# An Empirical Study on Comparison between Transfer Learning and Semi-supervised Learning

by

Ao Cheng

**Bachelor of Computer Science,**
**Huazhong University of Sci. & Tech., 2011**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF**

**Master of Computer Science**

In the Graduate Academic Unit of Computer Science

| | |
|---|---|
| Supervisor(s): | Huajie Zhang, Ph.D.,Computer Science |
| Examining Board: | Wei Song, ph.D., Computer Science, Chair |
| | Weichang Du, ph.D., Computer Science |
| | Guohua Yan, ph.D., Mathematics and Statistics |

This thesis is accepted by the

Dean of Graduate Studies

**THE UNIVERSITY OF NEW BRUNSWICK**

**June, 2013 of submission to Graduate School**

# Abstract

Transfer learning and semi-supervised learning attract considerable attention since the traditional machine learning methods yield insufficient performance in many practical applications with scarce labeled data. In such cases, knowledge transfer from a related domain or information extraction of unlabeled data, if done properly, would significantly upgrade the classifier by avoiding costly labeling expense. These two branches of machine learning try to use auxiliary data to make up for the shortage of labeled instances. In this study, a set of experiments are conducted on several typical algorithms for both transfer learning and semi-supervised learning to test whether these auxiliary data should be beneficial. The empirical study shows that these auxiliary instances may not be permanently helpful comparing to the traditional learning methods. However, when a special situation with an extremely small number of labeled instances arises, the auxiliary data would improve the performance significantly. The internal characteristics influencing the performance in each branch is also explored in this study.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| | |
|---|---|
| $k$NN | $k$ nearest neighbour |
| NB | Naive Bayes |
| QP | quadratic problem |
| SMO | sequential minimal optimization |
| SSL | semi-supervised learning |
| SVM | support vector machine |
| TL | transfer learning |

# Chapter 1

# Introduction

## 1.1 Motivation

Nowadays, the advent of the digital age has promoted the spread of information. From PCs to mobile devices, from Intranet to Internet, the collections of datasets become so large and complex that it becomes difficult to process using traditional data processing applications.

In practical application, either the process of labelling is costly or time-consuming, or sometimes the data updates so fast that easily get outdated. At this point, utilizing traditional machine learning methods with sufficient labeled data to build a new classifier is no longer cost-effective. For example, the indoor WiFi location problem, which aims to detect the location of a client device by using a large collection of WiFi signal data. The signal strength values are perhaps the results of a function of complex factors such

as time and space. Furthermore, since the signal distribution easily gets changed, such process needs to rebuild the model. Another example is online recommendation system. The system makes prediction on anonymous users' rating based on both their previous rating records and certain other users' high rating recommended items. The enormous number of goods and tremendous evaluation quotes make it impossible for customers to browse every item. With the exponential growth in available information, people get always overwhelmed by a large amount of data. However, the method of effectively extracting effective knowledge and utilizing them for achievement has become the crux of the matter.

For many practical problems such as those mentioned above, traditional machine learning methods are not applicable for these cases. One reason is the lack of label information provided. On the other hand, it is impossible for human labour to label such massive data. Thus, training data set would be insufficient to build a competent classifier. Most recent researches in this area focus on exploring different machine learning methods for certain proper conditions, especially on supervised learning and unsupervised learning algorithms. So far, no detailed analysis or systematic comparison of performance using the auxiliary data has been studied.

Therefore, our research provides references for these applications that weigh the auxiliary data in training process. In response to this challenge mentioned above, semi-supervised learning and transfer learning attract some researchers to investigate. Although these two branches of machine

learning build the classifier from unrelated different aspects, they both try to maximize the information extraction from auxiliary data, so as to build a better classifier, which saves a significant amount of efforts on labeling. Comparison of standard algorithms with these two branches could help us to explore the merits and faults.

The primary goal of the thesis is to analyze the performance of auxiliary data by comparing typical algorithms in transfer learning and semi-supervised learning, which can help us to get further access to the role of auxiliary data in different types and make improvement for performance of classifiers in data mining.

## 1.2 Thesis Overview

In this thesis, we first review transfer learning and semi-supervised learning algorithms including classification, learning strategies, and problem settings. These two branches of machine learning have already achieved significant success where many representative algorithms are proposed and applied to practice. In order to analyse and compare the performance of these methods, the experiment is conducted on 19 UCI datasets [1] across domains of semi-supervised learning and transfer learning. The empirical study is processed on the WEKA [2] software, which is a popular suite of machine learning software written in Java, developed at the University of Waikato. It is noteworthy that

[1] The data package is available from http://www.cs.waikato.ac.nz/ml/weka/datasets.html

transfer learning and semi-supervised learning undertake different problems, so they have similarities in their methodologies. The applicability for these methods can be measured by predictive accuracy of the test set. From the experimental results, the performance of listed approaches may fluctuate when more auxiliary instances are utilized. The results also show that these semi-supervised learning methods generally do not obtain better performance than classifiers learned only from the labeled data. Meanwhile, transfer learning methods only perform well when the source instances are related to the target data set, which usually produces positive transfer. The goal of transfer learning is to improve learning in the target task by leveraging knowledge from the source task. We also probe the similarity between the source data set and target data set, the key factor of the knowledge transfer performance.

Currently, both semi-supervised learning and transfer learning techniques have been applied successfully in many real world applications. In this thesis, we will give a scan to these applications, as well as the challenges they faced.

## 1.3 Organization

The remainder of this thesis is organized as follows. In Chapter 2, an overview of semi-supervised learning and transfer learning is provided in particular. With the background of these two branches in machine learning, we present the features and standard algorithms in detail, and then give a

brief introduction to the related work.

In Chapter 3, the experimental design of this thesis is described. The experiment is conducted for the purpose of investigating the performance of semi-supervised learning algorithms and transfer learning algorithms. The approach is to compare several widely used semi-supervised learning algorithms, as well as transfer learning. Meanwhile, we will give a quick look at the experiment platform WEKA. In Chapter 4, specific empirical results and analysis will be presented. In addition, the applications of these two machine learning branches are introduced in this section. Finally, Chapter 5 summarizes our work, draws conclusions of this study and makes recommendations for future research.

# Chapter 2

# Background

Machine Learning and Data mining methodologies have been explored and developed for the past decades, with achievement in many knowledge engineering areas including classification, regression, and clustering (e.g. [33], [34]). However, traditional data mining and machine learning research gives a basic assumption: the training and test data are under the same feature space and distribution, as well as sufficient training data are available. With the arrival of information age, in the practical applications of machine learning, labeled data could be scarce and expensive, or impossible to collect the needed training data and build models while auxiliary data is abundant and relatively easy to obtain, such as unlabeled data in the same domain and labeled data from other similar domains. It would be nice to improve the utilization of auxiliary data in order to reduce the need and effort of labeling the training data.

This chapter introduces background technologies and related work about transfer learning and semi-supervised learning. It is divided into two sections. In the first section, we give a general overview and define some notations about transfer learning including ‘‘what to transfer’’, and ‘‘how to transfer’’. Section 2.2 attempts to answer the following questions: What is semi-supervised learning, How can we learn from unlabeled data, and what are standard algorithms in semi-supervised learning?

## 2.1 Transfer Learning

Human learners appear to have inherent ways to transfer knowledge between tasks. That is, we recognize and apply relevant knowledge from previous learning experiences when we encounter a new task [27]. For example, people who played chess game once generally feel easier to learn checkers than the ones who have never played board game. The more related a new task is to the previous experience, the easier for the player to grasp. Common machine learning algorithms address isolated tasks while transfer learning attempts to transfer knowledge learned in one or more source tasks and use it to build a better classifier in a related target task.

### 2.1.1 Introduction

As we mentioned before, a major assumption in many machine learning and data mining algorithms is that the training and future data must be in

the identical feature space and distribution. Transfer learning, in contrast, permits the features, distributions, and tasks in training and testing to be various.

Due to the assumption of identical distribution, most learning methods fail to work effectively when the distribution changes. At this point, it is necessary for most statistical models to be rebuilt from re-collection of a large amount of labeled data on purpose to fit the new distribution. This motivates the branch of transfer learning, which transfers the knowledge between task domains.

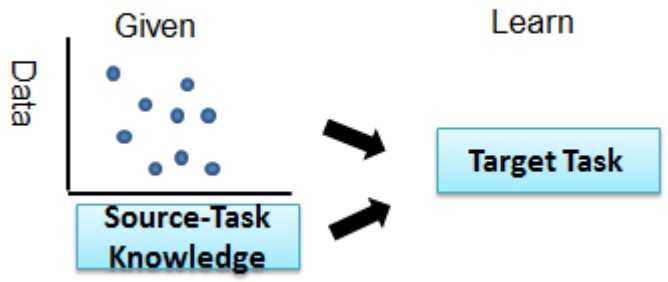


Figure 2.1: Transfer Learning

In this thesis, we use ‘‘domain’’ to describe a dataset and ‘‘task’’ to represent the classifier for predictive model. A *domain* $D$ consists of two elements: a feature space $\chi$ and a marginal probability distribution $P(X)$, where $X = \{x_1, .., x_n\} \in \chi$. Each instance from the domain is expressed as $\{x_1, .., x_n\}$, where $x_i$ is the $i - th$ term in feature spaces, then $\chi$ is the space

of all term vectors, and $X$ is a particular instance. A *task* is also constituted by two elements: a label space $Y$ and a function for predictive model $f(\otimes)$. Given one source domain $D_S$, learning task $T_S$, one target domain $D_T$ and learning task $T_T$, we can give a definition of transfer learning.

**Definition 2.1.1.** *Transfer Learning aims to improve learning of the target predictive function $f_T(\otimes)$ in $D_T$ using knowledge in $D_S$ and $T_S$, where $D_S \neq D_T$, or $T_S \neq T_T$ [25].*

In the definition, $D_S \neq D_T$ means either $\chi_S \neq \chi_T$ or $P_S(X) \neq P_T(X)$. In addition, $T_S \neq T_T$ implies either $Y_S \neq Y_T$ or $f_S(\otimes) \neq f_T(\otimes)$.

### 2.1.2 Classification of Transfer Learning

In transfer learning, ''what to transfer'' and ''when to transfer'' determine the strategy of transfer learning. The former raises a question which part of knowledge is available for transferring across domains, and the latter corresponds to in which situations the process should be finished.

According to the above definition 2.1.1, we can divide transfer learning into three categorizations, *inductive transfer learning, transductive transfer learning,* and *unsupervised transfer learning* in accordance with the different situations (refer to Table 2.1).

Table 2.1: Transfer Learning Settings

| **Settings** | **Source and Target Domains** | **Source and Target Tasks** |
|---|---|---|
| Inductive Transfer Learning | the same | different but related |
| Transductive Transfer Learning | different but related | the same |
| Unsupervised Transfer Learning | different but related | different but related |

#### 2.1.2.1 Inductive Transfer Learning

Inductive learning is concerned with the estimation of a function (a model) based on data from the whole problem space and using this model to predict output values for a new input vector, which can be any point in this space [22]. Transfer in inductive learning works by allowing source-task knowledge to affect the target inductive bias of task [27]. The setting of inductive transfer learning is that tasks between source domain and target domain are different. In inductive transfer learning methods, the target task inductive bias is adjusted based on the source-task knowledge. Although the data in source domain can be either labeled or unlabeled, most transfer learning methods in this setting are concentrated on the situation in which labeled data of source domain is available.

Inductive transfer learning approaches can be summarized into the following categorizations based on "what to transfer".

- Knowledge of Instance Transfer: transfer knowledge from instances is intuitively appealing. The basic assumption is some data from the source domain can be reused for the target task. Thus, approaches of this kind field focus on strategies to select the useful data from source domain.

- Knowledge of Feature-representations Transfer: the objective is to minimize the domain divergence by using strategies to pick ''good'' feature representations. That is to say, with learning new feature representation , the new target task should outperform the original one.

- Knowledge of Parameters Transfer: this case gives the assumption that some parameters or prior distributions are shared by source tasks and target tasks, so that knowledge transfer process can be implemented by exploring shared parameters or priors.

- Relational Knowledge Transfer: this case gives assumptions that some relationship among the data from source domain and target domain is similar. That is to say, it means the target attempts to apply the relationship among the data in source domain to ones in target domain.

#### 2.1.2.2 Transductive Transfer Learning

Transductive learning concerns the estimation of a function in only a single point of the space, regardless of its dimensionality [16], [28]. Hence, in the traditional machine learning setting, all test data can be seen during

the training process and the predictive model can make prediction to the future data which are unseen at the training time. However, the transductive transfer learning only concerns the source task $T_S$ and target task $T_T$ should be identical with some unlabeled data in the target domain.

Arnold et al. [1] first put forward the term *transductive transfer learning*. They studied the more challenging case of unsupervised transductive transfer learning which no labeled data in the target domain are available. S. Pan [25] gave a definition of transductive transfer learning which covers the previous work.

**Definition 2.1.2.** *Given a source domain $D_S$ and a corresponding learning task $T_S$, a target domain $D_T$ and a corresponding learning task $T_T$, transductive transfer learning targets on improving the learning of the target predictive function $f_T(\otimes)$ in $D_T$ using the knowledge in $D_S$ and $T_S$, where $D_S \neq D_T$ and $T_S \neq T_T$. In addition, some unlabeled target-domain data must be available at training time [25].*

Transductive transfer learning approaches can be also summarized into the following two categorizations based on ''what to transfer''.

