ROBUST DOMAIN ADAPTATION USING ACTIVE LEARNING

A Dissertation Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Kinjal Dhar Gupta

August 2016

ROBUST DOMAIN ADAPTATION USING ACTIVE LEARNING

Kinjal Dhar Gupta

APPROVED:

Dr. Ricardo Vilalta,
Department of Computer Science
University of Houston

Dr. Christoph Eick
Department of Computer Science

Dr. Guoning Chen
Department of Computer Science

Dr. Ashish Mahabal
Department of Astronomy
California Institute of Technology

Dean, College of Natural Sciences and Mathematics

Acknowledgements

I would like to express my sincere thanks and deepest gratitude to my adviser, Dr. Ricardo Vilalta, for being the most wonderful mentor I could have ever had. I would like to thank him for his undivided attention, constant cooperation, and extremely useful guidance in research and in real life throughout my PhD degree course. Without his mentorship, it would have been impossible for me to achieve the goal of completing my PhD studies. I would like to thank my parents Mr. Kalyan Dhar Gupta and Mrs Shibani Dhar Gupta, and my sister, Sharanya Dhar Gupta, for their constant love, care, support, and encouragement, without which I would never have been able to be at this place of my life. I would also like to thank all my committee members, Dr. Eick, Dr. Chen, and Dr. Mahabal for their valuable time and input to my research. I would like to thank Dr. Lucas Macri and Renuka Pampana for providing the pre-processed data used in this research. Last but not the least, a special thanks to all my friends and family members for their best wishes that I have kept receiving during the last couple of years.

ROBUST DOMAIN ADAPTATION USING ACTIVE LEARNING

An Abstract of a Dissertation

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Kinjal Dhar Gupta

August 2016

Abstract

Traditional machine learning algorithms assume training and test datasets are generated from the same underlying distribution, which is not true for most real-world datasets. As a result, a model trained on the training dataset fails to produce good classification accuracy on the test dataset. One way to mitigate this problem is to use domain adaptation techniques; these techniques build a new model on the unlabeled test dataset (target dataset) by transferring information from a related but labeled training dataset, (source dataset) even when their underlying distributions are different. One other important issue is that in domain adaptation, But there is no allowance for obtaining class labels on the test dataset during the training phase. This issue can be handled by active learning techniques that assume the existence of a budget that can be used to label instances on the target domain. Active learning finds the most informative instances of the test dataset that can be labeled by the expert to get a better classification accuracy on the unlabeled test dataset.

The goal of this research is to build an optimal classifier on the target dataset by using information related to model complexity. We propose a novel domain adaptation technique using active learning to find the optimal value of a parameter of a class of models that yields the best classifier on the target dataset without assuming the equivalence of the class-conditional probabilities across the domains, unlike other domain adaptation methods. This research also proposes a novel data-alignment technique that allows the use of the source model directly on the target if the distributions differ due to a linear shift, thus avoiding building a complete new classifier on the target domain. Empirical results show that our methods yield better classification accuracy than the state-of-art methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Machine Learning

Machine learning is a sub-field of computer science that deals with the teaching of a system to improve its performance over various tasks by gaining new experience. Two basic types of machine learning methods that are related to my research are *supervised learning* and *unsupervised learning*.

1.1.1 Supervised Learning

Supervised learning is a process that builds an optimized function or a model that maps a certain set of given inputs to some desired outputs. These outputs are in the form of labels or values already assigned by an expert (supervisor). The input

is a set of entities represented as data points in a space defined by features, each of which denotes a characteristic of the data points. The set of given input data points in the learning phase is called the *training dataset*. After the learning process ends, the model is generally tested on a new dataset, also known as the *testing dataset*, that has not been used for the training purpose. The error generated by the model is equal to the average of all the errors or discrepancies contributed by the each test data point as predicted by the model.

1.1.1.1 Types of Supervised Learning

There are two types of supervised learning: *classification* and *regression*. *Classification* is a supervised learning technique that maps the input data points to the desired output that are generated from the distribution of a discrete random variable. The learning process divides the data space into regions, each region belonging to a different class. The intrinsic assumption is that all points lying within a region can be classified as the same class as assigned by the model. Thus with a given small number of labeled data points, the algorithm is able to predict the class of any unknown data point that is sampled from the same underlying distribution. Some of the well known classification learning methods include Neural Networks, Naive Bayes, K-Nearest Neighbor, Decision Trees, Logistic Regression, and Support Vector Machines.

Regression is a supervised learning process that builds a function that maps each input data point to a desired output value generated from the distribution of a

continuous random variable.

1.1.2 Unsupervised Learning

Unsupervised learning differs from *supervised learning* by the fact that there are no desired outputs available. However in most cases the number of categories in which the data points should be divided is known. Thus the goal of unsupervised learning is to form clusters on the training data such that the data points belonging to the same category will be included in the same cluster. Some of the popular unsupervised learning methods include K-Means clustering, Self-Organizing Maps and Expectation-Maximization Algorithm.

1.2 Motivation

Traditional machine learning algorithms assume that the training and the test datasets are sampled from the same underlying distribution, which is not true for most real-world datasets. As a result, a model trained on the training dataset fails to produce good classification accuracy on the test dataset. In addition, it is often seen that labeling a new test dataset is very expensive.

So when a new unlabeled dataset is to be classified, there are two options: (1) to choose a labeled but related dataset, also called the *source* dataset, and transfer information from it to the new dataset, also called the *target* dataset, or (2) try to

obtain labels for some of the instances of the new dataset to build a model on it. The former method is known as *domain adaptation*[1][2][3][4] and the latter can be handled by *active learning*[5].

1.2.1 Domain Adaptation

Domain Adaptation is the process of building a new model on the target dataset by using some information from the source dataset. If the target has very few labels it is called a supervised domain adaptation and if there exists no target labels, it is called unsupervised domain adaptation. However, there is no allowance of extra target labels than those already available.

1.2.2 Active Learning

On the other hand *active learning* assumes that there exists a small budget to label some instances of the target data. Its goal is to find a sample of most informative examples of the target dataset; this process involves an expert during the learning phase. A typical active learning procedure selects one or a batch of instances from the target dataset and query their labels from an expert, then build a model on the same dataset to find the next set of most informative examples that need to be queried. The process continues until the budget runs out.

1.2.3 Disadvantages of Domain Adaptation

Domain adaptation suffers from certain disadvantages. In domain adaptation it is assumed that there exists at least one model in either the current data space or in some projected space that can classify both source and target perfectly. This is called the covariate shift[6]. This does not hold true in case of many real world datasets. Also these methods use weighted instances or transformed features from the source domain to build a model on the target domain. As a result the model formed on the target domain gets biased towards the source. Active learning reduces this bias by allowing us to incorporate some labeled instances from the target dataset. This led us to the idea of using active learning with domain adaptation.

1.3 Research Goal

This research combines domain adaptation with active learning to improve the accuracy of a model built on the target dataset. The motivation behind using domain adaptation is to utilize as much information as possible from a related but labeled dataset. The reason for using active learning is to reduce the bias induced by the source dataset during domain adaptation.

The aim of this research is to build an optimal classifier on the target dataset by using information related to model complexity. We propose a novel domain adaptation technique using active learning to find the optimal value of a parameter of a class of models that yields the best classifier on the target dataset without assuming the

equivalence of the class-conditional probabilities across the domains. We combine the prior distribution of the parameter, obtained from the source, with its likelihood obtained from a sample of most informative labeled instances of the target dataset. This research also proposes a novel data alignment technique that allows the use of the source model directly on the target if the distributions differ due to a linear shift, thus avoiding building a complete new classifier on the target domain.

1.4 Dissertation Layout

The layout of my dissertation is as follows :

- **Chapter 2** presents a background of various domain adaptation and active learning techniques and those which use both domain adaptation and active learning.
- **Chapter 3** describes the methodologies used in this research.
- **Chapter 4** includes the results of the experiments of this research.
- **Chapter 5** is a summary of contributions and our conclusion from this research.

Chapter 2

Background

2.1 Preliminary Concepts in Statistical Learning

Consider an n-component vector-valued random variable $(X_1, X_2, , X_n)$, where each X_i represents an attribute or feature; the space of all possible feature vectors is called the input space X. Consider also a set $w_1, w_2, , w_k$ corresponding to the possible classes; the space of all possible classes is called the output space W. A classifier typically receives a set of training examples as input from a source domain, $T_{tr} = (x_i, w_i)_{i=1}^{p}$, where $\mathbf{x} = (x_1, x_2, , x_n)$ is a vector in the input space, and w is a value in the (discrete) output space. We assume the training or source sample T_{tr} consists of independently and identically distributed (i.i.d.) examples obtained according to a fixed but unknown joint probability distribution, $P_{tr}(x, w)$, in the input-output space $X \times W$. The outcome of the classifier is a hypothesis or function $f(x|\theta)$

(parameterized by θ) mapping the input space to the output space, $f : X \rightarrow W$. We commonly choose the hypothesis that minimizes the expected value of a loss function $L(w, (x|\theta))$, also known as the risk:

$$R(\theta, P(x, w)) = E \sim P[L(w, f(x|\theta))] \tag{2.1}$$

where we typically adopt the zero-one loss function $L(w, f(x|\theta)) = 1_{\{x|w=f(x|\theta)\}}(x)$; $1(.)$ is an indicator function. We will also assume a testing sample from a target domain, $T_{te} = \{x_i\}_{i=1}^{q}$, that consists of i.i.d. examples obtained from the marginal distribution $P_{te}(x)$ according to a different joint distribution, $P_{te}(x, w)$, over $X \times W$.

