.07 | 1.00 | 100.00 | 28.61 |
| VOTING | **97.36** | **0.01** | 88.38 | 0.00 | **97.36** | **0.03** | **95.67** | **0.01** |
| RAND | 97.16 | 4.52 | 100.00 | 36.97 | 97.19 | 12.81 | 100.00 | 105.70 |
| RANDEMP | 96.90 | 0.88 | 100.00 | 4.97 | 96.99 | 2.50 | 99.99 | 14.20 |
| MAXMARG | 96.78 | 0.22 | 100.00 | 1.11 | 96.78 | 0.56 | 100.00 | 2.34 |
| MWU | 97.36 | 49.51 | 100.00 | 24.88 | 97.00 | 48.46 | 100.00 | 24.65 |
| MWUEMP | 96.87 | 1.00 | **99.73** | **1.00** | 96.97 | 1.00 | 98.86 | 1.00 |

MAXMARG and cheaper than all other baselines (except VOTING).

The goal of our experiments was to show that our protocols perform well, particularly on difficult or adversarially partitioned datasets. For easy datasets, any baseline technique can perform well. Indeed, VOTING performs the best on *Synthetic1* and *Synthetic4* and RANDEMP performs better than others on *Synthetic2*. For the remaining three cases on synthetic datasets, MWUEMP outperforms the other baselines. On real world data, VOTING usually performs well. However, as we have seen, for some datasets VOTING and MAXMARG fail to yield an $\varepsilon$-optimal classifier. In particular for *Mushroom*, using the two-party protocol, the accuracy achieved by VOTING is far from $\varepsilon$-optimal. These results show that there exists scenarios where VOTING and MAXMARG perform particularly worse and thus, are not safe strategies.

## 6.5 Summary

This chapter introduced the problem of learning classifiers across distributed data where the communication between datasets is the bottleneck to be optimized. This model focus on real-world communication bottlenecks is increasingly prevalent for massive distributed datasets. Several very general solutions were identified within this framework and introduced new techniques which provided provable exponential improvement by harnessing two-way communication. Additionally, this chapter also proposed a simple and efficient MWU-based protocol that learned an $\varepsilon$-optimal distributed classifier for hyperplanes in arbitrary dimensions. The protocol gracefully extended to $k$-players.

# CHAPTER 7

# CONCLUSION

Machine learning algorithms have witnessed great success over the past few decades. There has been substantial research contributions in different subareas and machine learning is being applied in a wide range of real-world applications. However, with increasing amounts of data at our disposal, machine learning algorithms face new challenges. The primary question is whether one should prefer simple algorithms trained on lots of data over more complicated models. Moreover, these big amounts of data are mostly unlabeled and labeling them is a challenging task. Thus, learning over large datasets that impose inherent constraints (such as, few labeled instances, labeling costs or lack of computational resources) poses several interesting research questions.

This thesis has focused on learning on a budget but with small or moderate amounts of data. The methods introduced in this thesis aim to reduce the overall cost by allowing the learner to transfer knowledge from related problem domains or settings. In addition, the thesis also proposes distributed strategies that are on a low communication budget. Such problem settings are a good starting point to address bigger challenges on large scale datasets. In this chapter, we summarize the contributions of this thesis, and discuss future research directions that are primarily aimed towards learning on big datasets.

## 7.1 Summary of Contributions

- **Semisupervised transfer learning.** We proposed a new semisupervised technique for domain adaptation, a subarea of transfer learning. Existing domain adaptation techniques were mostly supervised in the sense that these algorithms assume the existence of labeled data in both the source domain and the target domain. However, these supervised domain adaptation algorithms are wasteful as they fail to leverage unlabeled data, present in abundance, in both source and target domains. In this work, we presented a coregularization based approach to semisupervised domain adaptation. Our proposed approach (EASYADAPT++) built on the notion of augmented space (introduced in EASYADAPT (Daumé III, 2007)) and exploited unlabeled data in target domain to further enable the transfer of information from source to target. This semisupervised approach to domain adaptation is extremely simple to implement and can be applied as a preprocessing step to any supervised learner. Our theoretical analysis (in terms of Rademacher complexity) of EASYADAPT and EASYADAPT++ showed that the hypothesis class of EASYADAPT++ has lower complexity (compared to EASYADAPT) and hence, resulted in tighter generalization bounds. Experimental results on sentiment analysis tasks reinforced our theoretical findings and demonstrated the efficacy of the proposed semisupervised method when compared to (supervised) EASYADAPT as well as a few other baseline approaches.

- **Online transfer learning.** We proposed an Online MultiTask Learning (OMTL) framework which simultaneously learned the task weight vectors as well as the task relatedness adaptively from the data. Our contribution is in contrast with prior work on online multitask learning which assumed fixed task relatedness, a priori. Furthermore, whereas prior work in such settings assume only positively correlated tasks, our framework can capture negative correlations as well. Our proposed framework learns the task relationship matrix by framing the objective function as a Bregman divergence minimization problem for positive definite matrices. Subsequently, we exploited this adaptively learned task-relationship matrix to select the most informative samples in an online multitask active learning setting. Experimental results on a number of real-world datasets and comparisons with numerous baselines established the usefulness of our proposed framework.
- **Active transfer learning.** In this work, we harness the synergy between two important learning paradigms, namely, active learning and domain adaptation. We showed how active learning in a target domain can leverage information from a different but related source domain. Our proposed framework, Active Learning Domain Adapted (ALDA), used source domain knowledge to transfer information that facilitates active learning in the target domain. We proposed two variants of ALDA, namely, a batch B-ALDA and an online O-ALDA. Empirical comparisons with numerous baselines on real-world datasets showed the utility of transfer of information in active learning

settings.

- **Communication-efficient distributed learning.** We considered the problem of learning classifiers for labeled data that has been distributed across several nodes. Our goal was to find a single classifier, with small approximation error, across all datasets while minimizing the communication between nodes. This setting modeled real-world communication bottlenecks in the processing of massive distributed datasets. We proposed several very general sampling-based solutions as well as some two-way protocols which have a provable exponential speed-up over any one-way protocol. We focused on core problems for noiseless data distributed across two or more nodes. The techniques we introduced are reminiscent of active learning, but rather than actively probing labels, nodes actively communicate with each other - each node simultaneously learning the important data from another node. In addition, we presented a two-party multiplicative-weight-update based protocol that used $O(d^2 \log 1/\varepsilon)$ words of communication to classify distributed data in arbitrary dimension $d$, $\varepsilon$-optimally. This readily extended to classification over $k$ nodes with $O(kd^2 \log 1/\varepsilon)$ words of communication. Our multiplicative-weight-update protocols were simple to implement and were considerably more efficient than baselines compared with, as demonstrated by our empirical results.

## 7.2 Future Challenges

During the work of this thesis, we came across several interesting questions that could possibly culminate into full blown problems of their own. Of particular interests were questions that relate, and in some cases extend, our existing approaches to large-scale settings. Additionally, our proposed distributed model, despite being a good starting point for theoretical study lacks several aspects that make it infeasible in real-world scenarios. In the following, we highlight a few specific questions (related to the aforementioned directions) that we think could lead to fruitful research contributions in the future.

- **Large-scale transfer learning.** People have proposed multitask algorithms (Chapelle et al., 2010) that scale with the number of datapoints. However, consider a scenario of *personalized search* that has billions of users. Here each user is a task and each task has a few labeled points to start with. In our proposed OMTL, the task relationship matrix is quadratic in number of tasks which makes the approach practically infeasible since matrix operations (such as, matrix inverse, matrix logarithm) on such large matrices are computationally expensive. This calls for lightweight approaches to learn tasks relationships on web-scale data. Can we propose efficient multitask learning algorithms that scale with the number of tasks? Also, can this be done in an online fashion?

- **Lower bounds on two-way communication for our distributed model.** A new model for distributed learning has been proposed in Chapter 6. We show lower bounds for one-way communication and demonstrate exponential improvement with two-way communication. However, we do not know whether the bound obtained using two-way protocols is tight. Can we come up with *constant* lower-bounds on the communication cost of two-way protocols?
- **Real-life aspects of our distributed model.** In our proposed distributed model, our aim was to minimize the amount of communication (or the number of words exchanged). For example, the players in our setting communicate by exchanging data points as well as classifiers. However, exchanging data points can raise privacy issues. Consider a hospital network that aims to build a global classifier over its patient data. In this case, it is imperative that hospitals refrain from exchanging data points. Can we learn distributed models efficiently under privacy constraints? Again, consider a case where the nodes of a distributed system are physically separated by large geographical distances. It may not always be feasible for the nodes that are separated by huge distances to communicate small amounts, or probably any amount, of information. This is because the information so communicated may be attenuated while traveling for large distances over lossy channels or may get corrupted. Moreover, this information while traveling long distances gets exposed over greater time durations to attacks by malicious adversaries. So, a more realistic scenario could be that each node communicates with its local neighborhood. How can we learn distributed classifiers under this more realistic model of communication?

# APPENDIX A

# SEMISUPERVISED TRANSFER

In the following, we provide proofs for Theorem 4.2, Theorem 4.4 and Theorem 4.5. Note that the derivations and proofs make use of the kernel submatrices $A, B, C, D, E, F$ (as defined in Eq. 4.6 of the original paper).