- Knowledge of Instance Transfer: instance transfer methods of transductive transfer learning are generally based on importance sampling.
- Knowledge of Feature Representations Transfer: approaches are usually designed for dimensionality reduction, so as to reduce the divergence of distributions between domains for transductive transfer learning.

#### 2.1.2.3 Unsupervised Transfer Learning

Unsupervised transfer learning means to improve the target task by using knowledge in the source domain, where source task and target task are different and no labeled data are available at the training time. Little research work has been done until now in this area. Dai et al. [9] once proposed the *Self-taught clustering* to transfer clustering and Wang et al. presented *Transferred discriminative analysis* for transfer dimensionality reduction problems [30].

### 2.1.3 Algorithms of Transfer Learning

Although transfer learning algorithms are designed to address multiple tasks, transfer approaches tend to be highly dependent on the traditional machine learning algorithms. That is to say, many transfer methods can be viewed as extensions of traditional machine learning algorithms. Generally, most of such algorithms are focusing on instance transfer and feature-representation transfer.

In instance transfer, training instances in the source domain are re-weighted according to their effect on the target task [8]. It assumes that certain parts of the data in the source domain can be reused together with the labeled data in the target domain.

- TrAdaBoost: It is an extension of the Adaboost algorithm, proposed by Dai et al. [8]. The basic idea is to iteratively re-weight the source domain

data to improve the effect of the source data domain. To be more specific, in each iteration round, if a training instance is mistakenly predicted, this instance may be likely to be considered as ''bad'' data to the target task. Then the algorithm decreases its weight. TrAdaBoost follows the same strategy of AdaBoost to update the weights of incorrectly classified instances in target domain, but adopts its own strategy to handle with the weight of instances from source domain. Here we review some research work about this area.

- Migratory-Logistic: Liao et al. [19] introduced an active learning approach for selecting unlabeled instances in the target domain with the help of data from a different domain. An auxiliary variable is introduced as a byproduct for each instance of source domain to reflect its mismatch with the target domain. This approach is capable of learning in the presence of mismatch between the source domain and target domain.

- Explore auxiliary training data with $kNN$: Wu et al. [32] described a methodology for integrating auxiliary training data into $kNN$ as well as Support Vector Machine methods. In this paper, the training data play two separate roles, define the objective function and the hypothesis, so it tries to minimize a weighted sum of two loss functions, one for target domain and the other for source domain.

For feature-representations transfer, it tries to learn a ''good'' feature

representation for target domain, as well as reduce domain divergence and classification or regression model error. How to find ‘‘good’’ feature representations varies from the different types of source domains.

- Structural Correspondence Learning: J. Blizer et al. [15] introduced structural correspondence learning to adapt existing models from a resource-rich source domain to a resource-poor target domain for NLP. The structural correspondence learning supposes that both source and target domains have plenty of unlabeled data, but labeled data is available only in the source domain.

- Co-Clustering Based Classification: Dai et al. [7] proposed a co-clustering based classification (CoCC) algorithm to learn form the in-domain and apply it to the out-domain. Co-clustering is used as a bridge to propagate the class structure and knowledge from the in-domain to the out-of-domain. CoCC formulated the problem under an information-theoretic scheme, and designed an objective function to minimize the loss in mutual information before and after co-clustering based categorization.

- Lee et al. [18] presented a probabilistic approach for learning an informed prior about feature relevance from an ensemble of related tasks. The convex optimization algorithm for simultaneously learning the meta-priors and feature weights from an ensemble of related prediction tasks which share a similar relevance structure.

We know that transferring knowledge of feature-representation approaches rely on the condition of source domain data. If ample labeled data exist in source domain, supervised learning methods can be used to construct the feature representation. While there are no labeled data available in source domain, then unsupervised learning methods can be induced to set up the feature representations.

### 2.1.4 Negative Transfer Learning

When the data of source domain decrease performance for target task, then negative transfer has occurred. How to produce positive transfer among similar domains is a very important issue in developing transfer methods. During the training process with source domain data, if the class distribution of reused source domain data is very close to the target domain, the performance of the target task can be significantly improved through transfer. In contrast, if the selected data from source domain are not sufficiently related, then the performance with many approaches may decrease.

Eaton et al. [11] presented a novel graph-based method for knowledge transfer. Transfer to a new problem proceeds by mapping the problem into the graph, then learning a function on this graph that automatically determines parameters to transfer to the new learning task. Rosenstein et al. [24] empirically demonstrated that transfer learning often helps, but sometimes hurts performance if the sources of data are too much dissimilar. The Hierarchical Naive Bayes algorithm was designed to avoid negative

transfer in this paper. However, many transfer learning approaches are not able to guarantee the improvement of basic learner.

## 2.2 Semi-supervised Learning

Semi-supervised learning is a class of machine learning techniques that utilizes both labeled and unlabeled data. While traditional classifiers use only labeled data for training purposes, semi-supervised learning addresses this problem by using large amount of unlabeled data, along with the labeled data, to build better classifier.

Semi-supervised Learning methods arose when learning classifiers on a small set of labeled training data may not produce good performance. A number of semi-supervised learning methods have been proposed such as self-training, co-training, semi-supervised support vector machines, graph-based methods [5], [37], [14]. The general idea is to select some unlabeled examples according to a given selection criterion and then label them to enlarge the training dataset. In recent years, several semi-supervised learning approaches are explored for practical applications in different areas, such as information retrieval, text classification, and bioinformatics.

### 2.2.1 Introduction

According to the training data whether has been hand-labeled or not in the process of generating the classifier's output, there are three types of

learning: supervised learning can listed out, semi-supervised learning, and unsupervised learning.

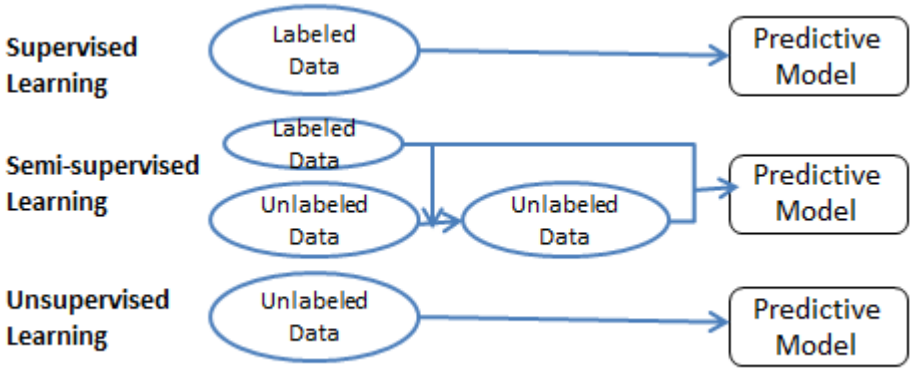


Figure 2.2: Supervised Learning, Semi-supervised Learning, and Unsupervised Learning

Supervised learning assumes that a set of training data has been provided, while unsupervised learning has no labeled data in training process. A combination of both labeled and unlabeled data results in semi-supervised learning. Figure 2.2 demonstrates the difference among these types of learning algorithms.

For a better understanding of semi-supervised learning, it is essential to take a look into supervised and unsupervised learning.

#### 2.2.1.1 Supervised Learning

In supervised learning, expert work or human labour cost are necessary for labelling the set of instances for training process. Theoretically, the more

labeled data, the more precise classifier will get. Hence, it is necessary to acquire a certain amount of hand labeled instances to reach a desired quality for generating the predictive model, which is time-consuming to a certain degree. Let $\mathbf{X} = \{X_1, ..., X_n\}$ and $\mathbf{Y} = \{y_1, ..., y_m\}$ be a set of instances and the labels set of the instances $X_i$, respectively. The objective of supervised learning is to map $\mathbf{X}$ to $\mathbf{Y}$ from a training set consisted of pairs $(X_i, y_i)$, where $X_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$.

The algorithms of supervised learning are summarized into two categories: generative algorithms and discriminative algorithms.

- Generative Model: This kind model is built with randomly generating observable data typically given some hidden parameters. It assumes that $P(X, y) = P(X|y)P(y)$, where $P(X, y)$ is the joint distribution of the data over observations and label sequences.

- Discriminative Model: It models the dependency of $y$ on observation of $X$, which concentrates on estimating $P(y|X)$ instead of how the instances are generated.

Supervised learning applications are widely used such as decision trees, naive Bayes, and support vector machines for their easy evaluation and acceptance to human. But the difficulty of collecting supervision or labels limits supervised learning in practice.

#### 2.2.1.2 Unsupervised Learning

On the other hand, unsupervised learning attempts to find out similar patterns in data to determine the output with no labeled data in training process. Since no clues on which shot types are, the system will define different classes clustering similar data based on the visual content. For example, around $10^6$ photoreceptors exist in each eye. Their activities are to constantly change the position with respect to the visual world. The brain indicates what objects are presenting in the world with the information that photoreceptors provide, such as what the ambient illumination condition is, and how they are displayed [10]. Let $\mathbf{X} = \{X_1, ..., X_n\}$ be a set of instances, which are drawn *iid*(independently and identically distributed) from a common distribution $\mathbf{D}$ with the target to explore the structure of data set $\mathbf{X}$. It is difficult to evaluate the accuracy of the generated model as no labeling information is given. Clustering usually take positive effects on in unsupervised learning. Consider the case in which the inputs and the photoreceptor activities created by various images of an object like a cat or a dog. These particular inputs form two clusters in the space of all possible activities, with much lower degrees of variation than $10^6$ [10].

Unsupervised learning is one of key fundamental problems of machine learning and statistics, including problems as diverse as clustering, dimensionality reduction, system identification, and grammar induction [17].

#### 2.2.1.3 Semi-supervised Learning

Semi-supervised learning might be viewed as a compromise between supervised learning and unsupervised learning. The semi-supervised learning algorithms are supplied with unlabeled data and some supervision information, to be more specific, it is a supervised method that avoids labelling a large number of instances. As we can see in Figure 2.2, first semi-supervised learning classifies the unlabeled data by using some of the labeled data. And then, this automatic labeled data is also used by the training process.

Unlike the unsupervised learning setting, some label information is provided in semi-supervised learning. Under some certain circumstances, unlabeled data can be learned with the help of labeled information. Because semi-supervised learning requires less human effort and gives higher accuracy, it attracts many attentions in both theory and practice. The standard setting and specific approaches for semi-supervised learning will be discussed in the following section.

### 2.2.2 Algorithms of Semi-supervised Learning

Do unlabeled data always help? Is semi-supervised learning meaningful? Actually, bad matching of problem structure with model assumption can downgrade performance. In order to make semi-supervised learning reliable, we have several prerequisites when these approaches are used.

The first one is that the information of unlabeled data should be relevant

to the classification. Although not all methods are probabilistic, it is easier to look at methods by mathematical formulation: unlabeled data by $P(X)$ is useful in the hypotheses by $P(y|X)$. The other key of assumption underlying the field is the smoothness: if two data points are linked by a path of high density, then their outputs are likely to be closed [29]. Thus if they are separated by a low-density region, then the outputs need not to be close.

Currently, a number of semi-supervised learning methods are proposed. Meanwhile, some frequently used methods include self-training, co-training, EM with generative mixture models, graph-based methods, and so on. In this section, some popular semi-supervised learning categorizations will be introduced.

#### 2.2.2.1 Generative Models

Generative Models are selected to apply for semi-supervised learning in the early stage. It assumes a model $P(X, y) = P(X|y)P(y)$ where $P(y)$ is an identifiable mixture distribution such as Gaussian mixture models [37]. That is, parameter $P(X, y)$ captures the labeled data and $P(X)$ represents the probability of the unlabeled data. Ideally, one labeled data per component is enough to determine the mixture distribution. But when the assumption can not be held, then the unlabeled data is not helping improve the performance. Nigam et al. [21] applied the Expectation-Maximization (EM) algorithm for text classification. The algorithm combines EM and a Naive Bayes classifier. It firstly trains a predictive model only by the labeled data, and then uses

the model to classify the unlabeled data. Then it trains a new predictive model by the old labeled data as well as the new labeled data. Fujino et al. extended generative mixture models by including a biased correction term and discriminative training using maximum entropy principle [12].

It is necessary to notice some details when using the generative models. First, the mixture model should be identifiable. If the model is identifiable, model parameters can be estimated on the basis of unlabeled data alone except for the class permutation, which can be readily inferred from a few complete samples [6]. Then we need to keep model correctness in mind. If the mixture model assumption is inaccurate, unlabeled data may actually hurt performance.

#### 2.2.2.2 Graph-based Methods

Graph-based semi-supervised methods are based on the idea of constructing a graph connecting similar data points. The underlying concept of this kind of algorithms is that a graph where the vertices are labeled and unlabeled instances, and the edgers between the nodes represent the similarities between the instances.