2.2 Reasons for Dataset Shift

Traditional machine learning algorithms assume that the joint distributions $P_{tr}(x, w)$ and $P_{te}(x, w)$ across the source (training) and the target(testing) domains respectively are the same. But in real-world scenarios this does not hold true. The result is $P_{tr}(x, w) \neq P_{te}(x, w)$, which is known as a dataset shift. There can be various reasons for this shift:

- **Simple covariate shift**: The joint distributions $P_{tr}(x, w)$ and $P_{te}(x, w)$ can be modeled as a product of their priors and posterior such as $P_{tr}(x, w) = P_{tr}(w|x)P_{tr}(x)$ and $P_{te}(x, w) = P_{te}(w|x)P_{te}(x)$. We say that a simple covariate shift has occurred when $P_{tr}(w|x) = P_{te}(w|x)$ and $P_{tr}(x) \neq P_{te}(x)$, i.e., the difference between the joint distributions is attributed to the change in the covariates $\mathbf{x}$. This is the basic assumption of most domain adaptation methods.

- **Prior probability shift**: A prior probability shift occurs when the distribution over the the source and the target classes are different, i.e., $P_{tr}(w) \neq P_{te}(w)$ where the joint distributions $P_{tr}(x, w)$ and $P_{te}(x, w)$ are modeled as $P_{tr}(x, w) = P_{tr}(x|w)P_{tr}(w)$ and $P_{te}(x, w) = P_{te}(x|w)P_{te}(w)$ respectively.

- **Sample selection bias**: When the change in the joint distributions of the source and the target domains occur due to an unknown sample rejection process, we call it a sample selection bias.

- **Imbalanced data**: When one or more classes in a dataset are very rare, as compared to others in the same dataset, we call it an *imbalanced dataset*. The imbalanced dataset shift problem is mainly caused due to a design issue. If the goal of a problem is to detect rare events, many times the dataset is balanced by discarding instances of the major class, to give more importance to the rare-event training instances. This results in a disparity between the source and the target distributions. For a conditional modeling case, imbalanced data problem can be regarded as a sample selection problem with a known selection bias.

- **Domain shift**: When there is a change in measurements across the source and the target domains, we call it a domain shift problem. For example, if the source data are measured in one particular system of units and the target data are measured in another, we say that a domain shift has occurred.

- **Source component shift**: When the observed data is made up of data from

different sources with different characteristics, and the proportions of those sources vary across the training and the test distributions, we call it a source component shift.

In this research we deal with two scenarios :

1. A scenario when a covariate shift occurs between the source and the target distributions due to a linear shift of the marginals $P_{tr}(x)$ and $P_{te}(x)$ but the conditionals $P_{tr}(w|x)$ and $P_{te}(w|x)$ remain the same, i.e., $P_{tr}(w|x) = P_{te}(w|x)$ and $P_{tr}(x) \neq P_{te}(x)$.

2. A much more generic scenario when a covariate shift occurs between the source and the target distributions but we cannot assume that the conditionals $P_{tr}(w|x)$ and $P_{te}(w|x)$ are the same, i.e. , $P_{tr}(w|x) \neq P_{te}(w|x)$ and $P_{tr}(x) \neq P_{te}(x)$.

2.3 Domain Adaptation

The dataset shift problem is handled in the literature by domain adaptation methods. Domain adaptation has gained much attention recently, mainly due to the pervasive character of problems where distributions change over time; it assumes the learning task remains constant (e.g., classifying variable Cepheid stars into two classes), but the marginal and class posterior distributions between source and target domain may differ (as opposed to transfer learning[8], where tasks can exhibit different input representations, i.e., different input spaces).

Domain adaptation methods can be broadly grouped into three categories as follows:

- **Instance-based domain adaptation**: The goal of instance-based domain adaptation is to identify the instances in the source that are more related to the target and give them more importance while calculating the risk. A new model is built on the weighted source data which is used as a classifier on the target. Several instance-based domain adaptation methods have been proposed such as finding the weight ratio $\frac{P_{te}}{P_{tr}}$ to weigh the instances, as direct estimation covariate shift[6][7], direct estimation using covariate shift assumption[9], solving sample selection bias problem[10], and kernel mean matching method[11]. The underlying assumption of instance based methods is that the target must be in support of the source.

- **Feature-based domain adaptation**: In some cases the source and the target may be overlapping, but it is not possible to determine a sample of instances of the source that are more important than the others. In that case, feature-based methods are used. The idea is to project both the source and the target datasets into a common feature space such that they can be aligned. A new model is built on the transformed source data that is used as the classifier on the target. Some of the proposed feature-based methods include structural corresponding learning[14], subspace alignment method[15] and others[20][21][39].

- **Iterative-based domain adaptation**: In iterative-based domain adaptation, a model is built on the target in an iterative way. In each iteration some

instances of the target domain are labeled depending upon their proximity to the margin and a model is built on them. Also some of the source instances are discarded at every iteration that are lie far away from the labeled target instances.

We explore these three types of domain adaptation in detail in the following sections of this chapter.

2.3.1 Instance-based Domain Adaptation

Instance-weighting is one of the most common domain adaptation techniques. Let M be the class of models for which we wish to obtain the best classifier M^*. Let $l(x, w, M)$ be the loss function. Our objective is to find M^* such that :

$$M^* = \arg\min_{M^* \in M} \sum_{(x,w) \in X \times W} P(x, w) l(x, w, M). \tag{2.2}$$

As $P(x, w)$ is unknown we use the sampled dataset $P(x, w)^*$ to approximate $P(x, w)$. Thus we minimize the following empirical risk to obtain M^* such that:

$$M^* = \arg\min_{M^* \in M} \sum_{(x,w) \in X \times W} P(x, w)^* l(x, w, M). \tag{2.3}$$

or

$$M^* = \arg\min_{M^* \in M} \sum_{i}^{N} l(x_i, w_i, M) \tag{2.4}$$

where N is the number of instances (x_i, w_i) in the data.

Similarly we can say that our main goal is to find M^* on target distribution $P_{te}(x, w)$ such that :

$$M^* = \arg\min_{M^* \in M} \sum_{(x,w) \in X \times W} P_{te}(x, w) l(x, w, M). \tag{2.5}$$

But since we have labeled instances only in the source dataset, we re-write the equation 2.5 in the following way:

$$M^* = \arg\min_{M^* \in M} \sum_{(x,w) \in X \times W} \frac{P_{te}(x, w)}{P_{tr}(x, w)} P_{tr}(x, w) l(x, w, M). \tag{2.6}$$

or

$$M^* = \arg\min_{M^* \in M} \sum_{(x,w) \in X \times W} \frac{P_{te}(x, w)}{P_{tr}(x, w)} P_{tr}(x, w)^* l(x, w, M). \tag{2.7}$$

or

$$M^* = \arg\min_{M^* \in M} \sum_{i=1}^{N} \frac{P_{te}(x_i^{tr}, w_i^{tr})}{P_{tr}(x_i^{tr}, w_i^{tr})} l(x_i^{tr}, w_i^{tr}, M). \tag{2.8}$$

Thus we see that weighting the loss for the instance (x_i^{tr}, w_i^{tr}) with $\frac{P_{te}(x_i^{tr}, w_i^{tr})}{P_{tr}(x_i^{tr}, w_i^{tr})}$ gives us the solution for domain adaptation. However it is not possible to calculate $\frac{P_{te}(x_i, w_i)}{P_{tr}(x_i, w_i)}$ for a pair (x_i, w_i), since we do not have enough labeled instances in the target domain. To find a solution to this problem we explore two cases: class imbalance and covariate shift.