## A.1 Proof of Theorem 3.3.2

**Proof.** Let $h_s^*$ and $h_t^*$ be the optimal source and target hypotheses in $\mathscr{H}_s$ and $\mathscr{H}_t$, respectively. Using triangle inequality for the loss function, we have

$$\varepsilon_t(h_t, f_t) \leq \varepsilon_t(h_t, h_t^*) + \varepsilon_t(h_t^*, f_t).$$

We use the notion of $d_{\mathscr{H}\Delta\mathscr{H}}$-distance in the next step, which is defined in Blitzer et al. (2007a) as

$$\sup_{h_1, h_2 \in \mathscr{H}} 2|\varepsilon_s(h_1, h_2) - \varepsilon_t(h_1, h_2)|$$

This gives us

$$\varepsilon_t(h_t, f_t) \leq \varepsilon_s(h_t, h_t^*) + \frac{1}{2} d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s, D_t) + \varepsilon_t(h_t^*, f_t).$$

We make use of triangle inequality again to get

$$\varepsilon_t(h_t, f_t) \leq \varepsilon_s(h_t, f_s) + \varepsilon_s(f_s, f_t) + \varepsilon_s(h_t^*, f_t) + \frac{1}{2} d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s, D_t) + \varepsilon_t(h_t^*, f_t).$$

We denote $\eta_s := \varepsilon_s(f_s, f_t)$, $\nu_s := \varepsilon_s(h_t^*, f_t)$, and $\nu_t := \varepsilon_t(h_t^*, f_t)$. Subtracting $\varepsilon_s(h_s, f_s)$ from both sides, we get

$$\begin{aligned}
\varepsilon_t(h_t,f_t)-\varepsilon_s(h_s,f_s) &\leq (\varepsilon_s(h_t,f_s)-\varepsilon_s(h_s,f_s))+\frac{1}{2}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s,D_t)+\eta_s+\nu_s+\nu_t \\
&\leq ME_s[h_t(x)-h_s(x)]+\frac{1}{2}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s,D_t)+\eta_s+\nu_s+\nu_t \\
&\qquad \text{(using M-Lipschitz property of loss function)} \\
&= ME_s[\langle h_t,k(x,\cdot)\rangle-\langle h_s,k(x,\cdot)\rangle]+\frac{1}{2}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s,D_t)+\eta_s+\nu_s+\nu_t \\
&\qquad \text{(using the reproducing kernel property)} \\
&= ME_s[\langle h_t-h_s,k(x,\cdot)\rangle]+\frac{1}{2}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s,D_t)+\eta_s+\nu_s+\nu_t \\
&\leq M||h_t-h_s||E_s[||k(x,\cdot)||]+\frac{1}{2}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s,D_t)+\eta_s+\nu_s+\nu_t \\
&= M||h_t-h_s||E_s[\sqrt{k(x,x)}]+\frac{1}{2}d_{\mathscr{H}_t\Delta\mathscr{H}_t}(D_s,D_t)+\eta_s+\nu_s+\nu_t.
\end{aligned}$$

(Note: Some of the steps involving reduction to the term $E_s\left[\sqrt{k(x,x)}\right]$ are similar to (Mansour et al., 2009).) ■

## A.2 Proof of Theorem 3.3.4: Complexity for EA

In this section, we bound the complexity of target hypothesis class $\mathscr{J}^t_{EA}$ for EA. The base hypothesis class $\mathscr{H}$ in Eq. 4.3 (of the original paper) is symmetric in source and target hypotheses. So the complexity of source class $\mathscr{J}^s_{EA}$ can be obtained by replacing adequate terms. We are interested in the complexity of the target hypothesis class $\mathscr{J}^t_{EA}$ which is defined as $\mathscr{J}^t_{EA} := \{h_2 : \mathscr{X} \mapsto \mathbb{R}, (h_1,h_2) \in \mathscr{H}\}$, where $h_1$ is not fixed a priori.

The Rademacher complexity of $\mathscr{J}^t_{EA}$ is defined as

$$\hat{R}_n(\mathscr{J}^t_{EA}) = E_\sigma\left[\sup_{(h_1,h_2)\in\mathscr{H}}\left|\frac{2}{l_t}\sum_{i=1}^{l_t}\sigma_i h_2(x_i)\right|\right] \tag{A.1}$$

The basic framework of proof is similar to the proof of the main theorem of (Rosenberg and Bartlett, 2007). The hypothesis class considered in their work is different than ours. They find the complexity of average hypothesis class (i.e., $x \mapsto (h_1(x)+h_2(x))/2$), while we are interested in class $\mathscr{J}^t_{EA}$, as defined above. We also note that $h_2 \in \mathscr{J}^t_{EA} \implies -h_2 \in \mathscr{J}^t_{EA}$ since $(h_1,h_2) \in \mathscr{H} \implies (-h_1,-h_2) \in \mathscr{H}$. This means that we can remove the absolute value sign from Eq. A.1. Since, $\forall i, h_2(x_i) = \langle k(x_i,\cdot),h_2\rangle$, we can restrict the supremum to $h_1$ and $h_2$ that are in the span of all samples and also in $\mathscr{H}$. The restricted condition on $(h_1,h_2)$ then becomes

$$\left\{(h_\alpha,h_\beta):\lambda_1\alpha'K\alpha+\lambda_2\beta'K\beta+\lambda(\alpha-\beta)'K(\alpha-\beta)\leq 1\right\}=\left\{(h_\alpha,h_\beta):(\alpha'\,\beta')M(\alpha'\,\beta')'\leq 1\right\}$$

where

$$M=\begin{pmatrix}(\lambda_1+\lambda)K & -\lambda K\\ -\lambda K & (\lambda_2+\lambda)K\end{pmatrix},$$

and $K$ is the kernel matrix for source labeled and target labeled samples. Using the reproducing kernel property, we get

$$\hat{R}_n(\mathscr{J}^t_{EA})=\frac{2}{l_t}E_\sigma\sup_{\alpha,\beta\in\mathbb{R}^{l_s+l_t}}\left\{\sigma'(C'B)\beta:(\alpha'\,\beta')M(\alpha'\,\beta')'\leq 1\right\}.$$

For a symmetric positive definite matrix M, it can be shown that

$$\sup_{(\alpha,\beta):(\alpha'\,\beta')M(\alpha'\,\beta')'\leq 1}x'\beta=||(M/M_{11})^{-1/2}x||=||(M^{-1})^{1/2}_{22}x||, \qquad \text{(A.2)}$$

and the maxima occurs at $\alpha=-M_{11}^{-1}M_{12}\beta$. $M/M_{11}$ is the Schur complement of block $M_{11}$ of matrix $M$ (i.e., $M/M_{11}=M_{22}-M_{21}M_{11}^{-1}M_{12}$).

The matrix $M$ may not always be full rank, however, it can be noted that if $\beta$ is in the null space of $K$, $(C'\,B)\beta$ will be zero. So, we can project $\beta$ onto the column space of $K$ (or row space due to $K$ being a symmetric matrix) to get $\beta_{pr}$ and the term $(C'\,B)\beta_{pr}$ is equal to $(C'\,B)\beta$. Specifically, $\beta_{pr}$ can be thought as computed by the operation $UU^T_{pr}\beta$, where $U$ is the full eigenvector matrix and $U_{pr}$ is the eigenvector matrix consisting of only the vectors having nonzero eigenvalues. So, the sup is restricted to the projected $\alpha_{pr}$ and $\beta_{pr}$, and the expression for Rademacher complexity can be rewritten as

$$\hat{R}_n(\mathscr{J}^t_{EA})=\frac{2}{l_t}E_\sigma\sup_{\alpha_{pr},\beta_{pr}\in ColSpace\{K\}}\left\{\sigma'(C'\,B)\beta_{pr}:(\alpha'_{pr}\beta'_{pr})M(\alpha'_{pr}\beta'_{pr})'\leq 1\right\}.$$

We proceed in a manner similar to that used in (Rosenberg and Bartlett, 2007) and diagonalize the kernel matrix $K$ to get orthonormal bases $U$ corresponding to the nonzero eigenvalues ($K=U'\Lambda U$). $\Lambda$ is a diagonal matrix of size $r\times r$, containing just the nonzero eigenvalues and $r$ is the rank of matrix $K$. Since $\alpha_{pr}$ and $\beta_{pr}$ are in the span of column space of $K$, there exist $a_s$ and $b$ such that

$$\alpha_{pr} = Ua \qquad \text{and} \qquad \beta_{pr} = Ub$$

The expression for complexity now becomes,

$$\hat{R}_n(\mathscr{J}^t_{EA}) = \frac{2}{l_t} E_\sigma \sup \left\{ \sigma' Wb : (a'\ b')P(a'\ b')' \leq 1 \right\}$$

where $W = (C'\ B)U$ and

$$P = \begin{pmatrix} (\lambda_1 + \lambda)\Lambda & -\lambda\Lambda \\ -\lambda\Lambda & (\lambda_2 + \lambda)\Lambda \end{pmatrix}$$

Using Eq. A.2, the supremum can be evaluated as

$$\hat{R}_n(\mathscr{J}^t_{EA}) = \frac{2}{l_t} E_\sigma ||(P^{-1/2})_{22} W' \sigma||.$$

We now make use of Kahane-Khintchine inequality (Latala and Oleszkiewicz, 1994) which is stated in the following lemma.