Blum et al. considered an algorithm based on finding minimum cuts in graphs, that uses pairwise relationships among the examples in order to learn from both labeled and unlabeled data [3]. The objective is to partition the graph in a way that minimizes the number of edges that are given different labels. A variety of other methods can be viewed as a regularized function

estimation over the graph. These methods intend to optimize the trade-off between fitting a prediction function and a regularization term which encourages smoothness on the whole graph [20]. The basic idea of these methods are quite similar, only differ in the specific choice of the function and regularization term. For instances, computing the marginal probabilities of the discrete Markov random Fields [26], uses Gaussian random fields and harmonic function methods to reduce the complexity [38], Local and Global Consistency [36], and so on.

#### 2.2.2.3 Self-training

Self-training is another commonly used approach of semi-supervised learning. It is a single-view bootstrapping semi-supervised learning algorithm with an underlying classifier and a selection strategy for unlabeled instances. Naive Bayes is often chosen for the basic classifier with the small amount of labeled data. In addition, the classifier is then used for labelling the unlabeled data. A certain selection metric is used for ranking the labeled instances in terms of their predicted class labels. These unlabeled instances with higher ranking are selected to enlarge the training set. The classifier is re-trained based on the new training set and the procedure repeated.

Self-training is widely used in natural language processing tasks, such as word sense disambiguation, subjective nouns identification, machine translation, etc.

#### 2.2.2.4 Co-training

Co-training is a multiple-view bootstrapping semi-supervised learning technique, first proposed by Blum and Mitchell [4]. Two classifiers are separated trained on the two sub-feature datasets, and predictions of each classifier on unlabeled data are used to expand the training set of the other classifier. These sub-feature datasets are two attribute sets that meet the following two assumptions: (I) each attribute set is sufficient to train a good classifier; (II) attribute sets must satisfy the conditional independence assumption given the class label [37]. Co-training strongly relies on the feature split assumption. However, it is impossible that there is always natural feature split. Then does co-training still help? Yes, some researches have showed that artificially split features into two sets are still useful.

# Chapter 3

# Methodology

As mentioned in the previous chapters, transfer learning and semi-supervised learning focus on problems from different views. In the field of machine learning, the fundamental motivation for both transfer learning and semi-supervised learning is improving the performance of classifier by leveraging the knowledge from the auxiliary data when it comes to the situation that only a small amount of labeled data are available. These approaches from transfer learning or semi-supervised learning may work well under some certain situations and assumptions. However, no related work has been conducted to answer the questions of ‘‘which one is better?’’ or ‘‘could they really help in every circumstance of scarce labeled data by using auxiliary data?’’ Thus we are not clear enough about the results on comparing approaches of semi-supervised learning and transfer learning under the same condition.

Motivated by above issues, in this thesis, we conduct an empirical study

on comparison between transfer learning and semi-supervised learning. The experiment is performed on some UCI datasets at WEKA platform. Specific settings and application framework will be discussed in next chapter .

In this empirical study, two typical algorithms are selected from each branch. In transfer learning, we use TrAdaBoost algorithm for both single source and multiple sources in instance transfer learning. On the other hand, for semi-supervised learning, self-training and co-training are pick up as representatives since they are commonly used techniques. Besides, learning curves are drawn in order to analyse the effect of the amount of labeled and unlabeled data on classification performance.

In this chapter, we have a general view of the methodology we utilized in the empirical study. In the next chapter, the evaluation and analysis of experiment results will be primary focused.

## 3.1 Learning Algorithms

Standard self-training, co-training, TrAdaBoost, MultiSourceTrAdaBoost will be used in our experiments, as well as the base learner Naive Bayes and Support Vector Machine(SVM). Since we have roughly introduced them in Chapter 2, we will pay more attention on the implementation part of algorithms in this section. Naive Bayes and SVM are chosen as base learner, since Naive Bayes performs well in text mining and SVM behaves satisfying with support of substantial theories.

### 3.1.1 Base Learner

#### 3.1.1.1 Naive Bayes

Naive Bayes is a highly practical Bayesian learning method with one independence assumption, which has been shown to be comparable to neural network. It is widely used because of its simplicity and efficiency, particularly for the large dataset.

**Notation:** Let an instance $X$ be described by the tuple of attribute values $< a_1, a_2, ..., a_n >$, $Y$ is a class label set. The Bayesian approaches assign the most probable target value $y_{MAP}$ to classify the new instance which is described by $< a_1, a_2, ..., a_n >$.

$$y_{MAP} = \arg\max_{y_j \in Y} P(y_j | a_1, a_2, ..., a_n) \tag{3.1}$$

This expression can be rewritten as

$$\begin{aligned} y_{MAP} &= \arg\max_{y_j \in Y} \frac{P(a_1, a_2, ..., a_n | y_j) P(y_j)}{P(a_1, a_2, ..., a_n)} \\ &= \arg\max_{y_j \in Y} P(a_1, a_2, ..., a_n | y_j) P(y_j) \end{aligned} \tag{3.2}$$

The Naive Bayes classifier assumes that these attribute values are conditionally independent to each other given the target value, which can be

formulated as followed:

$$P(a_1, a_2, ..., a_n|y_j) = \prod_i P(a_i|y_j) \tag{3.3}$$

When substituting Equation 3.3 into Equation 3.2, we can get the Naive Bayes classifier.

**Naive Bayes classifier**:

$$y_{NB} = \arg\max_{y_j \in Y} P(y_j) \prod_i P(a_i|y_j) \tag{3.4}$$

where $y_{NB}$ denotes the output by the classifier. The term $P(y_j)$ and a series of distinct $P(a_i|y_j)$ terms are learned from the training set. Each $P(y_j)$ could be easily calculated by counting the frequency that $y_j$ occurs in the training data set. Estimations for the set $P(a_i|y_j)$ can be also proceeded from the training data set.

#### 3.1.1.2 Support Vector Machine

In machine learning, the support vector machine(SVM) algorithm is used for classification and regression. In general, the training data set is in finite dimensional space. The SVM methods map the original space into a higher dimensional space so that the instances of the separate categories are divided by a hyperplane as wide as possible. This higher dimensional space is so called feature space. Then SVM can map new instance into the feature space and predict the category based on which part the instance falls on.

**Notation:** Let $\phi$ be the function mapping the instance from original finite-dimensional space into the feature space. Given a training set consists of labeled instances $(X_i, y_i)$ , where $i \in Z^+, \quad X_i \in R^n$, $y_i \in \{-1, 1\}$, where $i = 1, ..., N$, the support vector machines (Boser et al., 1992; Cortes and Vapnik, 1995) problem can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{w},b,\varepsilon} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad + \quad C\sum_i \varepsilon_i \\ &s.t. \quad y_i(\mathbf{w}^T\phi(X_i) + b) \geq 1 - \varepsilon_i, \quad where \quad \varepsilon_i \geq 0, bisaconstant. \end{aligned} \tag{3.5}$$

Here SVM tries to find the linearly separate hyperplane with the maximal margin. $C > 0$ is the penalty parameter of the error item, and $\mathbf{w}$ is the weight vector. Then learning SVM problem can be equivalently transformed to a quadratic optimization problem.

In addition, we define a function called kernel function to measure the similarity between two instances $X_i$ and $X_j$.

**Definition 3.1.1.** *Kernel Function:* $\quad K(X_i, X_j) \quad = \quad \phi(X_i)^T\phi(X_j)$

This function is called the kernel function. By using Lagrange multipliers $\alpha$, the optimal problem Equation 3.5 can be expressed as

$$\min_{\mathbf{w},b} \max_{\alpha \geq 0} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i \alpha_i \left[ y_i(\mathbf{w}^T\phi(X_i) - b) - 1 \right] \right\} \tag{3.6}$$

where $\forall i, \quad 0 \leq \alpha_i \leq C, \qquad \sum_i y_i\alpha_i = 0.$

**Sequential Minimal Optimization**

In the experiment, the SVM learning approach we used is Sequential Minimal Optimization(SMO). SMO is an iterative algorithm for solving the quadratic problem(QP) in Equation 3.6. For the standard SVM QP problem, the smallest possible optimization problem involves two Lagrange multipliers since the Lagrange multipliers must obey a linear equality constraint. For any two multipliers $\alpha_i, \alpha_j$

$$\begin{cases} 0 \leq \alpha_i, \alpha_j \leq C \\ y_j\alpha_i + y_i\alpha_j = Constant \end{cases} \tag{3.7}$$

For each step, SMO chooses two Lagrange multipliers to jointly optimize, figures out the optimal values and updates the SVM to reflect the new optimal values [23]. When all the Lagrange multipliers satisfy the constraints, the problem is solved.

### 3.1.2 TrAdaBoost

AdaBoost, short for Adaptive Boosting (Freund & Schapire, 1998), is an algorithm for constructing a ‘‘strong’’ classifier as linear combination for ‘‘weak’’ classifiers. AdaBoost aims to boost the accuracy of a ‘‘weak’’ learner by adjusting the weights of training instances, through increasing weight of wrongly classified instance.

TrAdaBoost is an extension of AdaBoost algorithm for transfer learning.

We know that in transfer learning, the source dataset and target dataset are not identical. TrAdaboost takes the same strategy to handle with the same-distribution instances as what is used in Adaboost. However, for the training instances from a different distribution, when they are wrongly predicted, TrAdaBoost proposes a new mechanism to decrease the weights of these instances so as to minimize the negative impact.

Let $\mathbf{X}_s$ be the same-distribution instance space, $\mathbf{X}_d$ be the different-distribution instance space, and $Y = \{0, 1\}$ be the set of class labels. We build a boolean function mapping from $\mathbf{X}$ to $\mathbf{Y}$, where $\mathbf{X} = \mathbf{X}_s \cup \mathbf{X}_d$. $\mathbf{U}$ is the test data set used to measure the accuracy of prediction model. The training dataset used to build the classifier is constituted by two labeled parts $D_S$ and $D_T$. $D_S$ represents the labeled instance of source domain, which is in different distribution from the target dataset that $D_S = \{X_i^s, y_i^s\}$, where $X_i^s \in X_d, \quad (i = 1, ..., n)$. While $D_T$ denotes the labeled instances of target domain that are the same distribution as the target dataset. Here $D_T = \{X_j^t, y_j^t\}$ where $X_j^t \in X_s, j = \{1, ..., m\}$. $m$ and $n$ are the sizes of dataset $D_T$ and $D_S$. So the training dataset $T = \{X_i, y_i\}$ can be defined as follows:

$$X_i \quad = \quad \begin{cases} X_i^s, & i = 1, ..., n; \\ X_i^t, & i = n+1, ..., n+m. \end{cases} \tag{3.8}$$

To be more specific, the objective is to leverage information from the source domain $D_S$ as much as possible; but we have no idea which part of the dataset

can be beneficial. Hence, we can first label a small number of data in the target domain $D_T$, and then use these data to find out the useful part of $D_S$.

The problem can be described in following statements. Given a target dataset with only a small amount of instances $D_T$ and remaining unlabeled test data $U$, and a source dataset $D_S$ with many labeled instances, the objective is to build a predictive model $h(x) : \mathbf{X} \longrightarrow Y$ that minimizes the prediction error on test set $U$. The TrAdaBoost algorithm proposed by Dai et al. [8] in Algorithm 1.

As we can see from the algorithm, in each iterated round, if the instance from source domain $D_S$ is wrongly predicted, we can consider this instance may conflict with the same distribution of training data. Then we decrease its training weight in order to minimize its effect on building classifier for the next round. For the instance from target domain $D_T$, its weight will be improved if it is mistakenly predicted. Hence, after several iterations, the instances from source domain that can fit the target distribution will have relatively larger weights, while the dissimilar instances from source domain will have relatively lower weights. This helps to adapt source domain data to the distribution of target domain.

### 3.1.3 MultiSourceTrAdaBoost

TrAdaBoost uses only one source domain for transfer learning, we are not sure whether multiple domains could help to improve the performance of training a target classifier. Yao et al.(2010) extends the TrAdaBoost frame-

**Algorithm 1** TrAdaBoost

**Input:** Source training data set $D_s$, target training data set $D_T$, the unlabeled data set $U$, a base learning algorithm **Learner**, and the maximum number of iterations $M$

**Output:** Target classifier $h_f(x) : \mathbf{X} \longrightarrow Y$

**Process:**

Initialize the initial weight vector $\mathbf{w}^1 = (w_1^1, ..., w_{n+m}^1)$. Users can assign the initial value for $\mathbf{w}^1$.

**for** $t = 1, ..., m$ **do**

  Set $\mathbf{p}^t = \mathbf{w}^1 / \sum_{i=1}^{n+m} w_i^t$

  Call **Learner**, providing it the combined training set $\mathbf{T}$ with the weight $\mathbf{p}^t$ over $\mathbf{T}$ and the unlabeled dataset $\mathbf{U}$. Then it returns a hypothesis $h_t : \mathbf{X} \longrightarrow Y$

  Calculate the error of $h_t$ on $D_T$: $\epsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t |h_t(x_i - y_i)|}{\sum_{i=n+1}^{n+m} w_i^t}$

  Set $\beta_t = \epsilon_t / (1 - \epsilon_t))$ and $\beta = 1/(1 + \sqrt{2 \ln n / M})$. Note that $\epsilon_t < 1/2$.