2.3.1.1 Class Imbalance

In a class-imbalance problem, it is assumed that the conditional distribution of X is the same for source and target domains, i.e., $P_{tr}(x|w = w_i)$=$P_{te}(x|w = w_i)$ for all w_i

$\in W$ but $P_{tr}(w) \neq P_{te}(w)$ as mentioned in [30]. Using Bayes formula, we can write

$$\frac{P_{te}(x,w)}{P_{tr}(x,w)} = \frac{P_{te}(w)}{P_{tr}(w)} \frac{P_{te}(x|w)}{P_{tr}(x|w)} \tag{2.9}$$

or

$$\frac{P_{te}(x,w)}{P_{tr}(x,w)} = \frac{P_{te}(w)}{P_{tr}(w)} \tag{2.10}$$

To solve the class-imbalance problem we obtain $\frac{P_{te}(w)}{P_{tr}(w)}$ as given in [31]. Alternatively, we can do a stratified sampling on the training instances from the source domain such that class distribution across two domains match. In such methods, under-represented classes are over-sampled, and over-represented classes are under-sampled[32][33][34]. For logistic regression functions it can be shown that

$$P_{te}(w|x) = \frac{g(w)P_{tr}(w|x)}{\sum_{w=1}^{c} g(w)P_{tr}(w|x)} \tag{2.11}$$

where c is the number of classes and

$$g(w) = \frac{P_{te}(w)}{P_{tr}(w)}$$

Other methods have been proposed as well for those learning algorithms that do not model $P(w|x) = P(w|x)$ directly[35]. The class distribution of the target domain sometimes is known[31] and sometimes it is estimated using EM algorithm[36].

2.3.1.2 Covariate Shift

Covariate shift is a common assumption made in domain adaptation. It is assumed that the marginal distributions differ, whereas the class conditionals remain same

across source and target domains, i.e., $P_{tr}(w|x) = P_{te}(w|x)$ but $P_{tr}(x) \neq P_{te}(x)$. This means that there exists an optimal model that can classify both target and source data perfectly. Ideally a model built on the source domain should be able to classify the target dataset - but that does not happen most of the times. The reason is the use of misspecified models[6], which means that the model built on the source never produces the optimal model, i.e., there exists no $M^* \in M$ such that $P(w|x = x_i, M) = P(w|x = x_i)$.

Under covariate shift, we can write the equation

$$\frac{P_{te}(x, w)}{P_{tr}(x, w)} = \frac{P_{te}(x)}{P_{tr}(x)} \frac{P_{te}(w|x)}{P_{tr}(w|x)} \tag{2.12}$$

or

$$\frac{P_{te}(x, w)}{P_{tr}(x, w)} = \frac{P_{te}(x)}{P_{tr}(x)} \tag{2.13}$$

One way [6] to minimize the covariate shift is to re-weight the log likelihood of each source instance (x_i, w_i) with the ratio $\frac{P_{te}(x)}{P_{tr}(x)}$ using maximum likelihood estimation. But the assumption is that $P_{te}(x)$ must be in the support of $P_{tr}(x)$. One such method is to build density estimators for source and target domains and estimate the ratio between them[9]. Another way is to learn the weights discriminatively[7]. This method assumes that $\frac{P_{te}(x_i)}{P_{tr}(x_i)} \propto \frac{1}{p(s=1|x,M)}$. The source and target is labeled as 0 and 1 respectively and a classifier M^* is built on this combined source and target data to compute the weights $g(x_i) = \frac{1}{p(s=1|x_i,M^*)}$.

One of the most-popular instance-based domain adaptation methods is *kernel*

mean matching. The goal of this method is to minimize the maximum mean discrepancy between two distributions by projecting the distribution in reproducing Hilbert kernel space (RKHS). The maximum mean discrepancy is given by the equation :

$$MMD(T_{tr}, T_{te}) = \| \frac{1}{m_{tr}} \sum_{i-1}^{m_{tr}} \phi(x_i^{tr}) - \frac{1}{m_{te}} \sum_{i-1}^{m_{te}} \phi(x_i^{te}) \|_H \tag{2.14}$$

The objective is:

$$min_\beta \| \frac{1}{m_{tr}} \sum_{i-1}^{m_{tr}} \beta(x_i^{tr}) \phi(x_i^{tr}) - \frac{1}{m_{te}} \sum_{i-1}^{m_{te}} \phi(x_i^{te}) \|_H \tag{2.15}$$

such that

$$\beta(x_i^{tr}) \in [0, B] \; and \; |\frac{1}{m_{tr}} \sum_{i-1}^{m_{tr}} \beta(x_i^{tr}) - 1| < \epsilon$$

The β values acts as the weights[10] for the source examples. The error of each source example is multiplied with the corresponding β value while calculating the risk of the training model, thus giving appropriate weights to the relevant source examples.

2.3.2 Feature-based Domain Adaptation

The difference in the distributions of the source and the target domains can also arise due to the change in the features. Such a change of representation of X can affect both the marginal distribution $P(x)$ and the conditional distribution $P(w|x)$. In that case, we assume that under some change in representation the joint distributions of the source and the target datasets will be equal. Let $t : X \rightarrow Z$ denote a transformation function that transforms an observation x represented in the original form into another form $z_i = t(x) \in Z$. We define variable Z such

that it satisfies $P(z) = \sum_{x \in X, t(x)=z} P(x)$ The joint distribution of Z and W is then $P(z, w) = \sum_{x \in X, t(x)=z} P(x, w)$. Thus if we can find t for the source and target domains, we can say that $P_{tr}(z, w) = P_{te}(x, w)$. Since now the distributions are the same, the optimal model built on transformed source data will also be the model on target data.

As mentioned in [37], it is to be noted that with a change of representation, the entropy of $P(w|z)$ is likely to increase from the entropy of $P(w|x)$, because Z is usually a simpler representation of the observation than X, and thus encodes less information. In other words, the Bayes error rate usually increases under a change of representation. Therefore, the criteria for good transformation functions include not only the distance between the induced distributions $P_{tr}(z, w)$ and $P_{te}(z, w)$ but also the amount of increment of the Bayes error rate.

The effect of representation change for domain adaptation was first formally analyzed in [1]. They proved a generalization bound for domain adaptation that is dependent on the distance between the induced $P_{tr}(z, w)$ and $P_{te}(z, w)$. One feature-based domain adaptation technique is *metric learning*[13][12]. It uses the Mahalanobis distance as a distance metric and introduces pair-wise constraints to determine the similarity between source and target instances. But it requires target labels. Another method that uses distance metric along with target labels is described in [38]. They propose one of the most common transformation used in feature based methods - *feature subset selection*. In this method the criterion of selecting features is to minimize an approximated distance function between the distributions in the two

domains.

There are many unsupervised domain adaptation techniques that use feature subset selection. One such method is *structural corresponding learning*[14]. It is widely used in sentiment analysis and deals with bag of words. It chooses K pivot features, which are frequent words in both domains, and are highly correlated with labels. It learns K classifiers to predict pivot features from remaining features. For each feature it adds K new features and represents source and target data with these features. It applies Principal Component Analysis (PCA) on source and target dataset to get a lower dimensional representation and learns a classifier on this representation.

One of the simplest and most common feature subset selection is *subspace alignment* method[15]. It applies PCA on both source and target and choose d eigenvectors for which it obtains the maximum accuracy on the source domain. It transforms the source subspace to align with the target subspace using a *subspace alignment matrix*. Once the source and target are aligned, it builds the classifier on the source data to classify the target. Algorithm 1 shows the steps of subspace alignment.

Algorithm 1 Subspace Alignment Algorithm

Input : Labeled Source data T_{tr}, unlabeled Target data T_{te} , Subspace dimension d

Output : Classifier M_T on target data.

1: $T^*_{tr} \leftarrow PCA(T_{tr}, d)$ (source subspace defined by d eigenvectors)
2: $T^*_{te} \leftarrow PCA(T_{te}, d)$ (target subspace defined by d eigenvectors)
3: $Z = T^*_{tr}(T^*_{tr})^T T^*_{te}$ (aligning source dataset with target)
4: $T^a_{tr} = T^*_{tr} Z$ (source data in aligned space)
5: $T^a_{te} = T_{te} T^*_{te}$ (target data in aligned space)
6: Build a classifier M_T on T^a_{tr}

Here $(T_{tr}^*)^T T_{te}^*$ corresponds to subspace alignment matrix and $(T_{tr}^*)^T$ is the transpose of T_{tr}^*.

2.3.3 Iterative-based Domain Adaptation

Iterative-based methods builds the model iteratively on the target dataset by using the information of source data. One of the most popular iterative-based algorithm is DASVM[16]. The idea is to assign pseudo-labels to target instances that are closest to the margin of the the classifier and fall within it, and discard those instances that are far from the boundary in every iteration. The algorithm then builds a model on combined remaining labeled source instances and target instances with pseudo-labels. The algorithm stops when there are no new target instances that are to be labeled.

2.3.4 Other Domain Adaptation Methods

The problem of domain adaptation has been attacked from various other angles: by proving error bounds as a function of empirical error and the distance between source and target distributions[1][2][22]; as part of multitask learning by finding a common representation using spectral functions[23]; within a co-training framework where target vectors are incorporated into the source training set based on confidence[24]; and by using regularization terms to learn models that perform well on both source and target domains[26].