**Lemma A.2.1** *For any vectors $a_1, a_2, \ldots, a_n$ and independent Rademacher random variables $\sigma_1, \sigma_2, \ldots, \sigma_n$, we have*

$$\frac{1}{\sqrt{2}} E \left\| \sigma^n_{i=1} \sigma_i a_i \right\|^2 \leq \left( E \left\| \sigma^n_{i=1} \sigma_i a_i \right\| \right)^2 \leq E \left\| \sigma^n_{i=1} \sigma_i a_i \right\|^2$$

**Proof.** Using the above inequality we get a lower and upper bound on the complexity as

$$\frac{2C^t_{EA}}{2^{1/4} l_t} \leq \hat{R}_n(\mathscr{J}^t_{EA}) \leq \frac{2C^t_{EA}}{l_t}, \tag{A.3}$$

where

$$\begin{aligned} \left(C^t_{EA}\right)^2 &= E_\sigma ||(P^{-1})^{1/2}_{22} W' \sigma||^2 \\ &= E_\sigma \left( \sigma' W (P^{-1})_{22} W' \sigma \right) \\ &= E_\sigma tr\{\sigma \sigma' W (P^{-1})_{22} W'\} \\ &= tr\{W (P^{-1})_{22} W'\}. \end{aligned} \tag{A.4}$$

The above expression can be written in terms of the original kernel submatrices by doing algebraic manipulations on the eigenbases using similar steps as in (Rosenberg and Bartlett, 2007). We finally get the result

$$\left(C^t_{EA}\right)^2 = \frac{1}{\lambda_2}\left(\frac{1}{1+\frac{1}{\frac{\lambda_2}{\lambda_1}+\frac{\lambda_2}{\lambda}}}.\right)tr(B).$$

Plugging it into Eq. A.3 gives the desired bounds on the Rademacher complexity of the EA target hypothesis class. ■

## A.3 Proof of Theorem 3.3.5: Complexity for EA++

**Proof.** In this section, we bound the complexity of the target hypothesis class $\mathscr{J}^s_{++}$ for EA++. The base hypothesis class $\mathscr{H}_{++}$ in Eq. 4.3 (of the original paper) in source and target hypotheses. So the complexity of source class $\mathscr{J}^s_{++}$ can be obtained by replacing adequate terms. We are interested in the complexity of the hypothesis class $\mathscr{J}^t_{++}$ which is defined as $\mathscr{J}^t_{++} := \{h_2 : \mathscr{X} \mapsto \mathbb{R}, (h_1,h_2) \in \mathscr{H}_{++}\}$, where $h_1$ is not fixed a priori.

The Rademacher complexity of $\mathscr{J}^t_{++}$ is defined as

$$\hat{R}_n(\mathscr{J}^t_{++}) = E_\sigma\left[\sup_{(h_1,h_2)\in\mathscr{H}_{++}}\left|\frac{2}{l_t}\sum_{i=1}^{l_t}\sigma_i h_2(x_i)\right|\right] \tag{A.5}$$

We proceed similar to the complexity proof of EA given in previous section. Note that $h_2 \in \mathscr{J}^t_{++} \implies -h_2 \in \mathscr{J}^t_{++}$ since $(h_1,h_2) \in \mathscr{H}_{++} \implies (-h_1,-h_2) \in \mathscr{H}_{++}$. This means that we can remove the absolute value sign from Eq. A.5. Since, $\forall i, h_2(x_i) = \langle k(x_i,\cdot),h_2\rangle$, we can restrict the supremum to $h_1$ and $h_2$ that are in the span of all samples and also in $\mathscr{H}_{++}$. The restricted condition on $(h_1,h_2)$ then becomes

$$\left\{(h_\alpha,h_\beta) : \lambda_1\alpha'K\alpha+\lambda_2\beta'K\beta+\lambda(\alpha-\beta)'K(\alpha-\beta)+\lambda_u(\alpha-\beta)'M(\alpha-\beta)\le 1\right\}$$
$$= \left\{(h_\alpha,h_\beta) : (\alpha'\,\beta')N(\alpha'\,\beta')'\le 1\right\}$$

where

$$M = \begin{pmatrix} D \\ E \\ F \end{pmatrix}\begin{pmatrix} D' & E' & F' \end{pmatrix},$$

$$N = \begin{pmatrix} (\lambda_1+\lambda)K & -\lambda K \\ -\lambda K & (\lambda_2+\lambda)K \end{pmatrix} + \lambda_u\begin{pmatrix} M & -M \\ -M & M \end{pmatrix},$$

and $K$ is the kernel matrix for source labeled, target labeled and target unlabeled samples. Using the reproducing kernel property, we get

$$\hat{R}_n(\mathscr{J}^t_{++}) = \frac{2}{l_t} E_\sigma \sup_{(\alpha,\beta)\in\mathbb{R}^{l_s+l_t+l_u}} \left\{ \sigma'(C'\,B\,E)\beta : (\alpha'\,\beta')N(\alpha'\,\beta')' \leq 1 \right\}.$$

Using Eq. A.2, the supremum in the above equation becomes $||(N^{-1})^{1/2}_{22}(C'\,B\,E)'\sigma||$.

If the matrix $N$ is not full rank, we can project $\beta$ and $\alpha$ onto the column space of $K$ without changing the supremum (as it is done in the previous proof). So, the sup is restricted to the projected $\alpha_{pr}$ and $\beta_{pr}$, and the expression for Rademacher complexity can be rewritten as

$$\hat{R}_n(\mathscr{J}^t_{++}) = \frac{2}{l_t} E_\sigma \sup_{\alpha_{pr},\beta_{pr}\in ColSpace\{K\}} \left\{ \sigma'(C'\,B\,E)\beta_{pr} : (\alpha'_{pr}\beta'_{pr})N(\alpha'_{pr}\beta'_{pr})' \leq 1 \right\}.$$

We proceed in a manner similar to the previous proof and diagonalize the kernel matrix $K$ to get orthonormal bases $U$ corresponding the nonzero eigenvalues ($K = U'\Lambda U$). $\Lambda$ is a diagonal matrix of size $r \times r$, containing just the nonzero eigenvalues and $r$ is the rank of matrix $K$. Since $\alpha_{pr}$ and $\beta_{pr}$ are in the span of column space of $K$, there exist $a_s$ and $b$ such that $\alpha_{pr} = Ua$, $\beta_{pr} = Ub$.

The expression for complexity now becomes,

$$\hat{R}_n(\mathscr{J}^t_{++}) = \frac{2}{l_t} E_\sigma \sup \left\{ \sigma' W b : (a'\,b')P(a'\,b')' \leq 1 \right\}$$

where $W = (C'\,B\,E)U$ and

$$P = \begin{pmatrix} (\lambda_1+\lambda)\Lambda & -\lambda\Lambda \\ -\lambda\Lambda & (\lambda_2+\lambda)\Lambda \end{pmatrix} + \lambda_u \begin{pmatrix} V' & 0 \\ 0 & V' \end{pmatrix} \begin{pmatrix} M & -M \\ -M & M \end{pmatrix} \begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix}$$

The solution to the above maximization problem is given by $||(P^{-1})^{1/2}_{22}W'\sigma||$. Using Kahane-Khintchine inequality and taking similar steps as in Eq. A.4, we get the following result:

$$\frac{2C^t_{++}}{2^{1/4} l_t} \leq \hat{R}_n(\mathscr{J}^t_{++}) \leq \frac{2C^t_{++}}{l_t}, \tag{A.6}$$

where $\left(C^t_{++}\right)^2 = tr\{W(P^{-1})_{22}W'\}$.

Let $T$ be the first term in the above expression for $P$. The second term can be written as $RR'$ where

$$R = \begin{pmatrix} V' & 0 \\ 0 & V' \end{pmatrix} \begin{pmatrix} D \\ E \\ F \\ D \\ E \\ F \end{pmatrix}$$

Using the matrix inversion lemma, we have $(T + \lambda_u RR')^{-1} = T^{-1} - \lambda_u T^{-1} R(I + \lambda_u R' T^{-1} R)^{-1} R' T^{-1}$. The term $tr\{W(T^{-1})_{22} W'\}$ evaluates to the same expression as the complexity of EA in previous proof. The second term can also be reduced in terms of original kernel submatrices by performing algebraic manipulations on eigenbases using similar steps as used in (Rosenberg and Bartlett, 2007). We finally get the result

$$\left(C^t_{++}\right)^2 = \left(\frac{1}{\lambda_2 + \left(\frac{1}{\lambda_1} + \frac{1}{\lambda}\right)^{-1}}\right) tr(B) - \lambda_u \left(\frac{\lambda_1}{\lambda\lambda_1 + \lambda\lambda_2 + \lambda_1\lambda_2}\right)^2 tr\left(E(I + kF)^{-1} E'\right),$$

where $k = \frac{\lambda_u(\lambda_1+\lambda_2)}{\lambda\lambda_1+\lambda\lambda_2+\lambda_1\lambda_2}$. Plugging it into Eq. A.6 gives the desired bounds on the Rademacher complexity of EA++ target hypothesis class. ■

# APPENDIX B

# DISTRIBUTED LEARNING

In all cases below we reduce to the *indexing problem* (Kushilevitz and Nisan, 1997): Let $A$ have $n$ bits either 0 or 1, and $B$ has an index $i \in [n]$. It requires $\Omega(n)$ one-way communication from $A$ to $B$ for $B$ to determine if $A$'s $i$th bit is 0 or 1, even allowing a $1/3$ probability of failure under randomized algorithms.

## B.1 Lower Bounds for One-Way Linear Separators

**Theorem B.1.1** *Using only one-way communication from A to B, it requires $\Omega(1/\varepsilon)$ communication to find an $\varepsilon$-error linear classifier in $\mathbb{R}^2$.*

**Proof.** We consider linear separators in $\mathbb{R}^2$ and suppose that points in $D_A$ and $D_B$ are distributed (almost) on the perimeter of a circle. This generalizes to higher dimensional settings by restricting points to lie on a 2-dimensional linear subspace. Figure B.1 shows a typical example where $D_A$ has exactly $1/\varepsilon$ negative points around a circle (each lies almost on the circle). These points form $1/2\varepsilon$ pairs of points, each close enough to each other and to the circle that they only effect points within the pair. Each pair can have two configurations:

- **Case 1.** left point just inside the circle and right point just outside the circle (red disks)
- **Case 2.** right point just inside the circle and left point just outside the circle (red boxes)

$D_B$ has only one positive point $b^+$ (blue plus) that interacts with exactly one pair of points from $D_A$, but $B$ does not know which pair to interact with ahead of time. The positive point $b^+$ is placed close to the arc of the circle with equal arc length to the negative points from $D_A$ on either sides such that it is just inside the circle.