  Update the weight vector:

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i) - y_i|}, & 1 \le i \le n \\ w_i^t \beta^{/-|h_t(x_i) - y_i|}, & n+1 \le i \le n+m. \end{cases}$$

**end for**

**return** the hypothesis

$$h_f(x) = \begin{cases} 1, & \prod_{\lceil N/2 \rceil}^{M} \beta_t^{-h_t(x)} \ge \prod_{\lceil N/2 \rceil}^{M} \beta_t^{-1/2} \\ 0, & otherwise \end{cases}$$

work for handling several source domains, in order to reduce the negative transfer effect. This algorithm is named MultiSourceTrAdaBoost. TrAdaBoost algorithm is an instance transfer approach that data from the source domain can be applied directly to the target domain classifier, as well as MultiSourceTrAdaBoost. As we can see from Algorithm 2 (see next page), MultiSourceTrAdaBoost utilizes $N$ source domains $\{D_{S_1}, ..., D_{S_N}\}$ as input, and a combination of $M$ weak classifiers as output. The strategies it used to assign the importance of instances from source domain and target domain remain the same as TrAdaBoost algorithm. The main difference is the weak classifier is selected from multiple source domains. Specifically, in every iteration, each source domain generates its own weak classifier. And we apply a mechanism to choose the one that most closely related to the target domain as the weak classifier at current iteration.

Let $\{D_{S_1}, ..., D_{S_N}\}$ be the N source domains, where $D_{S_k} = \{X_i^{S_k}, y_i^{S_k}\}$, $X_i^{S_k} \in X_d (i = 1, ..., n_{S_k})$. Let $D_T$ denote the labeled instances of target domain that are the same distribution as the target dataset. Here $D_T = \{X_j^t, y_j^t\}$ where $X_j^t \in X_s, j = (1, ..., m)$. $m$ and $n_{S_k}$ are the sizes of dataset $D_T$ and $D_{S_k}$. MultiSourceTrAdaBoost tries to improve the performance of target task classifier $h_f(x) : \mathbf{X} \longrightarrow Y$ by exploring the knowledge of several source domains. The specific description of MultiSourceTrAdaBoost algorithm is given in algorithm 2. From line 7 to 9, it is the mechanism to pick up a ‘‘weak’’ classifier among several ones by computing the error factor on target training dataset $D_T$. We think the smallest error factor response to the most

**Algorithm 2** MultiSourceTrAdaBoost

**Input:** Source training dataset $D_{S_1}, ..., D_{S_N}$, target training data $D_T$, and the maximum number of iterations $M$
**Output:** $h_f(x) : \mathbf{X} \longrightarrow Y$
Set $\alpha_S \doteq \frac{1}{2}\ln(1 + \sqrt{2\ln\frac{n_S}{M}})$, where $n_s \doteq \sum_k n_{S_k}$.
Initialize the weight vector for the source domains and target domain $(\mathbf{w}^{S_1}, ..., \mathbf{w}^{S_N}, \mathbf{w}^T)$, where $\mathbf{w}^{S_k} = (w_1^{S_k}, ..., w_{n_{S_k}}^{S_k})$, and $\mathbf{w}^T = (w_1^T, ..., w_m^T)$.
**for** $t = 1, ..., M$ **do**
    Empty the set of candidate weak classifiers $\tau \leftarrow \emptyset$. Assign the weight vector $(\mathbf{w}^{S_1}, ..., \mathbf{w}^{S_N}, \mathbf{w}^T)$ to value 1.
    **for** $k = 1, ..., N$ **do**
        Find the candidate weak classifier $h_t^k : \mathbf{X} \longrightarrow Y$ that minimizes the classification error over the combined set $D_{S_k} \cup D_T$ weighted according to $(\mathbf{w}^{S_k}, \mathbf{w}^T)$.
        Compute the error of $h_t^k$ on $D_T$:
        $\epsilon_t^k = \sum_j \frac{w_j^T\left[y_j^T \neq h_t^k(X_j^T)\right]}{\sum_i w_i^T}$
        $\Gamma \leftarrow \Gamma \cup (h_t^k, \epsilon_t^k)$
    **end for**
    Find the weak classifier $h_t : \mathbf{X} \longrightarrow Y$ such that
    $(h_t, \epsilon_t) = \arg\min_{(h,\epsilon)\in\tau} \epsilon$
    Set $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$, where $\epsilon_t < 1/2$
    Update the weight vector
    $w_i^{S_k} \leftarrow w_i^{S_k} e^{-\alpha_S\left|h_t(X_i^{S_k}) - y_i^{S_k}\right|}$
    $w_i^T \leftarrow w_i^T e^{\alpha_t\left|h_t(X_i^T) - y_i^T\right|}$
**end for**
**return** $h_f(x) = sign(\sum_t \alpha_t h_t(\mathbf{X}))$

closely related to the target. We can also see that when $N = 1$, the algorithm reduces to TrAdaBoost.

In our experiment of comparison between TrAdaBoost and MultiSource-TraBoost, we set the total number of labeled instances from different distribution remaining the same, whereas the number of source domains are different. Then the performance can be enhanced directly since the source domain number is the only control variable.

### 3.1.4 Self-training

As we mentioned in the background, self-training algorithm is a simple and common used method in semi-supervised learning. Here we give the general description of this algorithm.

Assumes that $L$ indicates the labeled instance set and $U$ denotes the unlabeled instance set, where the amount of unlabeled instances is much bigger than the number of labeled instances. $L$ and $U$ are in identical distribution. $\theta$ is the number of selected new labeled instances for each iteration of self-training. The Algorithm 3 in next page is the detailed procedure of self-training.

The self-training algorithm is first built on labeled instances with a base learner, and then the generated classifier makes predictions for large number of unlabeled instances. After the classification, only part of the instances are selected to enlarge the labeled instances set. These selected instances are removed from the unlabeled data set, meanwhile we add them to the labeled

data set. The classifier then is trained on the new labeled instances set and repeats the dataset update procedure until the unlabeled dataset is broken or the iteration threshold value is break.

**Algorithm 3** Self-Training

1: **Input:** labeled instances $L$, Unlabeled instances $U$ a base learning algorithm **Learner** , maximum number of iterations $M$, and $\theta$
2: **Output: Classifier** $C$
3: **Initialize:** Set $\mathbf{L}_t = \mathbf{L}, \mathbf{U}_t = \mathbf{U}$, where $\mathbf{L}_t$ and $\mathbf{U}_t$ are the labeled and unlabeled dataset at the $t$th iteration.
4: **for** $t \leftarrow 1 \quad to \quad M$ **do**
5: **if** $\mathbf{U}t$ is empty **then**
6: **return** $C$
7: **end if**
8: Empty the instances set $\mathbf{S}_t$, $\mathbf{S}_t \leftarrow \emptyset$
9: Train C on labeled instances set $\mathbf{L}_t$ by **Learner**
10: Apply C on unlabeled instances set $\mathbf{U}_t$, pick up $\theta$ instances into dataset $\mathbf{S}_t$.
11: $\mathbf{U}_{t+1} = \mathbf{U}_t - \mathbf{S}_t; \mathbf{L}_{t+1} = \mathbf{L}_t + \mathbf{S}_t$
12: **end for**
13: **return** $C$

Self-training is kind of single-view bootstrapping method.The core of the self-training framework is the base learner. Because the underlying classifier determines the prediction of new instances in $S_t$, with the incrementally update of labeled instances set, the prediction model keeps renewable until the iterative procedure ends. In our experiment, we use two different base learners: Naive Bayes and SVM. Moreover, the other main component of self-training is the selection metric. The common strategies used for selecting unlabeled instances at each iteration are ‘‘random selection’’ and ‘‘confident selection’’. In our experiment, we choose the ‘‘confident selection’’ as the

selection metric, though observations show that particularly selecting the most confident unlabeled data in self-training and co-training is not necessarily superior to randomly selecting unlabeled instances [13].

### 3.1.5 Co-training

By comparison to self-training algorithm, co-training focuses on the multiple-view learning. It is first motivated by solving web-page classification problem. The web-page classification problem can be partitioned into two distinct views, such as the words on that page and the hyper-links pointing to that page. It supposes that each of the two views is given enough information to train a classifier separately. Co-training combines two classifier to boost the performance.

A formal description of co-training is given in Algorithm 4. Assumes that $L$ denotes labeled instances, $U$ represents unlabeled instances. The set of attributes $\mathbf{A}$ can be divided into two parts $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ and each instance $X$ can be expressed by $X = (\mathbf{A}_1(X), \mathbf{A}_2(X))$. Two classifiers $f_1$ and $f_2$ are trained based on $\mathbf{A}_1$ and $\mathbf{A}_2$ respectively.

From Algorithm 4, we can see that the original attribute set is divided into two subsets according to the two sub-views, then two classifiers are initially trained by using the labeled instances on both subsets of attributes respectively. After that, each classifier makes prediction on the unlabeled instances and teaches the other classifier by selecting most confident predicted instances to extend the labeled training dataset. Each classifier is retrained with the

**Algorithm 4** Co-training

**Input:** Unlabeled instances set $L$, labeled instances set $U$, two learners $f^{(1)}, f^{(2)}$, and maximum iteration number $M$.
**Output:** The combination of predictions for new instances from classifiers $f^{(1)}, f^{(2)}$.
**Initialization:** Feature split $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$. ($\mathbf{A}_1$ or $\mathbf{A}_2$) alone is sufficient to train a good classifier, at the same time, $\mathbf{A}_1$ and $\mathbf{A}_2$ are conditionally independent given the class.
Create a pool of instances $\mathbf{X}_u$ by randomly choose some unlabeled instances from dataset $\mathbf{U}$.
labeled instances set $L$ can be split into two sets $L^{(1)}, L^{(2)}$ according to the sub-attributes set
**for** $t \leftarrow 1$ *to* $M$ **do**
  Train two classifiers: $f^{(1)}$ from $L^{(1)}$, $f^{(2)}$ from $L^{(2)}$.
  Classify the instances $\mathbf{X}_u$ with $f^{(1)}$, $f^{(2)}$ separately.
  Pick up $f^{(1)}$'s k-most-confident instances into $f^{(2)}$'s labeled data set $L^{(2)}$ from $\mathbf{X}_u$. Pick up $f^{(2)}$'s k-most-confident instances into $f^{(1)}$'s labeled data set $L^{(1)}$ from $\mathbf{X}_u$.
  Randomly choose $2k$ instances from $\mathbf{U}$ to replenish $\mathbf{X}_u$.
  **if** $\mathbf{X}_u$ is empty **then**
    **return**
  **end if**
**end for**
**return**

updated training set. The iterative process is ended when certain conditions are satisfied, such as the unlabeled dataset is empty or the maximum iterative number is reached.

Furthermore, we have to mentioned that co-training method is developed based on the following two assumptions: (1) each subset is sufficient to build a good classifier; (2) the two subsets are conditionally independent of each other. However, these two assumptions may not be satisfied in practical application. But the performance of co-training is still considerable when the attribute set is split randomly. In our experiment, we take the strategy that the features set is randomly split into two subsets.

## 3.2 Datasets

We apply some UCI datasets from a package of 37 classification problems, ''datasets-UCI.jar''[1]. 19 datasets are picked out from the package, and the detailed information of them is listed in Table 3.1. Other datasets in the package are rejected due to their inadequacy as follows.

- **Imbalanced distribution:** Some datasets have extremely skewed class distribution. Consider a problem where 99% of the data is in one class, while the rest 1% belongs to the other rate class. When we are sampling the labeled part for semi-supervised learning or splitting the dataset for transfer learning, these instances with minor class may not

[1]The data package is available from http://www.cs.waikato.ac.nz/ml/weka/datasets.html

appear in the generated training dataset.

- **Excess number of nominal attributes:** The datasets with an incredible number of nominal attributes may cause the memory overflow problem when used to train an SMO kernel. Since some kernels can only work with numeric attributes, which means SMO has to process these nominal attributes before processing data, such as converting them to numeric attributes. This usually causes the memory to be insufficient for building the model.

Table 3.1: Description of 19 UCI datasets

| **Dataset** | **# instances** | **# attributes** | **# classes** | **frequency of each class** |
|---|---|---|---|---|
| balance-scale | 625 | 5 | 3 | 288,49,288 |
| breast-cancer | 286 | 10 | 2 | 201,85 |
| colic | 368 | 23 | 2 | 232,136 |
| colic.ORIG | 268 | 27 | 2 | 244,124 |
| credit-a | 690 | 16 | 2 | 307,383 |
| credit-g | 1000 | 21 | 2 | 700,300 |
| diabetes | 768 | 9 | 2 | 500,268 |
| heart-c | 303 | 14 | 2 | 165,138 |
| heart-h | 294 | 14 | 2 | 188,106 |

*Continued on next page*

Table 3.1 -- *Continued from previous page*

| **Dataset** | **# instances** | **# attributes** | **# classes** | **frequency of each class** |
|---|---|---|---|---|
| heart-statlog | 270 | 14 | 2 | 150,120 |
| mushroom | 8124 | 23 | 2 | 4208,3916 |
| segment | 2310 | 20 | 7 | all for 330 |
| sick | 812 | 29 | 2 | 755,57 |
| sonar | 67 | 60 | 2 | 25,42 |
| soybean | 235 | 35 | 18 | 7,6,5,32,16,5,6,37,6,6,5 |
| -- | -- | -- | -- | 13,5,30,29,4,7,16 |
| vehicle | 846 | 19 | 4 | 212,217,218,199 |
| vote | 435 | 17 | 2 | 267,168 |
| vowel | 990 | 14 | 11 | all for 90 |
| waveform | 3347 | 41 | 2 | 1655,1692 |

### 3.2.1 Pre-process

Our experiment platform is WEKA, so the datasets should be in WEKA file format. In addition, every dataset needs to be preprocessed in WEKA software.