In this research, we address two scenarios. The first scenario assumes the existence of a covariate shift, and shift between the priors is due to a linear shift. Different from previous work in our strategy is to shift the target data to achieve a close alignment with the source data[17][18][19]. By making both sets as similar as possible, we expect the source model to generalize well on the target domain. We do not use any existing feature-based methods and thus our algorithm does not incur any loss of information or increase in entropy. The second scenario does not assume the existence of the covariate shift, nor does it assume that the source and target have shared support. Thus most of the domain adaptation methods become non-applicable. We use a different approach, as described in section 3.3, to solve this problem.

2.4 Active Learning

Active learning[5][42] is a technique that assumes the existence of a budget to label some instances of an unlabeled target dataset and aims to build an optimal classifier on the unlabeled dataset by finding the most informative examples in the dataset iteratively and querying their labels from an expert. The active involvement of the expert leads to the convergence of the process to a better learning model compared to passive machine learning techniques. We discuss some sampling scenarios and sampling strategies in the following sections.

2.4.1 Sampling Scenarios

There are three main sampling scenarios for active learning. They are described as follows :

- **Membership query synthesis**: Membership query synthesis[40] is one of the first investigated area of active learning. In this setting, the active learner generates the queries *de novo*, without sampling from any existing underlying distribution. Some of its application include predicting the absolute coordinates of a robot hand given the joint angles of its mechanical arm as inputs[41], and 'scientific robots' to execute a series of autonomous biological experiments[43].

- **Stream-based selective sampling**: Another sampling scenario is stream-based selective sampling[45][44]. In this scenario, an instance is first sampled from the actual distribution and then the learner decides to query it or not. The assumption is that the sampling an unlabeled instance is free or inexpensive. This method is also known as sequential active learning as the instances are sampled in sequence, one by one, as the learner decides whether to query it or discard it. For a uniform input distribution, stream-based selective sampling behaves like membership query synthesis.

  The decision to query the sample or not can be designed in various ways. One way to decide that is to check how informative the sample is and make a random biased decision such that more informative instances are more likely to be queried[46]. Another approach is to find the region of uncertainty and only

query if the sample belongs to that region[45]. A common way to achieve this is to set a threshold for the informativeness of the instance and query it only when its information is above this threshold. A more-robust approach is to find a set of classifiers in a region that is unexplored or unknown to the model class that would be consistent with the labeled training set. This region is known as the version space. The idea is to query the instance if any two or more classifiers disagree in their labels while classifying the instance. This process is more complex as the region has to be maintained after every sampled instance has been queried. Some of the applications of stream-based selective sampling include part-of-speech tagging[46], sensor scheduling[47], learning ranking functions for information retrieval[48] and word-sense disambiguation[49].

- **Pool-based sampling**: Pool-based sampling[27] assumes that we have a small labeled dataset and a large pool of unlabeled dataset that is fixed. The learner chooses to query the most informative instance from the pool and add it to the set of labeled instances. The difference between pool-based and stream-based sampling is that the former has access to the entire pool to choose the most informative instances, whereas the latter can only view one instance at a time. Some of the applications of pool-based sampling include text classification[27][50][51][52], information extraction[53][54], image classification and retrieval[51][55], and video classification and retrieval[56][57], speech recognition[58], and cancer diagnosis[59].

2.4.2 Query Strategy Frameworks

All active learning scenarios involve evaluating the informativeness of unlabeled instances. We explore only one such strategy in detail, that is related to this research- *uncertainty sampling*.

2.4.2.1 Uncertainty Sampling

Uncertainty sampling is one of the most common query strategies[28]. In this framework, the instance is queried based on its measure of uncertainty. The amount of uncertainty can be measured in three ways :

- **Least-confidence sampling**: The learner queries the instance that is the least confident:

  $$x^*_{LC} = \arg\max_x 1 - P_\theta(w^*|x).$$

  where $w^* = \arg\max_w P_\theta(w|x)$. The disadvantage of this strategy is that it does not incorporate information from other classes that are not the most probable ones.

- **Margin sampling**: Margin sampling[29] overcomes the disadvantage of least confident sampling by querying the instance that has the lowest margin. The margin is defined by the difference in the probability of the two most probable classes for a given instance. Thus,

  $$x^*_M = \arg\min_x P_\theta(w^*_1|x) - P_\theta(w^*_2|x).$$

where w_1^* and w_2^* are the first and second most probable classes for x.

- **Sampling using entropy**: The third sampling strategy involves querying using entropy[60] as the uncertainty measure:

$$x_H^* = \arg\max_x \sum_i^c P_\theta(w_i|x) log P_\theta(w_i|x).$$

where c is the number of distinct class labels.

The choice of uncertainty sampling strategy is application dependent. Other query strategies include query-by-committee (QBC) algorithm[61], expected gradient length (EGL) approach[62], expected error reduction[63], variance reduction[42], and density-weighted methods[42]. More details of its application can be found in [42]. In this research we use *pool-based*[27] active learning with *margin sampling*[29] as our query strategy.

2.5 Active Learning in Domain Adaptation

In recent years, there has been much research on combining active learning with domain adaptation. Some existing active learning techniques in domain adaptation use query-by-committee approach[64]. But this approach assumes that the classes are separable. On the other hand other approaches[65][66] focus on using the source model to train the examples from the target domain that are similar to the source domain, thus increasing the pool of queries without incurring a cost for labeling these examples. The model is built iteratively on this pool that increases after each

iteration. They also propose a convergence criterion of the algorithm based on the sampling techniques of the queries[65]. There are other algorithms [66][67] that use the source model as the initial model of their algorithm instead of any random model built on the target domain. However these methods do not check for redundancies in the queries.

Our method differs from other methods in that we transfer the optimal model complexity of the source dataset to build models on the target domain during active learning, and then use the queried labeled instances to find an optimal parameter value on the target dataset. We do not transfer the information of the instances in the source domain directly to the target. We explain our methodology in detail in section 3.3.

Chapter 3

Method

This section describes our proposed method in detail. It is often seen in real world datasets that the training and test datasets differ in their distributions considerably. As a result, a model built on the training dataset cannot be used directly on the test dataset without doing some necessary adjustments. To overcome this problem researchers use domain adaptation.

However most domain adaptation methods assume that existence of covariate shift between the source(train) and the target(test) domains, which may not hold true for most real world datasets. Also, these methods use weighted instances or transformed features from the source domain to build a model on the target domain. As a result, the model formed on the target domain is biased towards the source. Active learning reduces this bias by allowing us to incorporate some labeled instances from the target dataset. This led us to the idea of using active learning with domain

adaptation.

The goal of this research is to design an algorithm that builds an optimal classifier on the target domain without making the covariate shift assumption, i.e., there exists a model that can classify both source and target perfectly. Thus we cannot claim that the model built using any sample of the source dataset can yield optimal classification accuracy on the target dataset. However, it can be assumed that since both source and target are related, the distribution of complexity of the models across the two datasets will be similar. But this does not guarantee that the optimal model parameters in the source domain will be equal to the optimal model parameters in the target domain. Thus our main goal is to find the optimal parameter for the class of models that would generate the optimal classifier on the target domain. We use the optimal parameter of the class of models on the source to build models on the target domain through an active learning process. When we exhaust the budget of querying the target instances, we proceed to find the optimal value of the parameter on the target dataset for the same class of models using the labeled target instances and the prior distribution of the parameter obtained from the source dataset.

One aspect of any domain adaptation method is that it always builds a new model on the target dataset. In this research we address another scenario where the source and the target distributions differ only by a linear shift. We aim to align the datasets, and use the source model directly, thus avoiding building a new model on the target domain. Both these scenarios are described as follows:

1. For the following sections we will refer to as first scenario, the assumption that

the covariate shift occurs between the source and the target distributions due to a linear shift of the marginals $P_{tr}(x)$ and $P_{te}(x)$ but the conditionals $P_{tr}(w|x)$ and $P_{te}(w|x)$ remain the same, i.e. , $P_{tr}(w|x)$=$P_{te}(w|x)$ and $P_{tr}(x) \neq P_{te}(x)$. It is very specific in nature but can be applied in domains where there exists a shift between the train and the test datasets due to the nature of acquiring the data.

2. The second scenario, which is the main goal of this research, is much more generic than the previous method, as it does not have covariate shift assumption between the source and the target distributions. Thus, for this scenario, $P_{tr}(w|x) \neq P_{te}(w|x)$ and $P_{tr}(x) \neq P_{te}(x)$. However, we do assume that the prior distribution of the complexity of the classifiers on these two domains are the same.

In the case of scenario 1 we introduce a novel method[17][18] to estimate the linear shift instead of using any existing domain adaptation methods. In the case of scenario 2, we first estimate the prior of the complexity of the model from the source dataset. Then we use active learning to estimate the posterior distribution of the complexity of the model in the target dataset. We discuss the scenarios in detail in sections 3.1 and 3.3 respectively.

3.1 A Novel Data-Alignment Method

In this section we describe the scenario 1 which is specific to the assumption that the change in prior distribution of the source and target domain is due to a linear shift in the data. The goal is to find the shift across various features and align the target dataset with the source dataset so that the source classifier can be directly used on the target dataset for a better classification accuracy, as compared to simply using it directly. The advantage of this method is that it eradicates the necessity to build a complete new model on the target dataset.