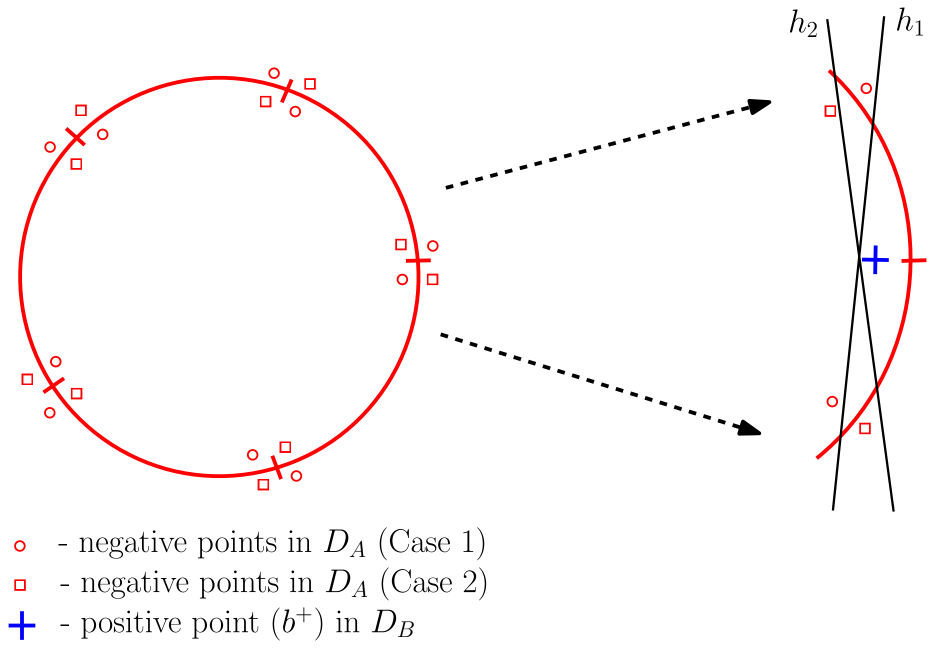

$h_2$
$h_1$
○ - negative points in $D_A$ (Case 1)
□ - negative points in $D_A$ (Case 2)
+ - positive point ($b^+$) in $D_B$

**Figure B.1**. An example to prove the lower bound results for one-way communication with linear separators. The figure on the left shows the distribution of the negative points in $D_A$ for Case 1 and Case 2. The right figure zooms in on only a small arc of the circle and shows what happens when $B$ decides the final classifier based on its single positive point $b^+$ and all negative points from $D_A$.

**Claim B.1.1** *Let $Z_j$ be a pair of points in $D_A$ and let $x_j$ be the position of $b^+$ (with respect to $Z_j$), as shown in Figure B.1. If $D_B$ has a point $b^+$ at $x_j$, then A needs to send at least one bit of information about $Z_j$ to B, for B to learn the perfect classifier.*

**Proof.** Suppose $A$ sends no information to $B$. In order to learn an optimal classifier, $B$ makes the classifier tangent to the circle, but offset to just include its point $b^+$. However, the point $b^+$ is so positioned that it always forces the classifier learned by $B$ to misclassify either the left negative point in $D_A$ (classifier $h_1$ in Figure B.1, if Case 1) or the right negative point in $D_A$ (classifier $h_2$ in Figure B.1, if Case 2), whichever point is just outside the circle. $B$ can guess Case 1 and angle the classifier to the left point, or guess Case 2 and angle the classifier to the right point. But in either case, without any information from $A$, it will be wrong half the time. However, if $A$ sends a single bit of information denoting whether some negative point pair belongs to Case 1 or Case 2, then $B$ can use this information to learn a perfect separator with zero error. ■

If we increase the number of points to $n$, by putting $\epsilon n$ identical points at each point in the construction then such misclassified points cause an $\varepsilon$ error.

We also note that, each pair of points are independent of the others and a classifier learned for any one negative point pair (in $D_A$) works for other negative point pairs.

In the above case, $A$ has $1/2\varepsilon$ point pairs that are all negative. Each pair is far enough away from all other pairs so as not to affect each other. $B$ has 1 positive point placed (as shown on the right of Figure B.1) for some point pair, not known to $A$. To reduce this problem to indexing, we let each of $A$'s point pairs correspond to one bit which is 0 (if Case 1) or 1 (if Case 2). $B$ needs to determine if the $i$th bit (corresponding to the negative point pair in $D_A$, which $b^+$ needs to deal with) is 0 or 1. This requires $\Omega(1/\varepsilon)$ one-way communication from $A$ to $B$, proving the lemma. ■

## B.2 Lower Bounds for One-Way Noise Detection

Although in the noiseless nonagnostic setting we can guarantee finding optimal separators with one-way communication, under the assumption they exist, we cannot detect definitively if they do exist. For intervals, the difficult case is when $A$ has only negative points, and for axis-aligned rectangles the difficult case is more general.

**Lemma B.2.1** *It requires $\Omega(|D_A|)$ one-way communication from A to B to determine if there exists a perfect classifier $h \in \mathcal{I}$.*

**Proof.** Consider the case where $A$ has $n/2$ points and they are all negative. All of its points have values in $[2n]$ and are even. $B$ has 2 positive points and $n/2-2$ negative, its points have values in $[2n+1]$ and are all odd. Its two positive points are consecutive odd points, say, $2i-1$ and $2i+1$. If $A$ has a point at index $2i$, then there is no perfect classifier, if it does not, then there is.

This is precisely the indexing problem with $A$'s points corresponding to a 1 if they exist for index $2i$ and to a 0 if they do not, and for $B$'s index $i$ corresponding to the value $i$ for which it has positive points at $2i-1$ and $2i+1$. Thus, it requires $\Omega(|D_A|)$ one-way communication, proving the lemma. ■

**Lemma B.2.2** *It requires $\Omega(|D_A|)$ one-way communication from A to B to determine if there exists a perfect classifier $h \in \mathcal{R}_2$, even if A and B have positive and negative points.*

**Proof.** Let $A$ and $B$ both have a positive point at $(2n, 0)$ and a negative point at $(0, 2n)$. $A$ also has a set of $n/2 - 2$ negative points at locations $(2i, 2i)$ for some distinct values of $i \in [n]$. $B$ has a (variable) positive point at some location $(2i-1, 2i+1)$ for $i \in [n]$. There exists a perfect classifier $h \in \mathcal{R}_2$ if, and only if, $A$ has no point at $(2i, 2i)$ where $i$ is the index of $B$'s variable point.

Again, this is precisely the indexing problem. $A$'s points along the diagonal correspond to $n$ bits being 1 if a point exists, and 0 if not, for each index $i$. $B$'s index corresponds to the value $i$ of its variable point. Thus, it requires $\Omega(|D_A|)$ one-way communication, proving the lemma. ■

# REFERENCES

ABE, N. AND MAMITSUKA, H. 1998. Query learning strategies using boosting and bagging. In *ICML'98*. San Francisco, USA.

ABERNETHY, J., BARTLETT, P., AND RAKHLIN, A. 2007. Multitask learning with expert advice. In *COLT'07*. San Diego, USA.

AGARWAL, A. AND DUCHI, J. 2011. Distributed delayed stochastic optimization. In *NIPS'11*. Granada, Spain.

AGARWAL, A., GERBER, S., AND DAUMÉ III, H. 2010. Learning multiple tasks using manifold regularization. In *NIPS'10*. Vancouver, Canada.

AGARWAL, A., RAKHLIN, A., AND BARTLETT, P. 2008. Matrix regularization techniques for online multitask learning. *Technical report, EECS Department, University of California, Berkeley*.

ALLENBY, G. M. AND ROSSI, P. E. 1998. Marketing models of consumer heterogeneity. *Journal of Econometrics 89,* 1-2, 57–78.

ANDO, R. K. AND ZHANG, T. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research 6*, 1817–1853.

ANDRZEJEWSKI, D., ZHU, X., AND CRAVEN, M. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *ICML'09*. Montreal, Canada.

ANTHONY, M. AND BARTLETT, P. L. 2009. *Neural Network Learning: Theoretical Foundations* 1st Ed. Cambridge University Press, New York, USA.

ARGYRIOU, A., EVGENIOU, T., AND PONTIL, M. 2007a. Multi-task feature learning. In *NIPS'07*. Vancouver, Canada.

ARGYRIOU, A., EVGENIOU, T., AND PONTIL, M. 2008. Convex multi-task feature learning. *Machine Learning 73,* 3, 243–272.

ARGYRIOU, A., MICCHELLI, C. A., PONTIL, M., AND YING, Y. 2007b. A spectral regularization framework for multi-task structure learning. In *NIPS'07*. Vancouver, Canada.

ARNOLD, A. AND COHEN, W. W. 2008. Intra-document structural frequency features for semi-supervised domain adaptation. In *CIKM'08*. Napa Valley, USA.

ARORA, N., ALLENBY, G. M., AND GINTER, J. L. 1998. A hierarchical bayes model of primary and secondary demand. *Marketing Science 17*, 29–44.

ASUNCION, A., SMYTH, P., AND WELLING, M. 2008. Asynchronous distributed learning of topic models. In *NIPS'08*. Vancouver, Canada.

AUER, P., BURGSTEINER, H., AND MAASS, W. 2002. Reducing communication for distributed learning in neural networks. In *ICANN '02*. London, UK.

AUER, P. AND WARMUTH, M. K. 1998. Tracking the best disjunction. *Machine Learning 32,* 2, 127–150.

BAKKER, B. AND HESKES, T. 2003. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research 4*, 83–99.

BALCAN, M.-F. AND BLUM, A. 2005. A PAC-style model for learning from labeled and unlabeled data. In *COLT'05*. Bertinoro, Italy.

BALCAN, M.-F. AND BLUM, A. 2006. *An Augmented PAC Model for Semi-Supervised Learning*. MIT Press, Chapter 22, 397–420.

BALCAN, M.-F. AND BLUM, A. 2010. A discriminative model for semi-supervised learning. *Journal of the ACM 57,* 3.

BALCAN, M.-F., BLUM, A., FINE, S., AND MANSOUR, Y. 2012. Distributed learning, communication complexity and privacy. In *COLT'12*. Edinburgh, Scotland.