- **Missing values:** If there are missing values in the dataset, *"weka.filters.unsupervised.attribute*
  will help to fill the blank. This filter needs some data to assist in deter-

mining distribution of each attribute. The filter will fail to replace the missing values without the help of these data.

- **Nominal attribute:** The numeric attributes will be discretized into nominal ones by the filter *''weka.filters.supervised.attribute.Discretize-B2-M-1.0-Rfirst-las''* based on the class information. Then, all these attributes are considered as nominal attributes in the experiment.

- **Redundant attribute:** Some attribute nearly has the same number of values as the number of instances, which could not make a difference for classification. Thus, we can remove this kind of redundant attribute from dataset by using the filter *''weka.filters.unsupervised.attribute.Remove''*

All the datasets in our experiment will be pre-processed as what we described above in WEKA software. The package we used for dataset manipulation at initial phase is *weka.filters*, which is employed for input data preprocessing, transformation, feature generation and so on.

### 3.2.2 Dataset Split

The 19 selected datasets do not have hierarchical structures, so the data are split randomly in order to generate the same-distribution and different distribution sets for transfer learning.

We first randomly choose one attribute from the features, and divide the dataset into two groups based on attribute values. One group is set to the target domain, then the other one is the source domain. When it comes to

MultiSourceTrAdaBoost, multiple source domains are needed for the training process. We continue splitting the group data of source domain. This assures that the number of instances from different-distribution are the same when comparing the two algorithms. Here we take ‘‘mushroom’’ dataset as an example to explain the specific process of data split.

The attribute ‘‘*cap-surface*’’ is randomly chosen as the split feature. This attribute has four different values: *fibrous, grooves, scaly* and *smooth*. The target domain contains all the instances whose ‘‘*cap-surface*’’ is *smooth*, while the source domain for TrAdaBoost consists of all the instances whose cap-surface is *fibrous, grooves* or *scaly*. This source domain will be further divided into several sub parts for the MultiSourceTrAdaBoost. Here, we give only two source domains as multiple sources. Since the feature *cap-surface* has left 3 values in original source domain including *fibrous, grooves* and *scaly*, the instances can be assigned into two groups to generate two source domains: one group has the instances with value *grooves* in attribute *cap-surface*, and the other one consists of all instances whose ‘‘*cap-surface*’’ is either *fibrous* or *scaly*. Hence, for TrAdaBoost and MultiSourceTrAdaBoost algorithm, their instances of target domain stay the same with each other, as well as the instances form the source domain(s). We will take a look at all 19 datasets split structure in the following Table 3.2. KL-divergence (Kullback & Leibler, 1951) is introduced on the feature space between the source domain(s) and target domain. KL-divergence, also named information gain, is a non-symmetric measure of the difference between two probability

distributions $P$ and $Q$ [31]. For discrete probability distribution P and Q, the KL-divergence of Q from P is defined as followed:

$$D_{KL}(P||Q) = \sum_i P(i) \ln(\frac{P(i)}{Q(i)}) \tag{3.9}$$

where $\sum_i P(i) = 1$ and $\sum_i Q(i) = 1$. If $P(i) = 0$, then we set the $0 \ln \frac{0}{Q(i)} = 0$. And if $Q(i) = 0$, the value is set to infinity: $P(i) * \ln(\frac{P(i)}{0}) = \infty$. In other words, if two distributions are exact the same, then the KL-divergence value should be 0, otherwise, the value should be a positive number. So if two domains are the same distribution, the KL-divergence for the two sets should be close to zero. By calculating the KL value for 19 datasets, the split process leads to two datasets in different distributions but related to each other.

Table 3.2: The description of data split in the experiment

| No. | Dataset | Split-feature | Size | |
|---|---|---|---|---|
| | | | $D_S$ | $D_T$ |
| 1 | balance-scale | right-distance | 375 | 250 |
| 2 | breast-cancer | breast-quad | 175 | 111 |
| 3 | colic | temp-extremities | 168 | 200 |
| 4 | colic.ORIG | pain | 223 | 252 |
| 5 | credit-a | A2 | 438 | 252 |
| 6 | credit-g | residence-since | 587 | 413 |
| 7 | diabetes | pres | 508 | 260 |
| 8 | heart-c | chol | 210 | 106 |
| 9 | heart-h | trestbps | 196 | 98 |
| 10 | heart-statlog | resting-blood-pressure | 170 | 100 |
| 11 | mushroom | cap-surface | 4880 | 3244 |
| 12 | segment | region-centroid-col | 1360 | 950 |
| 13 | sick | FTI | 2960 | 812 |
| 14 | sonar | goitre | 141 | 67 |
| 15 | soybean | crop-hist | 448 | 235 |
| 16 | vehicle | SKEWNESS ABOUT-MINOR | 625 | 846 |
| 17 | vote | immigration | 223 | 212 |
| 18 | vowel | Feature9 | 636 | 354 |
| 19 | waveform | X40 | 1650 | 1697 |

Table 3.3: The datasets split in the experiment

| Dataset | Class distribution | |
|---|---|---|
| | $D_S$ | $D_T$ |
| balance-scale | 0.53,0.08,0.39 | 0.36,0.08,0.56 |
| breast-cancer | 0.72,0.28 | 0.68,0.32 |
| colic | 0.55,0.45 | 0.7,0.3 |
| colic.ORIG | 0.68,0.32 | 0.63,0.37 |
| credit-a | 0.72,0.28 | 0.48,0.52 |
| credit-g | 0.7,0.3 | 0.7,0.3 |
| diabetes | 0.64,0.36 | 0.7,0.3 |
| heart-c | 0.94,0.06 | 0.58,0.42 |
| heart-h | 0.62,0.38 | 0.68,0.32 |
| heart-statlog | 0.55,0.45 | 0.57,0.43 |
| mushroom | 0.55,0.45 | 0.46,0.54 |
| segment | 0.09,0.16,0.11,<br>0.15,0.18,0.15,0.16 | 0.23,0.12,0.19,<br>0.13,0.08,0.13,0.12 |
| sick | 0.94,0.06 | 0.93,0.07 |
| sonar | 0.51,0.49 | 0.37,0.63 |
| soybean | 0.03,0.03,0.03,0.13,0.06,<br>0.03,0.03,0.13,0.03,0.03,<br>0.03,0.07,0.03,0.14,0.15,<br>0.03,0.01,0,0.01 | 0.03,0.02,0.03,0.14,0.07,<br>0.02,0.02,0.16,0.02,0.02,<br>0.02,0.06,0.02,0.13,0.12,<br>0.02,0.03,0.07,0 |
| vehicle | 0.24,0.27,0.25,0.24 | 0.24,0.26,0.27,0.23 |
| vote | 0.57,0.43 | 0.66,0.34 |
| vowel | 0.13,0.09,0.09,0.09,0.09,<br>0.08,0.08,0.06,0.1,0.09,0.1 | 0.02,0.1,0.09,0.09,0.09,<br>0.11,0.1,0.14,0.08,0.1,0.08 |
| waveform | 0.51,0.49 | 0.48,0.52 |

# Chapter 4

# Experiment

In this chapter, the pre-process of datasets, performance among different methods and analysis of experiment results are displayed.

The previous work for transfer learning demonstrated that both TrAdaBoost and MultiSourceTrAdaBoost algorithms improve the performance by leveraging knowledge from auxiliary data. For semi-supervised learning, researchers found unlabeled data from same distribution can help semi-supervised learning in some degree, but not always. Sometimes the utilization of unlabelled data may even lower the performance of classification when the model assumptions are not applicable to the distribution.

The purpose of experiments here is to investigate and compare the effects of both transfer learning and semi-supervised learning methods on 19 UCI datasets when the labelled data is scarce and time-consuming to obtain but auxiliary data is available. All of algorithms used in the experiments were

implemented with Java based on WEKA software platform.

## 4.1 Experimental Settings

### 4.1.1 Platform

WEKA is an open source data mining software with a collection of machine learning algorithms. With the tools for data pre-processing, classification, regression, clustering , association rules, and visualization, WEKA is a powerful software for either performing the task at hand or adding new algorithms. The packages mainly used in the studies are listed below.

■ ***weka.filters*** package is applied for data preprocessing including removing or adding attributes, resampling the dataset, modifying examples and so on. This step is indispensable since the input dataset needs to be standardized and organized.

■ ***weka.classifiers*** package is the core of WEKA which includes the common structure for data mining algorithms, as well as a wide range of classifiers. The underlying algorithm Naive Bayes and SMO already exist in WEKA framework, so we can directly call them. For the methods used in transfer learning and semi-supervised learning, they are not covered in current release. WEKA is friendly to extending new algorithms which based on its architecture.

*weka.classifiers.Classifier*, an abstract class, is the ancestor of all classifiers in WEKA. In order to make it easier for new algorithm implemen-

tation, WEKA supplies a range of other abstract classes derived from *weka.classifiers.Classifier*. And we will use two of them for our experiment.

- **Semi-supervised learning algorithm** *weka.classifiers.collective* package is applied for development of semi-supervised learning methods. For co-training algorithm implementation, it inherits the abstract class *CollectiveRandomizableMultipleClassifiersCombiner* because this abstract classifier supports a randomizable collective classifier combining multiple classifiers. While in the implementation of self-training paradigm, *weka.classifiers.collective* is not utilized since self-training is simple and easy to operate. It can directly extend from the abstract class *weka.classifiers.Classifier*.

- **Transfer learning algorithm** The implementation of TrAdaBoost and MultiSourceTrAdaBoost methods are both derived from the super class named *weka.classifiers.IteratedSingleClassifierEnhancer* since the classifier is iterated but not randomizable.

- **Evaluation** Evaluation is a vital part in package *weka.classifiers* that many common options for classifier are related to evaluation purpose. As we mentioned before, our experiment is performed by cross-validation. Hence, the test file is not specified.

### 4.1.2 Data Settings

The performance is measured by the prediction accuracy on test dataset. In order to inspect whether the algorithm would help to improve the classification performance, we set the corresponding underlying learner which is trained only from the labelled data as the baseline. The original dataset from the UCI package can be expressed as **D**. Instances of this dataset will be split into two sub sets **S** and **T**. For transfer learning algorithm, dataset **S** is considered as source domain and **T** is treated as target domain. While in semi-supervised learning and baseline training, we only reserve dataset **T** for this part experiment.

In addition, 4-fold cross-validation will be conducted for the dataset **T**, which aims to ensure that every instance from **T** has the equal opportunity of appearing in the training and testing set. In order words, 25% instances of dataset **T** will be set aside for testing data. The other 75% data are training dataset, used for training process. Then instances in training dataset are partitioned into labelled dataset **L** and unlabelled part **U** according to a given ratio $l_p$.

The training dataset can be estimated from different views. In transfer learning, the labelled dataset **L** is utilized as same-distribution training data, while the different-distribution training data from dataset **S** constitute the source domain. In the base learner training, the labelled dataset **L** can only be used during the training process. While for the semi-supervised algorithms, both **L** and **U** are participated in training the classifier. $l_p$ is the percentage

of labelled data in training dataset. In the experiment, we set $l_p$ to three different values: $5\%, 10\%$ and $20\%$ to examine whether the performance differences exist.

## 4.2 Performance and Result Analysis

Our experiments are conducted mainly from the following five parts: (1) Which one outperforms in using auxiliary data, transfer learning or semi-supervised learning? (2) Is multi source domains superior to the single source domain in transfer learning? (3) How does the relationship of source domain and target domain affect the performance of target classifier? (4) Does the auxiliary data really help upgrade the classifier? (5) How does the amount of labelled data affect the performance of the four algorithms?

### 4.2.1 Comparison between Transfer Learning and Semi-supervised Learning Methods

Both transfer learning and semi-supervised learning do not abide by the traditional machine learning assumptions. They attempt to improve the classifier by using auxiliary data. Transfer learning utilizes the similar and related labelled data from different distribution, while semi-supervised learning makes use of the unlabelled data in the same distribution. We would like to see which one performs better when it comes to the dataset with scarce labelled data. Here we compared the performance among TrAdaBoost,

MultiSourceTrAdaBoost, Self-training and Co-training on 19 UCI datasets.

The following Figures 4.1 - 4.6 display the performance evaluated by prediction accuracy of each approach, ‘‘TL-1’’ and ‘‘TL-2’’ represent the TrAdaBoost and MultiSourceTrAdaBoost method in transfer learning, at the same time, ‘‘SSL-1’’ and ‘‘SSL-2’’ denote self-training and co-training algorithm in semi-supervised learning. The sequence number on the horizontal axis stands for the 19 datasets in order, same as the column of ‘‘No.’’ shows in Table 3.2. The contrast of these two machine learning branches is executed on setting different $l_p$ values.