The problem we address is characterized by an original source domain where class labels abound, and by a target domain with no class labels. We show how the target domain can still be learned by assuming the difference in the joint input-output distribution is mainly due to a systematic shift of sample points. We illustrate an example in the context of Cepheid variable star classification, where we wish to discriminate stars according to their pulsation mode(s); specifically, we focus on the two most abundant classes, which pulsate in the fundamental and first-overtone modes. Such classification can in fact be attained for nearby galaxies with high accuracy (e.g., Large Magellanic Cloud) under the assumption of class label availability. The high cost of manually labeling variable stars, however, suggests a different mode of operation where a predictive model obtained on a data set from a source galaxy T_{tr}, is later used on a test set from a target galaxy T_{te}. Such scenario is not straightforwardly attained, as shown in figure 3.1 (left), where the distribution of Cepheids in

the Large Magellanic Cloud LMC galaxy (source domain, top sample), deviates significantly from that of M33 galaxy (target domain, bottom sample). In this example, we employ two features only: apparent magnitude in the y-axis, and log period in the x-axis, but our solution is general and allows for a multi-variate representation. Both the offset along apparent magnitude, and the significant degree of sample bias, are mostly due to the fact that M33 is $\sim 16 \times$ farther than the LMC. At these greater distances, the shorter-period (i.e., less luminous) Cepheids fall below the detection threshold and longer-period (i.e., more luminous) stars are preferentially detected. This is what we refer to as a systematic shift, where all objects or events (i.e., sample points) are exposed to the same change, and thus we expect a similar displacement throughout the entire target distribution.

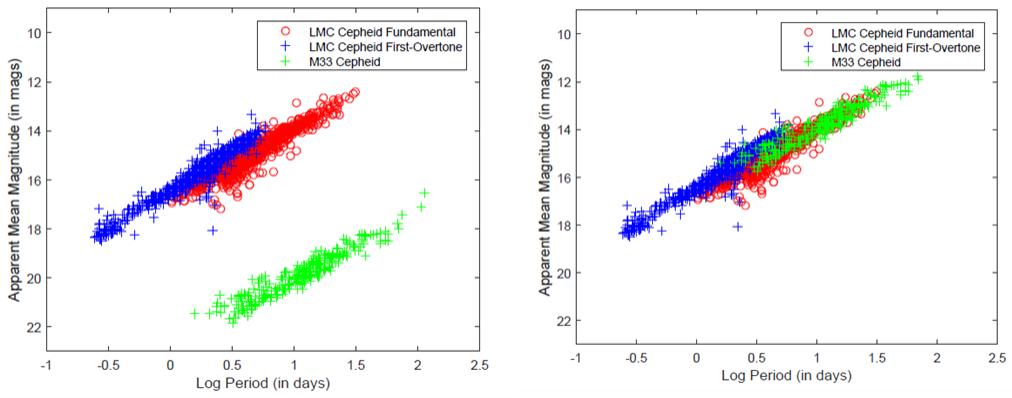

Figure 3.1: Left. The distribution of Cepheids along the Large Magellanic Cloud LMC (top sample), deviates significantly from M33 (bottom sample). Right. M33 is aligned with LMC by shifting along mean magnitude.

Our general solution is to shift the data on the target domain, T_{te}, to the point where the model previously obtained over the source domain, T_{tr}, can be re-used. We

call this the data-alignment problem[17]. This differs from previous methods where the goal is to generate a model afresh by re-weighting examples on the training set[3]; instead we leave the model intact, and transform the testing set until the model can be re-used. An example of a successful alignment is shown in Figure 3.1 (right), where shifting the sample of M33 along the y-axis (apparent mean magnitude) by the right amount, enables us to do classification with the same model obtained on the source domain.

3.1.1 Problem Formulation

We generalize our problem as follows. Assume a training set T_{tr} and a predictive model $f(x|\theta)$ obtained by training on T_{tr}. Assume also an unlabeled testing set $T_{te} = x$, where each feature vector can be split into two parts: $x = (X_A, X_B) = (x_1, x_2, ..., x_m, x_{m+1}, x_{m+2}, ..., x_n)$. Without loss of generalization, we assume each feature in the set $X_B = \{x_{m+1}, x_{m+2}, ..., x_n\}$ has suffered a shift. We wish to generate a new dataset $T'_{te} = x'$, where $x' = (X_A, X'_B) = (x_1, x_2, ..., x_m, x_{m+1} + \delta_{m+1}, x_{m+2} + \delta_{m+2}, ..., x_n + \delta_n)$, to achieve the goal of aligning T_{te} along T_{tr}. Our real focus is in the set $\Delta = \delta_{m+1}, \delta_{m+2}, ..., \delta_n$; we aim at choosing Δ to minimize $R(\theta, P_{te}(x, w))$.

Note that even if the alignment is done appropriately, the joint distributions between training and testing could differ because $P_{tr}(x) \neq P_{te}(x)$ due to a form of sample bias. A proper alignment simplifies the data set shift problem into a covariate shift problem.

Applying active learning after this process will guarantee a better performance of the model.

3.1.2 A Maximum-Likelihood Approach

Our method to attack the problem above consists of shifting T_{te} using maximum likelihood. To begin, we first assume the marginal distribution from which the training is drawn, $P_{tr}(x)$, is a mixture of Gaussians. We estimate parameters directly from our sample T_{tr}, since we know all class labels (i.e., we know which vector belong to each component or Gaussian). This enables us to have a complete characterization of the marginal distribution:

$$P_{\text{tr}}(x_1, x_2, ..., x_m, x_{m+1}, x_{m+2}, ..., x_n) = \sum_{i=1}^{c} \phi_i g_i(x|\mu_i, \Sigma_i) \tag{3.1}$$

where

$$g_i(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} exp\{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\} \tag{3.2}$$

where ϕ_i, μ_i, and Σ_i are the mixture coefficient (i.e., prior probability), mean and covariance matrix of the ith component respectively, n is the number of features, and c is the number of components.

Our next step is to define a new testing set $T'_{te} = \{x'\}$, where $x = (x_1, x_2, ..., x_m, x_{m+1}+ \delta_{m+1}, x_{m+2} + \delta_{m+2}, ..., x_n + \delta_n)$, since we know a shift has occurred along our input features. Our approach is to find the Δ that maximizes the log likelihood of T_{te} with

respect to distribution $P_{tr}(x)$:

$$L(\Delta|T_{te}) = \log \prod_{k=1}^{q} P_{tr}(x^k) = \sum_{k=1}^{q} \log P_{tr}(x^k) \tag{3.3}$$

To simplify our notation, we can rewrite the marginal distribution (equation3.1) as follows:

$$P_{\text{tr}}(x_1, x_2, ..., x_m, x_{m+1}, x_{m+2}, ..., x_n) = \sum_{i=1}^{c} \alpha_i v_i(x^k) \tag{3.4}$$

where

$$\alpha_i = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}}$$

and

$$v_i(x^k) = exp\{-\frac{1}{2}(x^k - \mu_i)^T \Sigma_i^{-1}(x^k - \mu_i)\}$$

When we apply a shift across the set of features, the resultant marginal test distribution can be modeled as follows:

$$P_{\text{te}}(x_1, x_2, ..., x_m, x_{m+1}, x_{m+2}, ..., x_n) = \sum_{i=1}^{c} \alpha_i z_i(x^k) \tag{3.5}$$

where

$$z_i(x^k) = exp\{-\frac{1}{2}(x^k + \delta - \mu_i)^T \Sigma_i^{-1}(x^k + \delta - \mu_i)\}$$

Here we define a new vector δ that captures the amount of shift applied to each feature; vector δ has values other than zero only for the set of features in X_B (Section 3.1.1).

Algorithm 2 Gradient Ascent Algorithm

Input : Target data T_{te} , Learning rate η, Weight vector W= [W_1,W_2,...W_m] where m is the number of features and W_i ϵ $\{0, 1\}$. Here W_i=1 denotes the the data has to be shifted along ith feature and W_i=0 denotes no shift along ith feature

Output : Final Δ^*

Initialize : Shift Δ as zero vector, , error ϵ

1: While $\epsilon \geq .001$

2: For every iteration $t > 0$

3: Evaluate the gradient G_t of

$$L(\Delta|T'_{te}) = \sum_{k=1}^{q} \log \left(\sum_{k=1}^{c} \alpha_i z_i(x^k) \right)$$

4: Update Δ as Δ_t=$\Delta_{t-1} \times W + \eta \times G \times W$

5: Endwhile

The equation for the log-likelihood function ends up as follows:

$$L(\Delta|T'_{te}) = \sum_{k=1}^{q} \log \left(\sum_{k=1}^{c} \alpha_i z_i(x^k) \right) \tag{3.6}$$

To solve this optimization problem, we use an iterative gradient ascent approach as shown in algorithm 2 ; we search the space of values in Δ for which the log-likelihood function reaches a maximum value. We stop the iteration when the change in the log-likelihood function is less than a small value (in our experiments = 0.001).