BALDRIDGE, J. AND OSBORNE, M. 2004. Active Learning and the Total Cost of Annotation. In *EMNLP'04*. Barcelona, Spain.

BALUJA, S. 1999. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *NIPS'99*. Vancouver, Canada.

BAUER, E. AND KOHAVI, R. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning 36,* 1-2.

BEKKERMAN, R., BILENKO, M., AND LANGFORD, J. 2011. *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, New York, USA.

BELKIN, M., MATVEEVA, I., AND NIYOGI, P. 2004. Regularization and semi-supervised learning on large graphs. In *COLT'04*. Banff, Canada.

BEN-DAVID, S., BLITZER, J., CRAMMER, K., AND PEREIRA, F. 2006. Analysis of representations for domain adaptation. In *NIPS'06*. Vancouver, Canada.

BENNETT, K. P. AND DEMIRIZ, A. 1998. Semi-supervised support vector machines. In *NIPS'98*. Vancouver, Canada.

BICKEL, S. AND SCHEFFER, T. 2006. Dirichlet-enhanced spam filtering based on biased samples. In *NIPS'06*. Vancouver, Canada.

BIE, T. D. AND CRISTIANINI, N. 2003. Convex methods for transduction. In *NIPS'03*. Vancouver, Canada.

BIE, T. D. AND CRISTIANINI, N. 2006. *Semi-supervised learning using semi-definite programming*. Cambridge-Massachussets: MIT Press.

BLACKWELL, D. 1956. An analog of the minmax theorem for vector payoffs. *Pacific Journal of Mathematics* 1, 1–8.

BLEI, D. M., NG, A. Y., AND JORDAN, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research 3*, 993–1022.

BLITZER, J., CRAMMER, K., KULESZA, A., PEREIRA, F., AND WORTMAN, J. 2007a. In *NIPS'07*. Vancouver, Canada.

BLITZER, J., DREDZE, M., AND PEREIRA, F. 2007b. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL'07*. Prague, Czech Republic.

BLITZER, J., MCDONALD, R., AND PEREIRA, F. 2006. Domain adaptation with structural correspondence learning. In *EMNLP'06*. Sydney, Australia.

BLOODGOOD, M. AND VIJAY-SHANKER, K. 2009. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. In *CoNLL'09*. Boulder, USA.

BLUM, A. AND CHAWLA, S. 2001. Learning from labeled and unlabeled data using graph mincuts. In *ICML'01*. San Francisco, USA.

BLUM, A. AND MITCHELL, T. 1998. Combining labeled and unlabeled data with co-training. In *COLT'98*. Madison, USA.

BONILLA, E. V., CHAI, K. M. A., AND WILLIAMS, C. K. I. 2007. Multi-task gaussian process prediction. In *NIPS'07*. Vancouver, Canada.

BOTTOU, L. AND BOUSQUET, O. 2008. The tradeoffs of large scale learning. In *NIPS'08*. Vancouver, Canada.

BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. 2011. *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Now Publishers.

BREFELD, U., BÜSCHER, C., AND SCHEFFER, T. 2005. Multi-view discriminative sequential learning. In *ECML'05*. Berlin, Germany.

BREFELD, U., GÄRTNER, T., SCHEFFER, T., AND WROBEL, S. 2006. Efficient co-regularised least squares regression. In *ICML'06*. Pittsburgh, USA.

BREFELD, U. AND SCHEFFER, T. 2006. Semi-supervised learning for structured output variables. In *ICML'06*. Pittsburgh, USA.

BREIMAN, L. 1996. Bagging predictors. *Machine Learning 24,* 2, 123–140.

BRINKER, K. 2003. Incorporating diversity in active learning with support vector machines. In *ICML'03*. Washington, USA.

CALLISON-BURCH, C., TALBOT, D., AND OSBORNE, M. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *ACL '04*. Barcelona, Spain.

CARAGEA, D., SILVESCU, A., AND HONAVAR, V. 2000. Incremental and distributed learning with support vector machines. In *NCAI'00*. Austin, USA.

CARUANA, R. 1997. Multitask learning. *Machine Learning 28,* 1.

CAVALLANTI, G., CESA-BIANCHI, N., AND GENTILE, C. 2008. Linear algorithms for online multitask classification. In *COLT'08*. Helsinki, Finland.

CESA-BIANCHI, N., CONCONI, A., AND GENTILE, C. 2005. A second-order perceptron algorithm. *SIAM Journal of Computing 34,* 3, 640–668.

CESA-BIANCHI, N., FREUND, Y., HAUSSLER, D., HELMBOLD, D. P., SCHAPIRE, R. E., AND WARMUTH, M. K. 1997. How to use expert advice. *J. ACM 44,* 3, 427–485.

CESA-BIANCHI, N., GENTILE, C., AND ORABONA, F. 2009. Robust bounds for classification via selective sampling. In *ICML'09*. Montreal, Canada.

CESA-BIANCHI, N., GENTILE, C., AND ZANIBONI, L. 2006. Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research 7*.

CESA-BIANCHI, N. AND LUGOSI, G. 2006. *Prediction, Learning, and Games*. Cambridge University Press, New York, USA.

CHAFI, H., SUJEETH, A. K., BROWN, K. J., LEE, H., ATREYA, A. R., AND OLUKOTUN, K. 2011. A domain-specific approach to heterogeneous parallelism. In *PPoPP '11*. San Antonio, USA.

CHAN, Y. S. AND NG, H. T. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *ACL'06*. Sydney, Australia.

CHAN, Y. S. AND NG, H. T. 2007. Domain adaptation with active learning for word sense disambiguation. In *ACL'07*. Prague, Czech Republic.

CHANG, C.-C. AND LIN, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2,* 3.

CHANG, M.-W., CONNOR, M., AND ROTH, D. 2010. The necessity of combining adaptation methods. In *EMNLP'10*. Cambridge, MA.

CHAPELLE, O., SHIVASWAMY, P., VADREVU, S., WEINBERGER, K., ZHANG, Y., AND TSENG, B. 2010. Multi-task learning for boosting with application to web search ranking. In *KDD'10*. Washington, USA.

CHAPELLE, O., WESTON, J., AND SCHÖLKOPF, B. 2003. Cluster Kernels for Semi-Supervised Learning. In *NIPS'02*. Cambridge, USA.

CHAPELLE, O. AND ZIEN, A. 2005. Semi–supervised classification by low density separation. In *AISTATS'05*. Barbados.

CHAWLA, N., BOWYER, K., HALL, L., AND KEGELMEYER, W. 2002. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research 16*, 321–357.

CHAZELLE, B. 2000. *The Discrepancy Method.* Cambridge.

CHELBA, C. AND ACERO, A. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language 20,* 4, 382–399.

CHEN, R., WENG, X., HE, B., AND YANG, M. Large graph processing in the cloud. In *SIGMOD'10*. Indianapolis, USA.

CHU, C.-T., KIM, S. K., LIN, Y.-A., YU, Y., BRADSKI, G., NG, A. Y., AND OLUKOTUN, K. 2007a. Map-reduce for machine learning on multicore. In *NIPS'07*. Vancouver, Canada.

CHU, W., SINDHWANI, V., GHAHRAMANI, Z., AND KEERTHI, S. S. 2007b. Relational learning with gaussian processes. In *NIPS'07*. Vancouver, Canada.

COLLINS, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP'02*. Stroudsburg, USA.

COLLINS, M. AND SINGER, Y. 1999. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. 100–110.

CONDIE, T., CONWAY, N., ALVARO, P., HELLERSTEIN, J. M., ELMELEEGY, K., AND SEARS, R. 2010. Mapreduce online. In *NSDI'10*. Berkeley, USA.

CORMODE, G., MUTHUKRISHNAN, S., AND YI, K. 2008. Algorithms for distributed functional monitoring. In *SODA'08*. San Francisco, USA.

CORMODE, G., MUTHUKRISHNAN, S., YI, K., AND ZHANG, Q. 2010. Optimal sampling from distributed streams. In *PODS'10*. Indianapolis, USA.

CRAMMER, K., DEKEL, O., KESHET, J., SHALEV-SHWARTZ, S., AND SINGER, Y. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research 7*, 551–585.

CRAMMER, K., KULESZA, A., AND DREDZE, M. 2009. Adaptive regularization of weight vectors. In *NIPS'09*. Vancouver, Canada.

CRAMMER, K. AND SINGER, Y. 2003. A new family of online algorithms for category ranking. *Journal of Machine Learning Research 3*, 1025–1058.

DAGAN, I. AND ENGELSON, S. P. 1995. Committee-based sampling for training probabilistic classifiers. In *ICML'95*. Tahoe City, USA.

DAI, W., JIN, O., XUE, G.-R., YANG, Q., AND YU, Y. 2009. Eigentransfer: A unified framework for transfer learning. In *ICML'09*. Montreal, Canada.

DAI, W., XUE, G.-R., YANG, Q., AND YU, Y. 2007a. Co-clustering based classification for out-of-domain documents. In *KDD'07*. San Jose, USA.

DAI, W., XUE, G.-R., YANG, Q., AND YU, Y. 2007b. Transferring Naive Bayes classifiers for text classification. In *AAAI'07*. Vancouver, Canada.

DAI, W., YANG, Q., XUE, G.-R., AND YU, Y. 2007c. Boosting for transfer learning. In *ICML'07*. Corvallis, USA.

DAS, A. S., DATAR, M., GARG, A., AND RAJARAM, S. 2007. Google news personalization: Scalable online collaborative filtering. In *WWW'07*. Banff, Canada.

DASGUPTA, S. AND HSU, D. 2008. Hierarchical sampling for active learning. In *ICML'08*. Helsinki, Finland.

DASGUPTA, S., KALAI, A. T., AND MONTELEONI, C. 2009. Analysis of perceptron-based active learning. *Journal of Machine Learning Research 10*, 281–299.