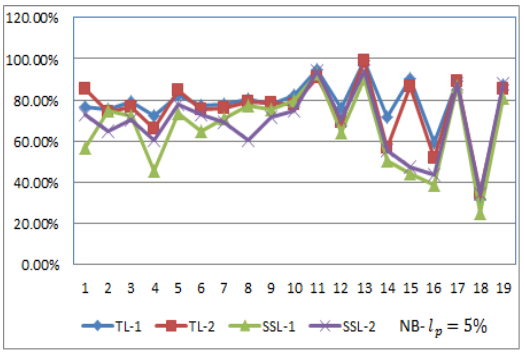


Figure 4.1: Transfer learning VS. Semi-supervised learning based on Naive Bayes when $l_p = 5\%$

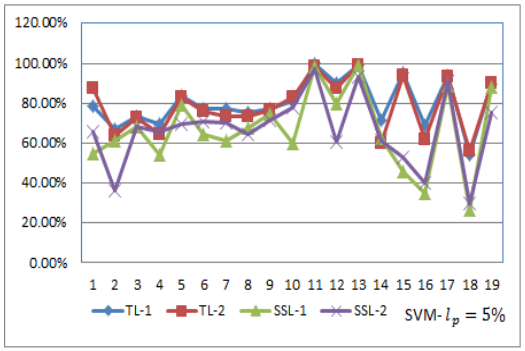


Figure 4.2: Transfer learning VS. Semi-supervised learning based on SVM when $l_p = 5\%$

From the Table 4.1 to Table 4.6, the labelled data from a related but different distribution bring more benefits than the unlabelled data in the same distribution on the majority of datasets. In addition, transfer learning methods have obvious advantages over semi-supervised learning ones when $l_p$ is small. Especially for dataset *sonar* and *soybean*, the accuracy achieves

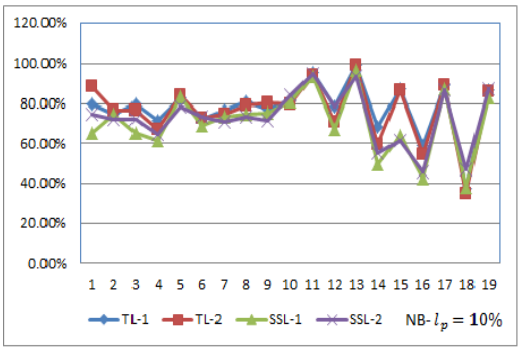


Figure 4.3: Transfer learning VS. Semi-supervised learning based on Naive Bayes when $l_p = 10\%$

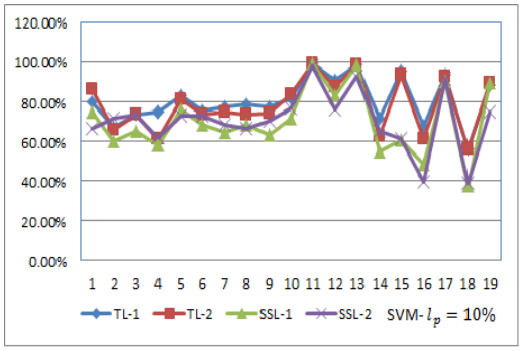


Figure 4.4: Transfer learning VS. Semi-supervised learning based on SVM when $l_p = 10\%$

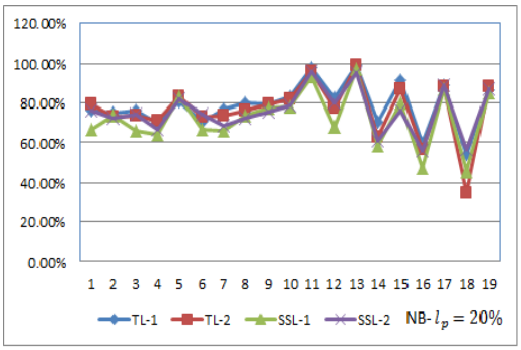


Figure 4.5: Transfer learning VS. Semi-supervised learning based on Naive Bayes when $l_p = 20\%$

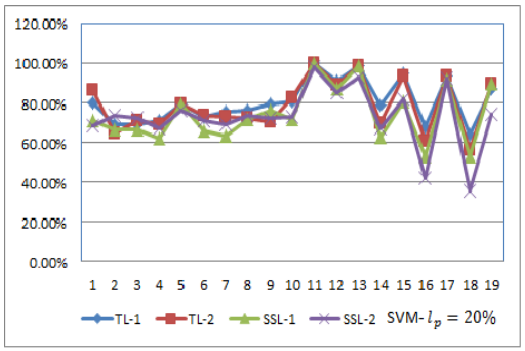


Figure 4.6: Transfer learning VS. Semi-supervised learning based on SVM when $l_p = 20\%$

below 60% for semi-supervised learning methods, which is quite bad. We noticed that these two datasets have quite a small quantity of instances. Since the ratio of labelled data for training process is set to $5\%, 10\%$, and $20\%$, the actual number of labelled data that participate in training the classifier can even less than 10 in the end. Thus, the semi-supervised learning methods failed to take good advantage of unlabelled data with such few labelled data. Although the number of labelled data in the same distribution stays unchanged for transfer learning methods, the labelled data from different distribution can make more contribution than the unlabelled data in the same distribution for our 19 experimental datasets.

### 4.2.2 Comparison between TrAdaBoost and MultiSource-TrAdaBoost

Yao and Doretto pointed out that the multi sources can improve the performance by greatly reducing the negative transfer as the number of source domain increases [35]. Comparing to MultiSourceTrAdaBoost algorithm, TrAdaBoost method utilizes only one source, therefore makes itself vulnerable to negative transfer. However, here the total number of instances from multiple source domains are equal to the amount of instances in a single source domain. In other words, we only care about the effect of source domain number: whether it will help when training the certain instances from multiple domains, instead of other factors such as instances conditions.

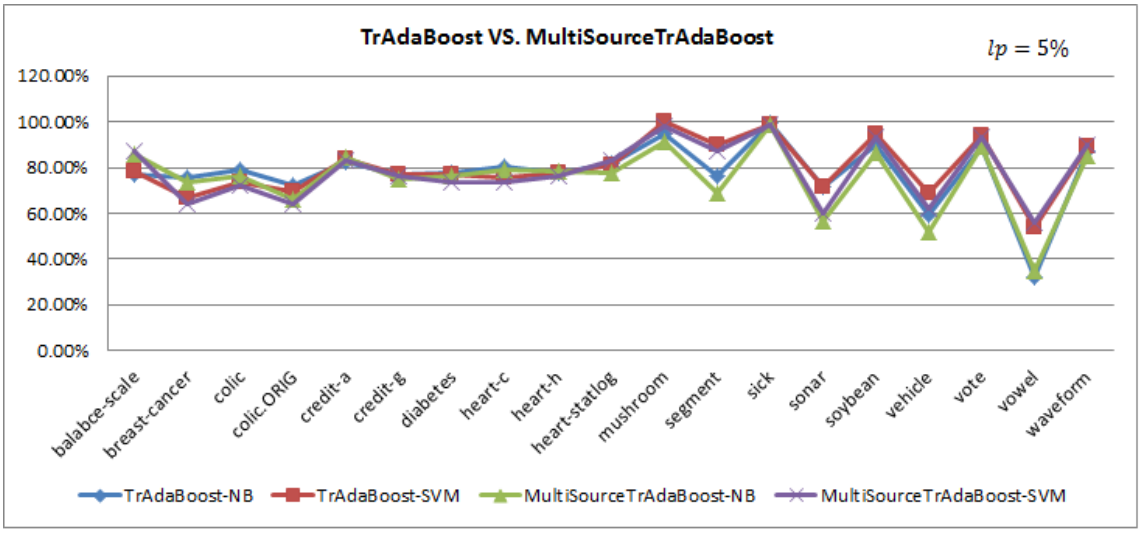


Figure 4.7: Comparison between TrAdaBoost and MultiSourceTrAdaBoost ($l_p = 5\%$)

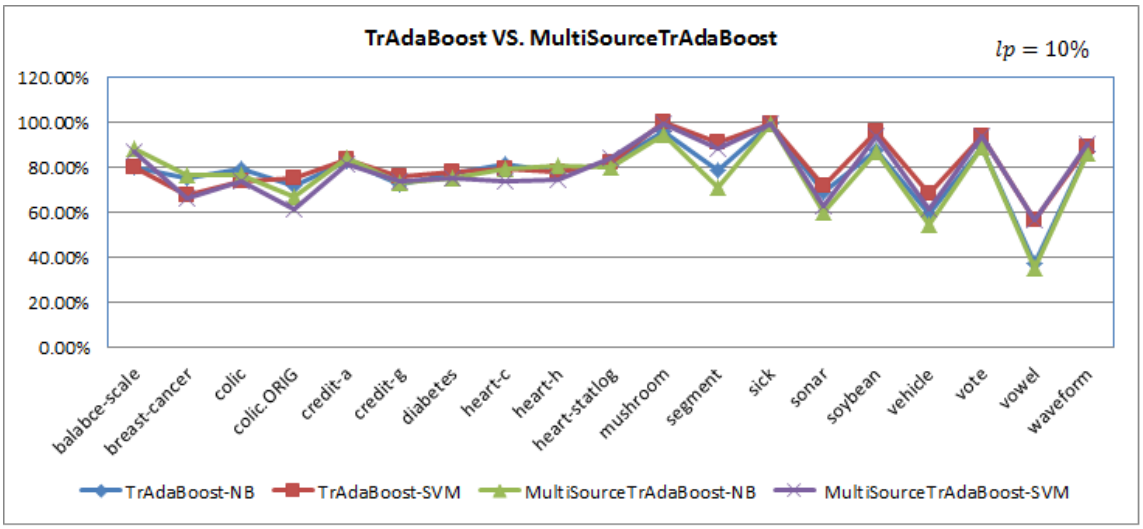


Figure 4.8: Comparison between TrAdaBoost and MultiSourceTrAdaBoost ($l_p = 10\%$)

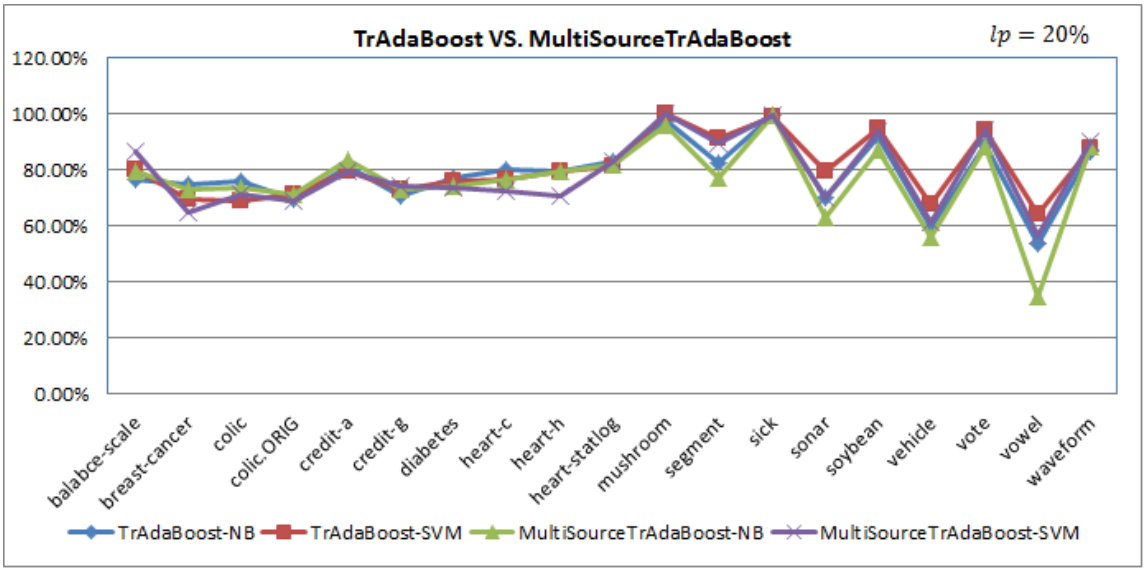


Figure 4.9: Comparison between TrAdaBoost and MultiSourceTrAdaBoost ($l_p = 20\%$)

Despite the line charts, we do t-test for these two methods so as to compare performance between TrAdaBoost and MultiSourceTrAdaBoost more straightforward. Table 4.1 shows the "$w/t/l$" counts of t-test for using TrAdaBoost against MultiSourceTrAdaBoost on 19 UCI datasets. The entry "$w/t/l$" means how many times the learning method $wins/ties/loses$ against the corresponding base learning method on 19 UCI datasets.

Table 4.1: $w/t/l$ counts of T-Test for using TrAdaBoost against MultiSource-TrAdaBoost.

| Base Learner | $l_p = 5\%$ | $l_p = 10\%$ | $l_p = 20\%$ |
|---|---|---|---|
| Naive Bayes | 15/2/2 | 16/1/2 | 12/0/7 |
| SVM | 14/2/3 | 9/3/7 | 10/5/4 |

From Figures 4.7, 4.8 and 4.9, the operation that divides instances into multiple sources may not improve the classifier compared to the case of single source domain. This is proved by the t-test results showed in Table 4.1. We can observe that the worst case for single source is 9/3/7, which suffers defeat on 7 datasets when $l_p = 10\%$ and base learner is SVM. In the case that the total instances stay the same, the filter mechanism of MultiSourceTrAdaBoost removes a certain number of instances, for the reason that these instances in current domain behave poorly compared with the ones from other source domains in the training process. As a matter of fact, if we can extract some information from these filtered instances, then the discard of instances results in less information exploration. However, when it comes to negative transfer, this strategy may help train the classifier with resistance to negative information.