We summarize the above steps in Algorithm 3

Algorithm 3 Proposed Data Alignment Algorithm

Input : Source data T_{tr}, Target data T_{te}, Optimal source classifier M_S^*.
Output : The optimal model M_T^* on the target dataset.

1: Fit the source data T_{tr} into a Gaussian Mixture Model as given in equation 3.1

$$P_{\text{te}}(x_1, x_2, ..., x_m, x_{m+1}, x_{m+2}, ..., x_n) = \sum_{i=1}^{c} \alpha_i z_i(x^k)$$

where

$$z_i(x^k) = exp\{-\frac{1}{2}(x^k + \delta - \mu_i)^T \Sigma_i^{-1}(x^k + \delta - \mu_i)\}$$

2: Define the shifted target dataset $T'_{te} = \{x'\}$, where $x = (x_1, x_2, ..., x_m, x_{m+1} + \delta_{m+1}, x_{m+2} + \delta_{m+2}, ..., x_n + \delta_n)$

3: Find the value of Δ in equation 3.6 that maximizes the following log-likelihood function using gradient ascent algorithm described in algorithm 2.

$$L(\Delta|T'_{te}) = \sum_{k=1}^{q} \log \left(\sum_{k=1}^{c} \alpha_i z_i(x^k) \right)$$

4: Substitute the value of Δ in the target dataset to obtain the shifted target dataset T^*_{te}

5: The optimal source classifier M_S^* becomes the optimal target classifier M_T^* as now it can be claimed that $P_{te}(x, w) = P_{tr}(x, w)$.

3.2 Data-Alignment Method Using Linear Mixture Models

Different from previous work, we assume a bi-variate Linear Mixture Model with Gaussian noise, and use Maximum Likelihood to find the shift between source and target distributions. The idea is to align the two datasets so that a model learnt on the source domain can be effectively used on the target domain.

3.2.1 Mathematical Formulation

We generate a mixture of linear regression models, as described in [79]. The log likelihood equation for LMC data for the two features X (Log Period) and Y (Apparent Mean Magnitude) can be written as follows:

$$LogL(\theta|y_1, y_2, ...y_n, x_1, x_2, ...x_n) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{2} \pi_j \sigma_j(y_i|x_i) \right) \tag{3.7}$$

where y_1, y_2 , ... y_n are observations of Y, x_1, x_2 , ... x_n are observations of X, and each component $\sigma_j(y_i|x_i)$ corresponds to a Gaussian distribution $N(x_i^T\beta_j, \sigma_j)$ with coefficient π_j .Parameter estimates are captured in θ. The equation can be expanded as follows:

$$LogL(\theta|y_1, y_2, ...y_n, x_1, x_2, ...x_n) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{2} \pi_j \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{(y_i - x_i^T\beta_j)^2}{2\sigma_j^2}} \right) \tag{3.8}$$

where $x_i = [x_{i1}x_{i2}]^T$ and $\beta_j = [\beta_{j1}x_{j2}]^T$. Here x_{i1} and x_{i2} indicate the value of Log Period and the bias variable for the ith observation respectively (x_i2=1 for all observations). β_j1 and β_j2 are the corresponding regressor variables for the jth component. To align M33 data with LMC data, we assume a shift δ across Mean Magnitude only. The log-likelihood equation for M33 data can be written as follows:

$$LogL(\theta, \delta|y_1, y_2, ...y_n, x_1, x_2, ...x_n) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{2} \pi_j \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{(y_i + \delta - x_i^T\beta_j)^2}{2\sigma_j^2}} \right) \tag{3.9}$$

To find the maximum value of δ we differentiate 3.9 with respect to δ and equate to zero. After some algebraic manipulation we derive the following equation:

$$\left(\frac{c_{i12}\alpha_{i1}e^{-\gamma_{i1}(\delta)} + c_{i22}\alpha_{i2}e^{-\gamma_{i2}(\delta)}}{\alpha_{i1}e^{-\gamma_{i1}(\delta)} + \alpha_{i2}e^{-\gamma_{i2}(\delta)}}\right) + \left(\frac{c_{i13}\alpha_{i1}e^{-\gamma_{i1}(\delta)} + c_{i23}\alpha_{i2}e^{-\gamma_{i2}(\delta)}}{\alpha_{i1}e^{-\gamma_{i1}(\delta)} + \alpha_{i2}e^{-\gamma_{i2}(\delta)}}\right) = 0 \quad (3.10)$$

where

$$c_{ij1} = \frac{(y_i + \delta - x_i^T\beta_j)^2}{2\sigma_j^2}, \quad c_{ij2} = \frac{(y_i + \delta - x_i^T\beta_j)}{\sigma_j^2}, \quad c_{ij3} = \frac{1}{2\sigma_j^2}$$

$$\alpha_{ij} = \pi_j \frac{1}{\sigma_j\sqrt{2\pi}} e^{-c_{ij1}}, \quad \gamma_{ij} = c_{ij2}\delta + c_{ij3}\delta^2$$

We shift the target dataset across the apparent mean magnitude by δ and classify the target dataset using the source model $f(x|\theta)$. If we have a small budget to query instances in the target domain , we use it to find the most informative instances of the target domain using active learning. We choose $f(x|\theta)$ as our initial model for active learning.

3.3 A Novel Method Of Domain Adaptation Using Active Learning

In this section we describe a novel method of domain adaptation using active learning. Most of the domain adaptation algorithms assume the existence of covariate shift. But in many real world scenarios, this assumption does not hold true. However, we

can assume that since the source and the target are related datasets, the complexity of a particular class of models across the two domains are similar. The goal of this novel method is to determine the best possible complexity for a particular class of the models given the target dataset, by using the information of complexity of the same class of models on the source dataset and the labels of the target dataset using active learning.

3.3.1 Problem Formulation

We generalize this problem similar to the way as given in section 3.1.1. We assume a training set T_{tr} and an unlabeled testing set T_{te}. The joint input-output distributions of source and target datasets are represented by $P_{tr}(x, w)$ and $P_{te}(x, w)$ respectively, where $P_{tr}(x, w)$=$P_{tr}(w|x)$ $P_{tr}(x)$ and $P_{te}(x, w)$=$P_{te}(w|x)$ $P_{te}(x)$. In this scenario we do not assume the existence of the covariate shift between the datasets T_{tr} and T_{te}. Thus we have $P_{tr}(w|x) \neq P_{te}(w|x)$ and $P_{tr}(x) \neq P_{te}(x)$. However, we consider the source and the target domains are related by having similar complexity on the predictive models $f_{tr}(x|\theta)$ and $f_{te}(x|\theta)$ built from a class of models M. Thus we can assume that the prior distributions of the parameters of M are same across both source and target distributions, i.e., $P_{tr}(\theta|M)$=$P_{te}(\theta|M)$. Since the class of models is fixed for both domains, we can write $P_{tr}(\theta)$=$P_{te}(\theta)$. However the posterior distribution of the parameters of the model given the dataset are not the same, i.e., $P_{tr}(\theta|T)\neq P_{te}(\theta|T)$, where T represents the data.

Our goal is to estimate the best parameters θ^*_{te} of the class of models M by maximizing posterior probability $P_{te}(\theta|T)$ that can be written as the following equation using Bayesian statistics:

$$P_{te}(\theta|T) = \frac{P_{te}(T|\theta)P_{te}(\theta)}{\sum_{i=1}^{n} P_{te}(T|\theta)P_{te}(\theta)} \tag{3.11}$$

where $P_{te}(T|\theta)$ and $P_{te}(\theta)$ are the likelihood and prior distribution of the parameters θ of the class of models M in the target domain. As the denominator is constant and invariant for a particular value of θ we can reduce the equation 3.11 to

$$P_{te}(\theta|T) = c_1 P_{te}(T|\theta)P_{te}(\theta) \tag{3.12}$$

where c_1 is a constant. For simplicity, we assume c_1=1 and θ as one dimensional parameter of the class of models M. To find the maximum value of $P_{te}(\theta|T)$, we calculate the maximum value value of the product of $P_{te}(T|\theta)$ and $P_{te}(\theta)$.

$$P_{te}(\theta|T) \propto P_{te}(T|\theta)P_{te}(\theta)$$

3.3.2 Estimating Prior Distribution of Parameter θ Using Domain Adaptation

We assume that the prior distribution of the parameter θ can be represented as Gaussian distribution over a single variable θ.

$$P(\theta) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{\theta-\mu}{2\sigma^2}} \tag{3.13}$$

where μ and σ are the mean and standard deviations of the Gaussian distribution. And σ^2 is the variance. Thus $P_{te}(\theta)$ and $P_{tr}(\theta)$ can be modeled like the equation 3.13 as :

$$P_{tr}(\theta) = \frac{1}{\sqrt{2\sigma_{tr}^2\pi}} e^{-\frac{\theta-\mu_{tr}}{2\sigma_{tr}^2}} \tag{3.14}$$

and

$$P_{te}(\theta) = \frac{1}{\sqrt{2\sigma_{te}^2\pi}} e^{-\frac{\theta-\mu_{te}}{2\sigma_{te}^2}} \tag{3.15}$$

where μ_{tr} and σ_{tr} are the mean and standard deviations of the parameter θ in the source domain respectively and μ_{te} and σ_{te} are the mean and standard deviations of the parameter θ in the target domain respectively As $P_{te}(\theta)$=$P_{tr}(\theta)$, μ_{te}=μ_{tr} and σ_{te}=σ_{tr}. Thus if we can estimate the values of μ_{tr} and σ_{tr} from the source dataset, we can substitute them for the values of μ_{te} and σ_{te} in the equation 3.15, thus obtaining the prior distribution of θ on the target dataset.