DASGUPTA, S., LITTMAN, M. L., AND MCALLESTER, D. A. 2001. Pac generalization bounds for co-training. In *NIPS'01*. Vancouver, Canada.

DAUMÉ, III, H. AND MARCU, D. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research 26,* 1, 101–126.

DAUMÉ III, H. 2004. Notes on CG and LM-BFGS optimization of logistic regression.

DAUMÉ III, H. 2007. Frustratingly easy domain adaptation. In *ACL'07*. Prague, Czech Republic.

DAUMÉ III, H. 2009. Bayesian multitask learning with latent hierarchies. In *UAI'09*. Montreal, Canada.

DAUMÉ III, H., KUMAR, A., AND SAHA, A. 2010. Frustratingly easy semi-supervised domain adaptation. In *ACL 2010 Workshop on Domain Adaptation for NLP (DANLP)*. Uppsala, Sweden.

DAUMÉ III, H., PHILLIPS, J., SAHA, A., AND VENKATASUBRAMANIAN, S. 2012. Protocols for learning classifiers on distributed data. In *AISTATS'12*. La Palma, Canary Islands.

DEAN, J. AND GHEMAWAT, S. 2004. Mapreduce: Simplified data processing on large clusters. In *OSDI'04*. San Francisco, USA.

DEKEL, O., GENTILE, C., AND SRIDHARAN, K. 2010a. Robust selective sampling from single and multiple teachers. In *COLT'10*. Haifa, Israel.

DEKEL, O., GILAD-BACHRACH, R., SHAMIR, O., AND XIAO, L. 2010b. Optimal distributed online prediction using mini-batches. arXiv:1012.1367.

DEKEL, O., LONG, P. M., AND SINGER, Y. 2006. Online multitask learning. In *COLT'06*. Pittsburgh, USA.

DEMIRIZ, A. AND BENNETT, K. P. 2000. *Optimization Approaches to Semi-Supervised Learning*. Boston: Kluwer Academic Publishers.

DESANTIS, A., MARKOWSKY, G., AND WEGMAN, M. N. 1988. Learning probabilistic prediction functions. In *COLT'88*. Cambridge, USA.

DONMEZ, P., CARBONELL, J. G., AND SCHNEIDER, J. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD'09*. Paris, France.

DREDZE, M., CRAMMER, K., AND PEREIRA, F. 2008. Confidence-weighted linear classification. In *ICML'08*. Helsinki, Finland.

DREDZE, M., KULESZA, A., AND CRAMMER, K. 2010. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning 79,* 1-2, 123–149.

DRUCK, G., SETTLES, B., AND MCCALLUM, A. 2009. Active learning by labeling features. In *EMNLP'09*. Singapore.

DUAN, L., TSANG, I. W., XU, D., AND CHUA, T.-S. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML'09*. Montreal, Canada.

DUCHI, J., AGARWAL, A., AND WAINWRIGHT, M. 2010. Distributed dual averaging in networks. In *NIPS'10*. Vancouver, Canada.

DUH, K., SUZUKI, J., AND NAGATA, M. 2011. Distributed learning-to-rank on streaming data using alternating direction method of multipliers. In *NIPS BigLearn Workshop*.

EKANAYAKE, J., LI, H., ZHANG, B., GUNARATHNE, T., BAE, S.-H., QIU, J., AND FOX, G. 2010. Twister: A runtime for iterative mapreduce. In *HPDC'10*. Chicago, USA.

EVGENIOU, T., MICCHELLI, C. A., AND PONTIL, M. 2005. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research 6*, 615–637.

EVGENIOU, T. AND PONTIL, M. 2004. Regularized multitask learning. In *KDD'04*. Seattle, USA.

FOSTER, D. P. AND VOHRA, R. V. 1993. A randomization rule for selecting forecasts. *Operations Research 41,* 4, 704–709.

FREUND, Y. AND SCHAPIRE, R. E. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT'95*. Barcelona, Spain.

FREUND, Y., SEUNG, H. S., SHAMIR, E., AND TISHBY, N. 1997. Selective sampling using the query by committee algorithm. *Machine Learning 28,* 2-3, 133–168.

FUJINO, A., UEDA, N., AND SAITO, K. 2005. A hybrid generative/discriminative approach to semi-supervised classifier design. In *AAAI'05*. Pittsburgh, USA.

FUNG, G. AND MANGASARIAN, O. L. 2001. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software 15*, 29–44.

GANJISAFFAR, Y., DEBEAUVAIS, T., JAVANMARDI, S., CARUANA, R., AND LOPES, C. V. 2011. Distributed tuning of machine learning algorithms using mapreduce clusters. In *Proceedings of the Third Workshop on Large Scale Data Mining: Theory and Applications*. LDMTA '11. San Diego, California.

GENTILE, C. 2001. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research 2*, 213–242.

GETZ, G., SHENTAL, N., AND DOMANY, E. 2006. Semi-supervised learning – a statistical physics approach. *CoRR abs/cs/0604011.*

GHOTING, A., KRISHNAMURTHY, R., PEDNAULT, E., REINWALD, B., SINDHWANI, V., TATIKONDA, S., TIAN, Y., AND VAITHYANATHAN, S. 2011. SystemML: Declarative machine learning on mapreduce. In *ICDE'11*. Bellevue, USA.

GRIRA, N., CRUCIANU, M., AND BOUJEMAA, N. 2005. Active semi-supervised fuzzy clustering for image database categorization. In *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*. MIR'05. Hilton, Singapore.

GUPTA, S. K., PHUNG, D., ADAMS, B., TRAN, T., AND VENKATESH, S. 2010. Non-negative shared subspace learning and its application to social media retrieval. In *KDD'10*. Washington, USA.

HAGHIGHI, A. AND KLEIN, D. 2006. Prototype-driven learning for sequence models. In *HLT-NAACL'06*. New York, USA.

HALEVY, A. Y., NORVIG, P., AND PEREIRA, F. 2009. The unreasonable effectiveness of data. *IEEE Intelligent Systems 24,* 2, 8–12.

HERBSTER, M. AND WARMUTH, M. K. 2001. Tracking the best linear predictor. *Journal of Machine Learning Research 1*, 281–309.

HESKES, T. 2000. Empirical bayes for learning to learn. In *ICML'00*. San Francisco, USA.

HINDMAN, B., KONWINSKI, A., ZAHARIA, M., AND STOICA, I. 2009. A common substrate for cluster computing. In *HotCloud'09*. San Diego, USA.

HOFMANN, T. AND BUHMANN, J. M. 1998. Active data clustering. In *NIPS'97*. Denver, USA.

HSU, D. AND LANGFORD, J. 2011. The end of the beginning of active learning. http://hunch.net/?p=1800.

HUA ZHOU, Z., CHUAN ZHAN, D., AND YANG, Q. 2007. Semi-supervised learning with very few labeled training examples. In *AAAI'07*. Vancouver, Canada.

HUANG, J., SMOLA, A. J., GRETTON, A., BORGWARDT, K. M., AND SCHÖLKOPF, B. 2007. Correcting sample selection bias by unlabeled data. In *NIPS'07*. Vancouver, Canada, 601–608.

HUANG, L., JORDAN, M. I., JOSEPH, A., GAROFALAKIS, M., AND TAFT, N. 2006. In-network pca and anomaly detection. In *NIPS'06*. Vancouver, Canada.

HUANG, Y. AND MITCHELL, T. M. 2006. Text clustering with extended user feedback. In *SIGIR'06*. Seattle, USA.

HWA, R. 2001. On minimizing training corpus for parser acquisition. In *Proceedings of the 2001 workshop on Computational Natural Language Learning - Volume 7*. ConLL'01. Toulouse, France.

ISARD, M., BUDIU, M., YU, Y., BIRRELL, A., AND FETTERLY, D. 2007. Dryad: Distributed data-parallel programs from sequential building blocks. In *EuroSys'07*. Lisbon, Portugal.

J. ABERNETHY, F. BACH, T. E. AND VERT, J.-P. 2006. Low-rank matrix factorization with attributes. *Technical Report*.

JIANG, J. 2008. A literature survey on domain adaptation of statistical classifiers.

JOACHIMS, T. 1999. Transductive inference for text classification using support vector machines. In *ICML'99*. Bled, Slovenia.

JOACHIMS, T. 2003. Transductive learning via spectral graph partitioning. In *ICML'03*. Washington, USA.

JONES, R. 2005. Learning to extract entities from labeled and unlabeled text. PhD Dissertation, Carnegie Mellon University.

KALAI, A. AND VEMPALA, S. 2002. Geometric algorithms for online optimization. In *Journal of Computer and System Sciences*. 26–40.

KANG, Z., GRAUMAN, K., AND SHA, F. 2011. Learning with whom to share in multi-task feature learning. In *ICML'11*. Bellevue, USA.

KAPOOR, A., HORVITZ, E., AND BASU, S. 2007. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *IJCAI'07*. Hyderabad, India.

KEARNS, M. 1998. Efficient noise-tolerant learning from statistical queries. *Journal of ACM 45,* 6, 983–1006.

KEARNS, M. AND VAZIRANI, U. 1994. *An introduction to computational learning theory*. MIT Press, Cambridge, USA.

KIVINEN, J., SMOLA, A. J., AND WILLIAMSON, R. C. 2004. Online learning with kernels. *IEEE Transactions on Signal Processing 52,* 8, 2165–2176.

KIVINEN, J. AND WARMUTH, M. K. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation 132,* 1, 1–63.

KONDOR, R. I. AND LAFFERTY, J. D. 2002. Diffusion kernels on graphs and other discrete input spaces. In *ICML'02*. Sydney, Australia.

KOWALCZYK, W. AND VLASSIS, N. 2005. Newscast EM. In *NIPS'05*. Vancouver, Canada.