### 4.2.3 Effects of Source Domain to Performance of Target Task

Theoretically, the more similar for source domain(s) and target domain, the more benefits we can get from the source domain(s). It is not difficult to explain. Because some knowledge is specific for individual domains or tasks, whereas some knowledge is common between different domains. The mutual information can help improve performance for the target task [25]. If two domains are closer to each other, then the more common knowledge

they should share. In this section, we design the experiment to validate this viewpoint.

As what we mentioned in Chapter 3, the original datasets are split to generate source domain and target domain based on one attribute that we randomly choose. In order to observe how the similarity of source domain and target domain affects the transfer learning performance, the dataset ‘‘$credit-a$’’ is selected to be divided into two parts according to different attributes such as $attribute \quad A1$, $attribute \quad A2$, $attribute \quad A3$, $attribute \quad A9$, and $attribute \quad A10$.

Table 4.2: Information Gain value for different attributes in credit-a

| Attr | Information Gain | Attr | Information Gain | Attr | Information Gain |
|---|---|---|---|---|---|
| A1 | 6.02957169397E-4 | A6 | 0.1091602084431 | A11 | 0.11530082487874 |
| A2 | 0.02804516154760 | A7 | 0.0501885928293 | A12 | 7.212855736536E-4 |
| A3 | 0.04789832097358 | A8 | 0.0690887319356 | A13 | 0.0100363888467 |
| A4 | 0.02960326183246 | A9 | 0.4257094266728 | A14 | 0.0079190225429 |
| A5 | 0.02960326183246 | A10 | 0.1562855228837 | A15 | 0.0205966937212 |

These attributes have different information gains, which is a synonym for Kullback-Leibler divergence. It is used to decide which attributes are the most relevant. If the dataset is split based on the attribute with higher information gain, we can see the two sub-datasets have less mutual information, that is to say, less relevant. The information gain values $IG$ of attributes in dataset ‘‘$credit-a$’’ are listed in Table 4.2. Apparently, we can see $IG(A9) > IG(A10) > IG(A5) > IG(A2) > IG(A1)$.

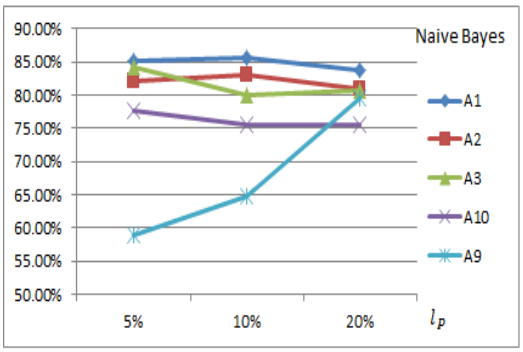


Figure 4.10: Performance comparison among different source domains-NB

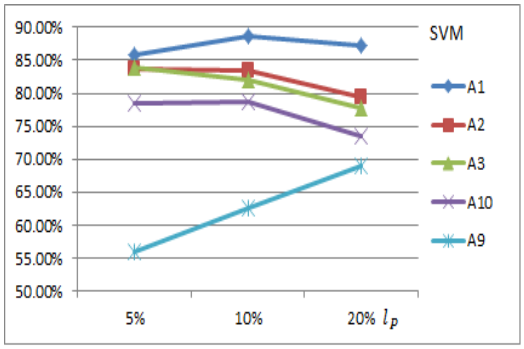


Figure 4.11: Performance comparison among different source domains-SVM

Figures 4.10 and 4.11 show the results of different source domains on base learner Naive Bayes and SVM. With a given $l_p$, the performance of sub-datasets splitting on attribute A1 is superior to others, and in most case, the performance of sub-datasets splitting on attribute A9 is at a huge disadvantage. Moreover, the predictive accuracy decreased with the increment of splitting attribute's information gain value in general. The experimental results illustrate that, if the source domain(s) have more mutual knowledge with the target domain, the performance for transfer learning is better. That is to say, the more closely two distributions related, the more information can be transferred from the source domain.

### 4.2.4 Influence of Auxiliary Data

In the real world application, there are many cases that the labelled data are not sufficient to produce a good classifier by using traditional machine learning methods. Hence, many researches turn to the auxiliary data to

improve the classifier performance. In order to examine whether auxiliary data really do help to a better performance, we set the corresponding base classifiers that learned from only the original labelled data as the baseline results to observe the effect of auxiliary data.

On purpose to intuitively watch the performance of auxiliary data, Table 4.3 and Table 4.4 display the "*w/t/l*" counts of t-test for using transfer learning and semi-supervised learning approaches against responding base learner.

Table 4.3: Comparison between TL methods and base learner

| | $l_p$ | TrAdaBoost | MultiSourceTrAdaBoost |
|---|---|---|---|
| NaiveBayes | 5% | 19/0/0 | 14/1/4 |
| | 10% | 18/0/1 | 16/0/3 |
| | 20% | 15/0/4 | 13/1/5 |
| SVM | 5% | 18/1/0 | 16/1/2 |
| | 10% | 17/1/1 | 14/4/1 |
| | 20% | 16/0/3 | 16/1/2 |

Table 4.4: Comparison between SSL methods and base learner

| | $l_p$ | Self-training | Co-training |
|---|---|---|---|
| NaiveBayes | 5% | 5/0/14 | 12/0/7 |
| | 10% | 9/0/10 | 12/0/7 |
| | 20% | 6/1/12 | 8/0/11 |
| SVM | 5% | 7/3/9 | 11/0/8 |
| | 10% | 4/1/14 | 9/0/10 |
| | 20% | 10/0/9 | 9/0/10 |

In Table 4.3, transfer learning methods upgrade the classifier obviously

on 19 experimental datasets. Although the distribution of source domains are different from the one of target domain, source domain can also bring positive knowledge for training the target task. These kind auxiliary data are labelled and related to the target domain. However, the labelled data from source domain are not always beneficial, we can see that sometimes the transfer learning method can hurt the performance, since it is defeated by the base learners which train the classifier only by the small number of labelled data in the target domain.

Yuan et. al demonstrated that semi-supervised learning methods may not always perform as well as ideally expected [13]. Generally, if the selected unlabelled instances are correctly classified at the initial stage, the standard self-training and co-training could improve the performance. As in Table 4.4, the semi-supervised learning approaches may not help in most of the 19 datasets, especially for the standard self-training method.

From the 19 UCI datasets, the unlabelled auxiliary data are failed to improve the classifier if the unlabelled data are incorrectly predicted by the initial classifier. While the labelled auxiliary data from a related domain are not that vulnerable. The effectiveness of transfer method depends on the how close the source task is related to the target one. If the relationship is strong then the transfer method can take advantage of it. Both TrAdaBoost and MultiSourceTrAdaBoost are inductive transfer learning methods that the labelled auxiliary data are re-weighted according to their effectiveness on the target domain. Although the performance of these auxiliary data are also

affected by the weak initial classifier, their class labels are not changed but their impacts on target domain are decreased.

### 4.2.5 Relationship between Amount of Labelled Data Matter and Auxiliary Data Performance

In our experiments, we use three different percentages of labelled data in target domain: $5\%, 10\%$ and $20\%$. Here we choose TrAdaBoost algorithm as representative for transfer learning and co-training for semi-supervised learning to analysis the performance with different values of $l_p$.

Table 4.5 shows the result of the accuracy comparison among different given percentages of labelled data ($l_p$) for TrAdaBoost algorithm. The item ''Average'' is the mean value of using three $l_p$ in the current base learner. And the predictive accuracy differences between current $l_p$ and average value are also showed in the table. It can be observed that the performance does not change obviously with the different values of $l_p$ on experimental datasets excluding the *vowel*. In addition, the TrAdaBoost gets improved performance on 8 datasets when $l_p$ increased from 5% to 10%, and 8 datasets when $l_p$ increased from 10% to 20% with the Naive Bayes as base leaner. And for the base leaner SVM, the situation is similar: prediction accuracy increases on 9 datasets when $l_p$ changes from 5% to 10%, and only 6 datasets when $l_p$ varies from 10% to 20%.

Here we only list out the TrAdaBoost algorithm as the representative.

Table 4.5: Accuracy comparison among different $l_p$ for TrAdaBoost

| Dataset | Naive Bayes | | | | SVM | | | |
|---|---|---|---|---|---|---|---|---|
| | Average | 5% | 10% | 20% | Average | 5% | 10% | 20% |
| balance-scale | 77.60 | -0.80 | 2.00 | -1.20 | 79.47 | -1.07 | 0.53 | 0.53 |
| breast-cancer | 75.08 | 0.60 | -0.30 | -0.30 | 67.87 | -1.20 | -0.30 | 1.50 |
| colic | 78.17 | 0.83 | 1.33 | -2.17 | 72.00 | 1.50 | 1.50 | -3.00 |
| colic.ORIG | 71.26 | 1.15 | 0.46 | -1.61 | 71.95 | -2.30 | 3.22 | -0.92 |
| credit-a | 82.01 | 0.13 | 0.9 | -1.06 | 82.14 | 1.59 | 1.19 | -2.78 |
| credit-g | 73.28 | 3.71 | -0.89 | -2.82 | 75.30 | 1.94 | 0.48 | -2.42 |
| diabetes | 77.18 | 0.51 | -0.26 | -0.26 | 76.92 | 0.38 | 0.77 | -1.15 |
| heart-c | 80.50 | -0.31 | 0.63 | -0.31 | 77.00 | -1.57 | 2.20 | -0.63 |
| heart-h | 78.23 | -0.80 | -0.68 | 1.36 | 78.23 | -0.68 | -0.68 | 1.36 |
| heart-statlog | 82.33 | -0.33 | -0.33 | 0.67 | 81.33 | -0.33 | 0.67 | -0.33 |
| mushroom | 95.85 | -1.46 | -0.26 | 1.72 | 99.85 | -0.04 | -0.01 | 0.05 |
| segment | 79.02 | -2.81 | -0.60 | 3.40 | 90.70 | -0.60 | 0.25 | 0.35 |
| sick | 99.22 | 0.04 | 0.04 | -0.08 | 99.01 | 0.00 | 0.00 | 0.00 |
| sonar | 70.15 | 1.49 | -1.49 | 0.00 | 74.13 | -2.49 | -2.49 | 4.98 |
| soybean | 89.65 | 0.57 | -2.41 | 1.84 | 95.18 | -0.28 | 0.57 | -0.28 |
| vehicle | 59.43 | -0.15 | -0.15 | 0.30 | 68.17 | 0.60 | -0.30 | -0.30 |
| vote | 88.99 | 0.16 | 0.16 | -0.31 | 93.87 | 0.00 | 0.00 | 0.00 |
| vowel | 40.96 | -9.04 | -3.67 | 12.71 | 58.19 | -4.24 | -1.98 | 6.21 |
| waveform | 86.68 | 0.24 | 0.24 | -0.47 | 88.49 | 0.96 | 0.08 | -1.04 |

The situation of MultiSourceTrAdaBoost algorithm is exactly like the TrAdaBoost. Theoretically, it will be helpful to upgrade target classifier with more labelled data in target domain. But on our 19 UCI datasets, the TrAdaboost performance does not vary vigorously with the small variability in the percentages of labelled data in target domain. It may because the strategy we use to re-weight the source domain instances have weakened the effectiveness of these instances with more labelled data exist in target domain.