For this purpose, we generate K samples of the source dataset using uniform random sampling without replacement. Each of the K samples contain P percent of the source dataset. Both K and P can be regarded as design parameters of the experiments and are thus user-defined.

We build classifiers on each of the K samples using a range of m θ values, $\theta \in \{\theta_1, \theta_2, \theta_3, ..\theta_m\}$ based on the class of models M. For each sample denoted by S_{tr}^k we get a value θ_i^{k*}, where $1 \leq i \leq m$, that gives the maximum classification accuracy on the Kth sample S_{tr}^k. Thus we get K values of θ from the all the source sample datasets. We fit these values of θ in a single variate gaussian distribution as given

in the equation 3.14 and estimate the values of μ_{tr} and σ_{tr}. As a result we get the values for μ_{te} and σ_{te}. Thus we get the maximum value of the prior $P_{tr}(\theta)$ at θ=μ_{tr}. Also we can say that the best parameter on the source θ^*_{tr} is given by μ_{tr}.

3.3.3 Modeling the Likelihood of Parameter θ Using Active Learning

To model the likelihood of the parameter we use active learning to query the most informative instances in the target dataset, given a budget of cost B for querying by the expert, as described in algorithm 4. We use the state-of-art pool-based active learning using margin sampling as the uncertainty sampling technique. The algorithm first selects K instances from the pool P^U_{te} at random and queries their labels from the expert and add the labeled instances in Q_L. It then builds a model M_S on Q_L using the value of θ^*_{tr}. It removes the queried labels from the pool of unlabeled instances. Then it finds the example x_i with the maximum margin and queries it. It adds x_i to Q_L with its label w_i. it then removes x_i from P^U_{te} and builds a model M_S on Q_L. The process is repeated until the cost reaches the budget. After the budget has been used to query B instances on the target dataset, we proceed to estimate the likelihood $P_{te}(T|\theta)$.

It can be safely assumed that the probability of the dataset being classified correctly by a hypothesis is inversely proportional to the error made by the hypothesis on the dataset. Thus we define the likelihood of the parameter θ as a decaying

Algorithm 4 Active Learning Algorithm

Input : Target Pool P_{te}, learning algorithm L_{Base} with parameter θ=θ^*_{tr}, Budget B, Initial cost K.

Output : A set of labeled target examples Q^*_L.

Initialize : Q_L={}, $Cost$=0

1: Select K instances randomly from P_{te}.
2: Query K instances by expert and add them to Q_L.
3: $Cost$=$Cost$+K
4: Build a model M_S on Q_L using L_{Base} with θ^*_{tr}
5: Set the unlabeled pool dataset as $P^U_{te} = P_{te}$ - Q_L
6: While $Cost \leq B$
7: Find the instance x_i from P^U_{te} that has the minimum margin such that

$$x_i = \arg\min_x P_\theta(w^*_1|x) - P_\theta(w^*_2|x).$$

where w^*_1 and w^*_2 are the first and second most probable classes for x.
8: Query x_i by expert and add it to Q_{Lnew} such that Q_{Lnew} =$Q_L \cup (x_i, w_i)$ where w_i is the true label of x_i
9: $Cost$=$Cost$+1
10: Set Q_L=Q_{Lnew}.
11: Remove x_i from P^U_{te} such that P^{Unew}_{te}=P^U_{te} -x_i
12: Set P^U_{te}=P^{Unew}_{te}
13: Build a model M_{Snew} on Q_L using L_{Base} with θ^*_{tr}
14: Set $M_S = M_{Snew}$
15: Endwhile
16: Q^*_L= Final Q_L

function as follows:

$$P_{te}(T|\theta) = N_0 e^{-\lambda t} \tag{3.16}$$

where N_0 is a constant at t=0 , λ is a function of θ and t is equal to the classification error on the queried examples of the target dataset, also known as the in-sample error, denoted by E_{in}. At t=0, we define likelihood $P_{te}(T|\theta) = 1$. Thus substituting the values of t and $P_{te}(T|\theta)$ in equation 3.16 , we getN_0=1. Thus the equation of

likelihood becomes:

$$P_{te}(T|\theta) = e^{-\lambda t} \tag{3.17}$$

where t=E_{in} and λ=$k_1 g(\theta)$, where k_1 is a constant.

To obtain E_{in} we build a model on the queried examples for each of the values of θ_{te} where θ^*_{tr}-$\sigma_{te} \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma_{te}$ where σ_{te}=σ_{tr} is the standard deviation of the prior probability distribution $P_{te}(\theta)$ obtained from equation 3.15 as described in section 3.3.2.

We define E_{out} as the error on the out-sample error, i.e., the error estimated on the instances in input space that are not included in the training phase of building a classifier. We know from Vapnik-Chervonenkis inequality[82] that with a certain probability 1- δ , the bound on the error E_{out} is given by the equation :

$$E_{out} \leq E_{in} + \sqrt{\frac{8}{N} \ln \frac{4m_H(2N)}{\delta}} \tag{3.18}$$

where E_{in} is the in-sample error on N training instances and $m_H(q)$ is a polynomial function that defines that largest number of dichotomies on q training instances given the class of hypotheses H:

$$m_H(q) \leq \sum_{i=0}^{d_{VC}(H)} \frac{N!}{i!N-i!} \tag{3.19}$$

where $d_{VC}(H)$ is the Vapnik-Chervonenkis dimension, or more commonly known as VC dimension.The VC dimension $d_{VC}(H)$ is defined as the maximum number of examples that can be shattered by the hypothesis H[82]. Thus it depends on the complexity of the hypothesis and in turn is dependent on θ. The VC dimension of

various classes of hypothesis are known in the literature. For example, VC dimension $d_{VC}(H)$ of neural networks with sigmoid gate functions has a lower bound $\sigma(wlogw)$ and a upper bound of $O(w^2)$ [81] where w is the number of weights in the network. In our research the parameter θ we use is the number of hidden nodes h in a neural network. Thus $d_{VC}(H)$ of a network with h hidden nodes can be estimated as minimum $wlogw$ where w=$(i+1)\times\ h\ +\ (h\ +\ 1)\times\ o$ where i and o are the number of input features and output classes respectively.

We refer to the second term of the left-hand side of the equation 3.18 as $g(\theta)$.We say that the λ of the equation 3.17 is directly proportional to $g(\theta)$.Thus λ is defined as:

$$\lambda = k_1 g(\theta) = k_1 \sqrt{\frac{8}{N} \ln \frac{4m_\theta(2N)}{\delta}} \tag{3.20}$$

where k_1 is the constant of proportion. Thus, if we substitute the value of t with E_{in} and λ with $g(\theta)$, obtained from equation 3.20, in the equation 3.17, we will get a value of $P_{te}(T|\theta)$ for every value of θ_{te} where θ^*_{tr}-$\sigma \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma$.

Also, we know that $l > d_{VC}(H)$ from [82], the minimum value of $m_H(q)$ can be obtained using lowest value of l which is $d_{VC}(H)$+1, which has a lower bound $wlogw$. Also since $m_H(q)$ is always greater than zero and a monotonically increasing function, a higher value of θ will generate a higher VC dimension $d_{VC}(H)$ and in turn a higher value of λ, irrespective of the lower or upper bound of VC dimension $d_{VC}(H)$. Thus this method generalizes to all hypothesis whose VC dimension is greater than 1.

3.3.4 Estimating the Posterior of Parameter θ

Once we have estimated the value of the priors $P_{te}(\theta)$ and the values of the likelihood $P_{te}(T|\theta)$ for every value of θ_{te} where θ^*_{tr}-$\sigma_{te} \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma_{te}$, we multiply the priors $P_{te}(\theta)$ with the corresponding likelihood values $P_{te}(T|\theta)$ to obtain the posterior as given in the equation 3.11. The value of θ for which we get the maximum value of the posterior is our optimal value, θ^*_{te} of the parameter θ . We use this value θ^*_{te} to build the optimal classifier $f(x|\theta^*_{te})$ on the target domain using the B queried instances as our training dataset. We present our proposed method in an algorithm as given in Algorithm 5

$$\theta^*_{te} = \arg\max_{\theta} P_{te}(T|\theta)P_{te}(\theta). \tag{3.21}$$

The same method of determining $f(x|\theta^*_{te})$ can be also applied to the any scenario where the covariate shift assumption holds, including the scenario described in section 3.1.