KUBAT, M. AND MATWIN, S. 1997. Addressing the curse of imbalanced training sets: One-sided selection. In *ICML'97*. Nashville, USA.

KULIS, B., SUSTIK, M. A., AND DHILLON, I. S. 2009. Low-rank kernel learning with bregman matrix divergences. *Journal of Machine Learning Research 10.*

KUSHILEVITZ, E. AND NISAN, N. 1997. *Communication Complexity*. Cambridge University Press.

LANG, K. J. AND BAUM, E. B. 1992. Query learning can work poorly when a human oracle is used. In *IJCNN'92*. Beijing, China.

LANGFORD, J., SMOLA, E. J., AND ZINKEVICH, M. 2009. Slow learners are fast. In *NIPS'09*. Vancouver, Canada.

LASKOV, P. AND LIPPMANN, R. 2010. Machine learning in adversarial environments. *Machine Learning 81,* 2.

LATALA, R. AND OLESZKIEWICZ, K. 1994. On the best constant in the Khinchin-Kahane inequality. *Studia Mathematica 109*, 101–104.

LAWRENCE, N. D. AND JORDAN, M. I. 2005. Semi-supervised learning via gaussian processes. In *NIPS'05*. Vancouver, Canada.

LAZAREVIC, A. AND OBRADOVIC, Z. 2001. The distributed boosting algorithm. In *KDD'01*. San Francisco, USA.

LEWIS, D. D. 1995. A sequential algorithm for training text classifiers: Corrigendum and additional data. *SIGIR Forum 29,* 2, 13–19.

LEWIS, D. D. AND CATLETT, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In *ICML'94*. New Brunswick, US.

LI, X. AND BILMES, J. 2007. A bayesian divergence prior for classifier adaptation. In *AISTATS'07*. San Juan, USA.

LI, Y. AND LONG, P. M. 2002. The relaxed online maximum margin algorithm. *Machine Learning 46,* 1-3, 361–387.

LIANG, P. AND JORDAN, M. I. 2008. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *ICML'08*. Helsinki, Finland.

LIBERTY, E., WOOLFE, F., MARTINSSON, P.-G., ROKHLIN, V., AND TYGERT, M. 2007. Randomized algorithms for the low-rank approximation of matrices. *PNAS 104,* 51.

LING, X., DAI, W., XUE, G.-R., YANG, Q., AND YU, Y. 2008. Spectral domain-transfer learning. In *KDD'08*. Las Vegas, USA.

LITTLESTONE, N. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning 2,* 4, 285–318.

LITTLESTONE, N. 1991. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In *COLT'91*. Santa Cruz, US.

LITTLESTONE, N. AND WARMUTH, M. K. 1994. The weighted majority algorithm. *Information and Computation 108,* 2, 212–261.

LIU, Q., LIAO, X., CARIN, H. L., STACK, J. R., AND CARIN, L. 2009. Semisupervised multitask learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31,* 6, 1074–1086.

LOMASKY, R., BRODLEY, C. E., AERNECKE, M., WALT, D., AND FRIEDL, M. 2007. Active class selection. In *ECML'07*. Warsaw, Poland.

LONG, M., CHENG, W., JIN, X., WANG, J., AND SHEN, D. 2010. Transfer learning via cluster correspondence inference. In *ICDM'10*. Sydney, Australia.

LONG, M., WANG, J., DING, G., CHENG, W., ZHANG, X., AND WANG, W. 2012. Dual transfer learning. In *SDM'12*. Anaheim, USA.

LOW, Y., GONZALEZ, J., KYROLA, A., BICKSON, D., GUESTRIN, C., AND HELLERSTEIN, J. M. 2012. Distributed graphlab: A framework for machine learning in the cloud. *CoRR abs/1204.6078*.

LUGOSI, G., PAPASPILIOPOULOS, O., AND STOLTZ, G. 2009. Online multi-task learning with hard constraints. In *COLT'09*. Montreal, Canada.

MAEIREIZO, B., LITMAN, D., AND HWA, R. 2004. Co-training for predicting emotions with spoken dialogue data. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. ACLdemo '04. Barcelona, Spain.

MAHOUT. 2012. Apache mahout: Scalable machine-learning and data-mining library. http://mahout.apache.org.

MANN, G., MCDONALD, R., MOHRI, M., SILBERMAN, N., AND WALKER, D. 2009. Efficient large-scale distributed training of conditional maximum entropy models. In *NIPS'09*. Vancouver, Canada.

MANN, G. S. AND MCCALLUM, A. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research 11*, 955–984.

MANSOUR, Y., MOHRI, M., AND ROSTAMIZADEH, A. 2009. Domain adaptation: Learning bounds and algorithms. In *COLT'09*. Montreal, Canada.

MATOUSEK, J. 1991. Approximations and optimal geometric divide-and-conquer. In *STOC'91*. New Orleans, USA.

MCDONALD, R., HALL, K., AND MANN, G. 2010. Distributed training strategies for the structured perceptron. In *NAACL HLT'10*. Los Angeles, California.

MELVILLE, P. AND MOONEY, R. J. 2004. Diverse ensembles for active learning. In *ICML'04*. Banff, Canada.

MERUGU, S. AND GHOSH, J. 2005. A distributed learning framework for heterogeneous data sources. In *KDD'05*. Chicago, USA.

MILLER, D. J. AND UYAR, H. S. 1997. A Mixture of Experts Classifier with Learning Based on Both Labeled and unlabeled data. *NIPS'97*.

MITCHELL, T. M. 1999. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*. San Sebastian, Spain.

MONTELEONI, C. AND KÄÄRIÄINEN, M. 2007. Practical online active learning for classification. In *IEEE CVPR Workshop on Online Learning for Classification*. Minneapolis, USA.

MUSLEA, I., MINTON, S., AND KNOBLOCK, C. A. 2000. Selective sampling with redundant views. In *AAAI*. Austin, USA, 621–626.

NEWMAN, D., ASUNCION, A., SMYTH, P., AND WELLING, M. 2008. Distributed inference for latent dirichlet allocation. In *NIPS'08*. Vancouver, Canada.

NEWMAN, D., ASUNCION, A., SMYTH, P., AND WELLING, M. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research 10*, 1801–1828.

NIGAM, K., MCCALLUM, A. K., THRUN, S., AND MITCHELL, T. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning 39,* 2-3, 103–134.

NOVIKOFF, A. 1962. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*. Vol. 12. New York, USA.

OLSSON, F. AND TOMANEK, K. 2009. An intrinsic stopping criterion for committee-based active learning. In *CoNLL'09*. Boulder, USA.

OUYANG, H. AND GRAY, A. G. 2011. Data-distributed weighted majority and online mirror descent. *CoRR abs/1105.2274*.

PALIT, I. AND REDDY, C. K. 2010. Parallelized boosting with map-reduce. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*. ICDMW '10.

PAN, S. J., NI, X., TAO SUN, J., YANG, Q., AND CHEN, Z. 2010a. Cross-domain sentiment classification via spectral feature alignment. In *WWW'10*. Raleigh, USA.

PAN, S. J., TSANG, I. W., KWOK, J. T., AND YANG, Q. 2010b. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks PP,* 99, 1–12.

PAN, S. J. AND YANG, Q. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering 22,* 10, 1345–1359.

PEARCE, R., GOKHALE, M., AND AMATO, N. M. 2010. Multithreaded asynchronous graph traversal for in-memory and semi-external memory. In *SC'10*. New Orleans, USA.

PILLONETTO, G., DINUZZO, F., AND DE NICOLAO, G. 2010. Bayesian online multitask learning of gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32,* 2, 193–205.

POWER, R. AND LI, J. 2010. Piccolo: Building fast, distributed programs with partitioned tables. In *OSDI'10*. Vancouver, Canada.

PREDD, J. B., KULKARNI, S. R., AND POOR, H. V. 2006. Distributed learning in wireless sensor networks. *IEEE Signal Processing Magazine*.

PROVOST, F. J., ARONIS, J., AND FISHER, H. 1996. Scaling up inductive learning with massive parallelism. In *Machine Learning*. 33–46.

PROVOST, F. J. AND HENNESSY, D. N. 1996. Scaling up: Distributed machine learning with cooperation. In *AAAI'96*. Portland, USA.

QI, G.-J., HUA, X.-S., RUI, Y., TANG, J., AND ZHANG, H.-J. 2008. Two-dimensional active learning for image classification. In *CVPR*. Anchorage, USA.

RAGHAVAN, H., MADANI, O., AND JONES, R. 2006. Active learning with feedback on features and instances. *Journal of Machine Learning Research 7*, 1655–1686.

RAI, P. AND DAUMÉ III, H. 2010. Infinite predictor subspace models for multitask learning. In *AISTATS'10*. Sardinia, Italy.

RAI, P., SAHA, A., DAUMÉ III, H., AND VENKATASUBRAMANIAN, S. 2010. Domain adaptation meets active learning. In *NAACL 2010 Workshop on Active Learning for NLP (ALNLP)*. Los Angeles, USA.

RAINA, R., NG, A. Y., AND KOLLER, D. 2006. Constructing informative priors using transfer learning. In *ICML'06*. Pittsburgh, USA.

RECHT, B., RE, C., WRIGHT, S. J., AND NIU, F. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS'11*. Granada, Spain.

REICHART, R., TOMANEK, K., HAHN, U., AND RAPPOPORT, A. 2008. Multi-task active learning for linguistic annotations. In *ACL'08*. Columbus, USA.

RILOFF, E., WIEBE, J., AND WILSON, T. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *CONLL'03*. Edmonton, Canada.

ROSENBERG, C., HEBERT, M., AND SCHNEIDERMAN, H. 2005. Semi-supervised self-training of object detection models. In *WACV-MOTION'05*. Washington, USA.

ROSENBERG, D. S. AND BARTLETT, P. L. 2007. The Rademacher complexity of co-regularized kernel classes. In *AISTATS'07*. San Juan, USA.