Table 4.6: Accuracy comparison among different $l_p$ for Co-training

| Dataset | Naive Bayes | | | | SVM | | | |
|---|---|---|---|---|---|---|---|---|
| | Average | 5% | 10% | 20% | Average | 5% | 10% | 20% |
| balance-scale | 74.53 | -1.73 | 0.27 | 1.47 | 66.93 | -0.93 | -0.53 | 1.47 |
| breast-cancer | 69.67 | -4.80 | 2.40 | 2.40 | 73.18 | 0.42 | -1.11 | 0.69 |
| colic | 72.50 | -2.00 | 0.00 | 2.00 | 71.33 | -3.33 | 2.17 | 1.17 |
| colic.ORIG | 64.14 | -3.45 | 0.69 | 2.76 | 65.06 | 0.46 | -2.99 | 2.53 |
| credit-a | 79.50 | -1.72 | -0.93 | 2.65 | 73.15 | -3.31 | 0.26 | 3.04 |
| credit-g | 73.68 | -0.80 | -0.10 | 0.90 | 71.51 | -0.81 | 1.37 | -0.57 |
| diabetes | 69.62 | -0.38 | 1.15 | -0.77 | 69.36 | 1.03 | -0.90 | -0.13 |
| heart-c | 68.87 | -8.49 | 4.72 | 3.77 | 68.24 | -4.09 | -1.26 | 5.35 |
| heart-h | 72.79 | -1.36 | -1.36 | 2.72 | 71.43 | 0.00 | -1.02 | 1.02 |
| heart-statlog | 79.33 | -4.33 | 4.67 | -0.33 | 76.00 | 2.00 | 1.00 | -3.00 |
| mushroom | 94.78 | -1.01 | -0.02 | 1.03 | 97.90 | -0.65 | 0.12 | 0.52 |
| segment | 76.46 | -5.93 | 2.07 | 3.86 | 74.08 | -13.43 | 2.27 | 11.17 |
| sick | 94.38 | -0.53 | -0.53 | 1.07 | 92.98 | 0.00 | 0.00 | 0.00 |
| sonar | 57.21 | -1.99 | -1.99 | 3.98 | 64.68 | -3.48 | 1.00 | 2.49 |
| soybean | 61.84 | -14.18 | -0.14 | 14.33 | 65.67 | -12.48 | -3.97 | 16.45 |
| vehicle | 48.42 | -4.52 | -2.71 | 7.24 | 40.42 | -0.60 | -1.06 | 1.66 |
| vote | 87.89 | -0.16 | -1.10 | 1.26 | 91.67 | -1.10 | 0.31 | 0.79 |
| vowel | 45.95 | -11.49 | 1.22 | 10.26 | 34.75 | -5.08 | 4.24 | 0.85 |
| waveform | 87.25 | 0.43 | -0.10 | -0.33 | 74.82 | 0.37 | 0.26 | -0.63 |

Unlike transfer learning methods, semi-supervised learning algorithms achieve improvement with a higher $l_p$. From Table 4.6, the results show that the prediction accuracy increases on 15 datasets when $l_p$ change from 5% to 10% and 14 datasets when $l_p$ change from 10% to 20% with the base classifier Naive Bayes. The situation stays consistent when it comes to $l_p$ increasing to 20%. We already know that the initial classifier can make wrong prediction on the selected unlabelled instances, which result in the bad performance of semi-supervised learning approaches. Generally, the standard self-training and co-training should improve the performance if the selected unlabelled instances are labelled correctly. With more labelled data, the initial classifier becomes relatively robust. Thus the new classified unlabelled data can be useful for the iterative training process.

Finally, we compared these algorithms with the baseline which is built on the original labelled data using the corresponding base learner algorithm. Table 4.3 and Table 4.4 list how each algorithm with $l_p$ in the corresponding row wins in $w$ datasets, ties in $t$ datasets, and loses in $l$ datasets, comparing to the baseline. In transfer learning, TrAdaboost and MultiSourceTrAdaBoost behave quite well with a small $l_p$ on the 19 experimental datasets. With the increase of $l_p$, both TrAdaBoost and MultiSourceTrAdaBoost win on less datasets but still have some advantages. The impact of source domain instances should be weakened when the amount of labelled instances increase. However, for the semi-supervised learning methods, self-training cannot win corresponding baseline on most of the 19 datasets. Since co-training

utilizes two learners to maximize the agreement on the unlabelled data, its performance is better than self-training. Nevertheless, the superiority is not obvious to the baseline with $l_p = 20\%$ that co-training wins only 8 datasets using Naive Bayes and 9 datasets for SVM. It is evident that the semi-supervised learning method cannot perform better than the baseline.

# Chapter 5

# Conclusion

This chapter will give a conclusion for the thesis. First, a summary about our work will be made, followed by the evaluation of contributions for this thesis, The future works will be extended to be the end of this chapter.

## 5.1 Summary

Our study has investigated TrAdaBoost, MultiSourceTrAdaBoost, self-training and co-training algorithms on 19 UCI datasets. We first introduced the related background of two branches in machine learning: transfer learning and semi-supervised learning, which includes the definition, classification setting, and features. What they have in common is trying to leverage useful information from auxiliary data when the labeled data is insufficient to build a good classifier. Transfer learning, as the name suggests, transfer knowledge

from some related source domains, in order to help improve the learning in the target domain. Based on different settings for transfer, the transfer learning can be categorized into inductive transfer learning, transductive transfer learning, and unsupervised transfer learning. Besides, various approaches have been developed for transfer learning. Most of these methods can be concluded into instance transfer and feature-representation transfer. Examples are given for each category, such as TrAdaBoost in instance transfer, Structural Correspondence learning in feature-representation transfer. Furthermore, Negative transfer learning is also mentioned since how to avoid negative transfer is an indispensable issue for efficient knowledge transfer.

After that, we took a quick overview of semi-supervised learning. The basic feature of semi-supervised learning is to use both labeled and unlabeled data in the same distribution during training process, which is distinguished from other machine learning methods. This not only help to obtain supervision information from a small number of labeled instances, but also explore the underlying distribution of data utilizing a large amount of unlabeled instances [29]. Several typical and underlying algorithms in semi-supervised learning are introduced, namely, generative models, graph-based methods, self-training and co-training.

Chapter 3 mainly discussed about the methodology about our experiments. This part revolves around discussions of how to choose the algorithms as the representative, what strategy is used to compare the performance, how to pre-process the datasets, and how to handle the experiment settings. The

selected methods are specified from two aspects: notations and pseudo-code.

Chapter 4 showed the results and analysis of our experiments. The empirical comparison is conducted on 19 UCI datasets using the TrAdaBoost, MultiSourceTrAdaBoost, standard self-training and co-training with two base learner Naive Bayes and Support Vector Machine. The interesting thing is that although both transfer learning and semi-supervised learning seek help from extra data, the consequences varied due to the different kind auxiliary data they used respectively.

## 5.2 Contributions

Our study has two main contributions. Firstly the advantages and disadvantages are compared between transfer learning and semi-supervised learning in using auxiliary data. Up to now, research works are concentrated on creating new methods of using auxiliary data, which mainly derived from two branches in machine learning: transfer learning and semi-supervised learning. Many of these approaches are only applicable to some certain problem. No detailed work has been done to systematically analyse and compare these methods under the identical situations.

Second, we give a further investigation to the value of auxiliary data in training process. Our results show that, some of these semi-supervised learning methods with unlabeled auxiliary data do not outperform base classifier learning only from the labeled instances. On the other hand, some

transfer learning methods with labeled auxiliary data from a related domain perform quite well when the labeled data is scarce in target domain. However, with the increment of the labeled data in target domain, transfer learning methods may not be advantageous. Meanwhile, we notice that, despite the outperformance of single domain in transfer learning, MultiSourceTrAdaBoost is quite stable when the given percentage of labeled data is changing. On our 19 UCI experimental datasets, the results show that the labeled instances from a related domain can leverage more positive information compared to the unlabeled instances form the same domain.

In practical application, either the process of labelling is pricey or time-consuming, or sometimes the data updates so fast that they are easily outdated. Exploring information from auxiliary data attracts increasing attentions. At this point, our research provides useful help and reference for auxiliary data application .

## 5.3 Future Works

On our 19 UCI experimental datasets, negative transfer learning does not occur vigorously for TrAdaBoost and MultiSourceTrAdaBoost. However, it can probably happen when the source domain data contribute to hurt the performance of target classifier. In the future, more classifiers and models remain to be tested on more data for validity and generalizability of this study.

Moreover, according to the experimental results in this study, we can think about the combination of these two kind auxiliary data. For example, extending inductive transfer learning methods to semi-supervised learning and digging deeper into the relationship between semi-supervised learning and transfer learning.

# Bibliography


[1] Andrew Arnold, *A comparative study of methods for transductive transfer learning*, Seventh IEEE International Conference on Data Mining Workshops. (2007), 77--82.

[2] Machine Learning Group at the University of Waikato, *Weka 3.6.8*, http://www.cs.waikato.ac.nz/ml/weka/.

[3] A. Blum and S. Chawla, *Learning from labeled and unlabeled data using graph mincuts*, 2001, pp. 19--26.

[4] Avrim Blum and Tom Mitchell, *Combining labeled and unlabeled data with co-training*, Morgan Kaufmann Publishers, 1998, pp. 92--100.

[5] O. Chapelle, B. Scholkopf, and Eds. A. Zien, *Semi-supervised learning*, MA: MIT Press, Cambridge, 2006.

[6] A. Corduneanu and T. Jaakkola, *Stable mixing of complete and lncomplete lnformation*, (2001).

[7] W. Dai, G. Xue, Q. Yang, and Y. Yu, *Co-clustering based classification for out-of-domain documents*, Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007), 210--219.

[8] W. Dai, Q. Yang, G. Xue, and Y. Yu, *Boosting for transfer learning*, Int'l Conf. on Machine Learning (2007).

[9] ———, *Self-taught clustering*, Proceeding ICML '08 Proceedings of the 25th international conference on Machine learning (2008), 200--207.

[10] Peter Dayan, Maneesh Sahani, and Grgoire Deback, *Unsupervised learning*, In The MIT Encyclopedia of the Cognitive Sciences, The MIT Press, 1999.

[11] E. Eaton, M. desJardins, and T. Lane, *Modeling transfer relationships between learning tasks for improved inductive transfer*, Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases (2008), 317--332.

[12] A. Fujino, N. Ueda, and K. Saito, *A hybrid generative/disciminative approach to semi-supervised classifier design*, Proceeding AAAI'05 Proceedings of the 20th national conference on Artificial intelligence **2** (2005), 764--769.

[13] Y. Guo, X. Niu, and H. Zhang, *An extensive empirical study on semi-supervised learning*, Data Mining (ICDM),IEEE 10th International Conference on (2010), 186--195.

[14] Z. Guo, Z. Zhang, E. P. Xing, and C. Faloutsos, *Semi-supervised learning based on semiparametric regularization*, in Proceedings of the 2008 SIAM International Conference on Data Mining (2008), 132--142.

[15] R. McDonald J. Blitzer and F. Pereira, *Domain adaptation with structural correspondence learning*, Proceedings of the Conference on Empirical Methods in Natural Language Processing (2006), 120--128.

[16] Nilkola K. Kasabov, *Transductive support vecot machines and applications in bioinformatics for promoter recognition*, Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003. **1** (2003).

[17] M. White L. Xu and D. Schuurmans, *Optimal reverse prediction a unified perspective on supervised, unsupervised and semi-supervised learning*, Proceedings of 26th international conference on Machine learning (2009).

[18] S. Lee, V. Chatalbashev, D. Vickrey, and D. Koller, *Learning a meta-level prior for feature relevance from multiple related tasks*, Proceedings of the 24th international conference on Machine learning (2007), 289--406.

[19] X. Liao, X. Ya, and L. Carin, *Logistic regression with an auxiliary data source*, Proceeding ICML '05 Proceedings of the 22nd international conference on Machine learning (2005).

[20] W. Liu, J. Wang, and S. Fu, *Robust and scalable graph-based semisupervised learning*, Proceedings of the IEEE **100** (2012), 2624--2638.

[21] K. Nigam, A. Kachites Mccallum, S. Thrun, and T. Mitchell, *Text classification from labeled and unlabeled documents using em*, Machine Learning, 1999, pp. 103--134.

[22] Shaoning Pang, *Inductive vs transductive inference, global vs local models: Svm, tsvm, and svmt for gene expression classification problems*, Proceedings. 2004 IEEE International Joint Conference on Neural Networks, 2004. **2** (2004).

[23] John C. Platt, *Sequential minimal optimization: A fast algorithm for training support vector machines*, 1998.

[24] M.T. Rosenstein, Z. Marx, and L.P. Kaelbling, *To transfer or not not to transfer*, Proceedings of NIPS 2005 Workshop on Inductive Transfer: 10 Years Later (2005).

[25] J. P. Sinno and Q. Yang, *A survey on transfer learning*, Technical Report, Hong Kong University of Science and Technology (2008).

[26] M. Szummer and T. Jaakkola, *Bpartially labeled classification with markov random walks*, Advances in Neural Information Processing Systems **14** (2002), 945--952.

[27] L. Torrey and J. Shavlik, *Transfer learning*, IGI Global, 2009.

[28] V. Vapnik, *Statistical learning theory*, (1998).

[29] B. Wang, *Semi-supervised learning and opinion-oriented information extraction*, University of New Brunswick, Faculty of Computer Science, 2009.

[30] Z. Wang, Y. Song, and Y. Yu, *Transferred dimensionality reduction*, in Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases(ECML PKDD '08 ) (2008), 550--565.

[31] Wikipedia, *Wikipedia for kullback-leibler divergence*, http://en.wikipedia.org/wiki/Kullback-Leibler_divergence, April 2013.

[32] P. Wu and T. Dietterich, *Improving svm accuracy by training on auxiliary data sources*, ICML '04 Proceedings of the twenty-first international conference on Machine learning (2004).

[33] X.Wu, V.Kumar, J.R. Quinlan, J.Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A.F.M. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, and D. Steinberg, *Top 10 algorithms in data mining*, Knowledge and Information Systems **14** (2008), 1--37.

[34] Q. Yang and X. Wu, *10 challenging problems in data mining research*, Information Technology and Decision Making **5** (2006), 597--604.

[35] Y. Yao and G. Doretto, *Boosting for transfer learning with multiple sources*, 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010), 1855--1862.

[36] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf, *Learning with local and global consistency*, **16** (2004), 321--328.

[37] X. Zhu, *Semi-supervised learning literature survey*, (2008), 1530.

[38] X. Zhu, Z. Ghahramani, and J. D. Lafferty, *Semi-supervised learning using gaussian fields and harmonic functions*, [ in Proc. Int. Conf. Mach. Learn (2003), 912--919.

# Vita

Candidate's full name: Ao Cheng
University attended (with dates and degrees obtained): Huazhong University of Science and Technology, Bachelor of Computer Science, 2011