ROSENBLATT, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review 65,* 6, 386–408.

ROTH, D. AND SMALL, K. 2006a. Active learning with perceptron for structured output. In *ICML Workshop on Learning in Structured Output Spaces*. Pittsburgh, USA.

ROTH, D. AND SMALL, K. 2006b. Margin-based active learning for structured output spaces. In *ECML'06*. Berlin, Germany.

SATPAL, S. AND SARAWAGI, S. 2007. Domain adaptation of conditional probability models via feature subsetting. In *PKDD'07*. Warsaw, Poland.

SCHEFFER, T., DECOMAIN, C., AND WROBEL, S. 2001. Active hidden markov models for information extraction. In *IDA'01*. Cascais, Portugal.

SEEGER, M. 2001. Learning with labeled and unlabeled data. In *Technical Report*.

SERVEDIO, R. A. AND LONG, P. 2011. Algorithms and hardness results for parallel large margin learning. In *NIPS'11*. Granada, Spain.

SETTLES, B. 2009. Active learning literature survey. In *Computer Sciences Technical Report 1648*. University of Wisconsin-Madison.

SETTLES, B. AND CRAVEN, M. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP'08*. Honolulu, USA.

SETTLES, B., CRAVEN, M., AND RAY, S. 2008. Multiple-instance active learning. In *NIPS'08*. Vancouver, Canada.

SEUNG, H. S., OPPER, M., AND SOMPOLINSKY, H. 1992. Query by committee. In *COLT'92*. Pittsburgh, USA.

SHAHSHAHANI, B. AND LANDGREBE, D. 1994. The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing 32,* 5, 1087 –1095.

SHENG, V. S., PROVOST, F., AND IPEIROTIS, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD'08*. Las Vegas, USA.

SHI, X., FAN, W., AND REN, J. 2008. Actively transfer domain knowledge. In *ECML/PKDD'08*. Antwerp, Belgium.

SHIMODAIRA, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference 90,* 2, 227–244.

SINDHWANI, V., NIYOGI, P., AND BELKIN, M. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *ICML Workshop on Learning with Multiple Views*. Bonn, Germany.

SINDHWANI, V. AND ROSENBERG, D. S. 2008. An RKHS for multi-view learning and manifold co-regularization. In *ICML'08*. Helsinki, Finland.

SINGH, S. P., KEARNS, M. J., AND MANSOUR, Y. 2000. Nash convergence of gradient dynamics in general-sum games. In *UAI'00*. Stanford, USA.

SMOLA, A. J. AND KONDOR, R. I. 2003. Kernels and regularization on graphs. In *COLT'03*. Washington, USA.

SONNTAG, D. 2004. Distributed nlp and machine learning for question answering grid. In *Proceedings of the workshop on Semantic Intelligent Middleware for the Web and the Grid at ECAI*. Valencia, Spain.

SRIDHARAN, K. AND KAKADE, S. M. 2008. An information theoretic framework for multi-view learning. In *COLT'08*. Helsinki, Finland.

STORKEY, A. J. AND SUGIYAMA, M. 2006. Mixture regression for covariate shift. In *NIPS'06*. Vancouver, Canada.

SUGIYAMA, M., NAKAJIMA, S., KASHIMA, H., VON BÜNAU, P., AND KAWANABE, M. 2007. Direct importance estimation with model selection and its application to covariate shift adaptation. In *NIPS'07*. Vancouver, Canada.

SZUMMER, M. AND JAAKKOLA, T. 2001. Partially labeled classification with markov random walks. In *NIPS '01*. MIT Press, Cambridge, MA.

TEO, C. H., VISHWANTHAN, S., SMOLA, A. J., AND LE, Q. V. 2010. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research 11*, 311–365.

THRUN, S. AND O'SULLIVAN, J. 1996. Discovering structure in multiple learning tasks: The tc algorithm. In *ICML'96*. Bari, Italy, 489–497.

TOMANEK, K., WERMTER, J., AND HAHN, U. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *EMNLP/CoNLL'07*. 486–495.

TSUDA, K., RÄTSCH, G., AND WARMUTH, M. K. 2005. Matrix exponentiated gradient updates for on-line learning and bregman projections. *Journal of Machine Learning Research 6*, 995–1018.

TUMER, K. AND GHOSH, J. 1996. Estimating the bayes error rate through classifier combining. In *ICPR'96*. Vienna, Austria.

TUR, G. 2009. Co-adaptation: Adaptive co-training for semi-supervised learning. In *ICASSP'09*. Taipei, Taiwan.

VIJAYANARASIMHAN, S. AND GRAUMAN, K. 2008. Multi-level active prediction of useful image annotations for recognition. In *NIPS'08*. Vancouver, Canada.

VIJAYANARASIMHAN, S. AND GRAUMAN, K. 2009. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *CVPR'09*.

VITTER, J. S. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software 11*, 37–57.

VLACHOS, A. 2008. A stopping criterion for active learning. *Computer Speech and Language 22,* 3, 295–312.

VOVK, V. G. 1990. Aggregating strategies. In *COLT'90*. Rochester, USA.

WANG, H., HUANG, H., NIE, F., AND DING, C. 2011. Cross-language web page classification via dual knowledge transfer using nonnegative matrix tri-factorization. In *SIGIR'11*. Beijing, China.

WANG, J. 2007. On transductive support vector machines. In *Prediction and Discovery*. American Mathematical Society.

WANG, Z., SONG, Y., AND ZHANG, C. 2009. Knowledge transfer on hybrid graph. In *IJCAI'09*. Pasadena, USA.

XING, D., DAI, W., XUE, G.-R., AND YU, Y. 2007. Bridged refinement for transfer learning. In *PKDD'07*. Warsaw, Poland.

XUE, Y., DUNSON, D., AND CARIN, L. 2007a. The matrix stick-breaking process for flexible multi-task learning. In *ICML'07*. New York, USA.

XUE, Y., LIAO, X., CARIN, L., AND KRISHNAPURAM, B. 2007b. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research 8*, 35–63.

YAROWSKY, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL'95*. Cambridge, Massachusetts.

YE, J., CHOW, J.-H., CHEN, J., AND ZHENG, Z. 2009. Stochastic gradient boosted distributed decision trees. In *CIKM'09*. Hong Kong.

ZADROZNY, B. 2004. Learning and evaluating classifiers under sample selection bias. In *ICML'04*. Banff, Canada.

ZAHARIA, M., CHOWDHURY, M., FRANKLIN, M. J., SHENKER, S., AND STOICA, I. 2010. Spark: Cluster computing with working sets. In *HotCloud'10*. Boston, USA.

ZHANG, J., GHAHRAMANI, Z., AND YANG, Y. 2005. Learning multiple related tasks using latent independent component analysis. In *NIPS'05*. Vancouver, Canada.

ZHANG, Y. AND SCHNEIDER, J. 2010. Learning multiple tasks with a sparse matrix-normal penalty. In *NIPS'10*. Vancouver, Canada.

ZHANG, Y. AND YEUNG, D.-Y. 2010. A convex formulation for learning task relationships in multi-task learning. In *UAI'10*. Catalina, USA.

ZHAO, W., MA, H., AND HE, Q. 2009. Parallel k-means clustering based on mapreduce. In *Cloud Computing'09*. Beijing, China.

ZHENG, Z. AND PADMANABHAN, B. 2002. On active learning for data acquisition. In *ICDM'02*. Washington, USA.

ZHONG, E., FAN, W., PENG, J., ZHANG, K., REN, J., TURAGA, D., AND VERSCHEURE, O. 2009. Cross domain distribution adaptation via kernel mapping. In *KDD'09*. Paris, France.

ZHU, J. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *ACL'07*. Prague, Czech Republic.

ZHU, X., LAFFERTY, J., AND GHAHRAMANI, Z. 2003. Semi-supervised learning: From gaussian fields to gaussian processes. Tech. rep., School of CS, CMU.

ZHUANG, F., LUO, P., SHEN, Z., HE, Q., XIONG, Y., SHI, Z., AND XIONG, H. 2010. Collaborative dual-plsa: Mining distinction and commonality across multiple domains for text classification. In *CIKM'10*. Toronto, Canada.

ZHUANG, F., LUO, P., XIONG, H., HE, Q., XIONG, Y., AND SHI, Z. 2011. Exploiting associations between word clusters and document classes for cross-domain text categorization. *Journal Statistical Analysis and Data Mining 4,* 1, 100–114.

ZINKEVICH, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *ICML'03*. Washington, USA.

ZINKEVICH, M., WEIMER, M., SMOLA, A. J., AND LI, L. 2010. Parallelized stochastic gradient descent. In *NIPS'10*. Vancouver, Canada.

## DISSEMINATION OF THIS WORK

- **Co-regularization Based Semi-supervised Domain Adaptation**.
  Hal Daumé III, Abhishek Kumar, Avishek Saha.
  In Proceedings of Advances of Neural Processing Systems (NIPS), December 2010.

- **Online Learning of Multiple Tasks and Their Relationships**.
  Avishek Saha, Piyush Rai, Hal Daumé III, Suresh Venkatasubramanian.
  In Proceedings of Artificial Intelligence and Statistics (AISTATS), April 2011.

- **Active Supervised Domain Adaptation**.
  Avishek Saha, Piyush Rai, Hal Daumé III, Suresh Venkatasubramanian, Scott L. DuVall.
  In Proceedings of European Conference on Machine Learning (ECML), September 2011.

- **Protocols for Learning Classifiers on Distributed Data**.
  Hal Daumé III, Jeff M. Phillips, Avishek Saha, Suresh Venkatasubramanian.
  In Proceedings of Artificial Intelligence and Statistics (AISTATS), April 2012.

- **Efficient Protocols for Distributed Classification and Optimization**.
  Hal Daumé III, Jeff M. Phillips, Avishek Saha, Suresh Venkatasubramanian.
  In Algorithmic Learning Theory (ALT), October 2012.