Algorithm 5 Proposed Domain Adaptation Method Using Active Learning

Input : Source data T_{tr}, Target data T_{te}, learning algorithm L_{Base} with parameter θ, Budget B, Initial cost K.

Output : Model M_T^* on T_{te}

1: Find the prior distribution $P_{te}(\theta)$ of the parameter θ on the source dataset as described in section. 3.3.2.
2: Find the optimal parameter value θ_{tr}^*=μ_{tr} from $P_{tr}(\theta)$ where μ_{tr} is the mean of $P_{tr}(\theta)$. And as per assumption, $P_{te}(\theta)$=$P_{tr}(\theta)$.
3: Use θ_{tr}^* as the value of θ for the model M_T on the target dataset to find a set of B most informative instances Q_L labeled using active learning as described in algorithm 4, starting with K target instances.
4: Build model M_T on T_{te} for each of the values of θ_{te} where θ_{tr}^*-$\sigma_{te} \leq \theta_{te} \leq \theta_{tr}^*$+$\sigma_{te}$ where σ_{te} is the standard deviation of $P_{te}(\theta)$.
5: For each θ_{te} in the above step, find error $t = E_{in}$ on the Q_L.
6: For each θ_{te} in the above step, find the value of λ using equation 3.20.
7: For each λ in the above step, find the value of likelihood $P_{te}(T|\theta) = e^{-\lambda t}$.
8: For each θ_{te} find the value of posterior $P_{te}(\theta|T)$=$P_{te}(\theta|T)P_{te}(\theta)$.
9: Choose the θ as θ_{te}^* that maximizes $P_{te}(\theta|T)$
10: Build the model M_T^* on T_{te} using Q_L as the training set and θ_{te}^* as the optimal value for the parameter θ.

Chapter 4

Experiments and Results

This section includes the experimental results of the methods described in chapter 3. We use two separate sets of data for the two different scenarios. For scenario 1, in which we assume that the dataset shift has occurred due to a linear shift in the marginal distributions of the source and the target datasets, we use three datasets from astronomy domain which include data of the Cepheid stars from three different galaxies: Large Magellanic Clouds (LMC), Small Magellanic Clouds(SMC) and Triangulum galaxy(M33). For scenario 2, in which we do not assume that the target dataset has a covariate shift, we use Supernova datasets from astronomy domain and Landmine datasets that include information from the images for a landmine detection problem. We present the results for these two scenarios in section 4.1 and section 4.2 respectively.

4.1 Experiments For Novel Data-Alignment Method

In this section we show the results obtained by running the novel data-alignment method as described in section 3.1. Section 4.1.1 describes the datasets used for this purpose. Section 4.1.2 describes the experimental design and shows the results. Section 4.1.3 contains a variation of the method as described in Section 3.1. Section 4.1.4 describes three new features generated by us from the given datasets that improve their cross-validation accuracy. We combine these features with those used in sections 4.1.2 and 4.1.3 and present the results of our proposed data alignment method on these higher dimensional datasets in section 4.1.5.

4.1.1 Cepheid Star Datasets

Classical Cepheid variables (also commonly referred to as Population I Cepheids) are stars both massive 4-11 M_{Sun} (solar masses) and luminous $1 \times 10^3 - 5 \times 10^4$ L_{Sun} (solar luminosities), that undergo regular pulsations with periods ranging from 2 to 100 days. The pulsations are driven by cyclical changes in the opacity of hydrogen and helium in the outer atmosphere of these stars[68]. A tight correlation exists between period (P) and luminosity (L) for Cepheid variables:

$$log_{10}L = a + b \times log_{10}P \tag{4.1}$$

where a and b are fitting parameters. Figure 4.1 shows an example of such relation for a sample of Cepheids populating the Large Magellanic Cloud galaxy. The flux of each star is represented by its apparent magnitude m, defined as $m = -2.5 \times log_{10}\frac{L}{d^2}$,

where d is the distance from Earth to the star measured in parsecs. Hence, smaller numbers correspond to brighter magnitudes (higher fluxes).

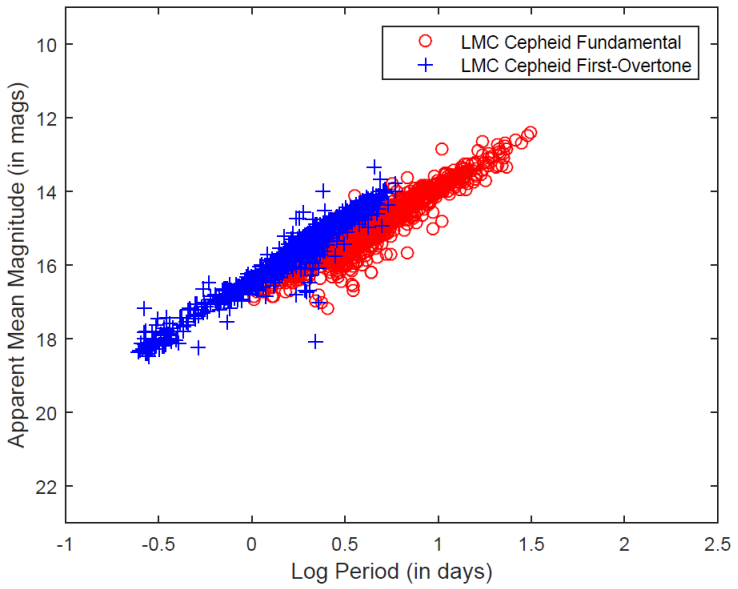

Figure 4.1: A linear relation exists between the logarithm of the period and the magnitude of a Cepheid star. The graph shows both fundamental and first-overtone Cepheids from the Large Magellanic Cloud that overlap more than expected due to interstellar dust and poor image resolution

Cepheid variables can be classified according to their pulsation mode(s): fundamental and first-overtone. Our analysis uses data obtained by two surveys: (1) the third phase of the Optical Gravitational Lensing Experiment (OGLE-III)[1] and (2) the M33 Synoptic Stellar Survey[2]. OGLE[72] is a collaboration of mostly Polish astronomers originally designed to search for dark matter using gravitational microlensing. The project has been observing the Magellanic Clouds and the Galactic

[1]Data repository available at http://ogledb.astrouw.edu.pl/ ogle/CVS/
[2]Data repository available at http://faculty.physics.tamu.edu/lmacri/M33SSS/

Bulge using a dedicated 1.3-m telescope at Las Campanas Observatory in Chile. As a side product of the search for microlensing events, OGLE has discovered tens of thousands of variable stars, including Cepheids. We analyze the third catalog of OGLE Large Magellanic Cloud Cepheids[73], on both the visible (V) and infrared (I) bands; the training set for the visible band contains 3,059 points (i.e., light curves), while the training set for the infrared band contains 3,056 points. Data is of excellent quality, with statistical uncertainties of 0.001% and 0.00035% on each band respectively. The approximate detection thresholds of these observations are $V = 21.5$ and $I = 21$ mag, well below the faintest apparent magnitude of any Cepheid in the Large Magellanic Cloud. The M33 Synoptic Stellar Survey[74] is a follow-up study of Cepheids and other variables initially discovered by the DIRECT project[75][76]. These testing sets contain 323 points in the visible band, and 322 points in the infrared band (statistical uncertainties of 0.0125% and 0.0056% respectively). The approximate detection thresholds of these observations are $V = 23$ and $I = 21.5$ mag, which are adequate to detect all fundamental mode Cepheids with P > 3 days and first-overtone Cepheids with P > 2 days.

4.1.2 Experimenting on Cepheids Datasets with 2 features

In the first set of experiments, we refer to data from the Large Magellanic Cloud galaxy as the source domain, and to data from M33 galaxy as the target domain. As a pre-processing step, parameters for the mixture of Gaussians on the source domain are estimated directly through the data. Since class labels are available, it

Table 4.1: Classification Performance under Visible Band. Numbers in parentheses represent standard deviations. Last two columns correspond to proposed methodology.

Learning Algorithm	Accuracy on LMC (source domain)	Accuracy on M33 (target domain)		
		Cross-Validation	Standard	Proposed Method
Neural Networks	94.77 (1.18)	96.18 (0.16)	90.09 (1.33)	96.78 (0.90)
SVM Polynomial 1	94.48 (1.19)	92.55 (0.26)	95.30 (0.28)	95.76 (0.88)
SVM Polynomial 2	94.59 (1.23)	95.28 (0.19)	93.69 (0.61)	95.98 (0.80)
SVM Polynomial 3	94.62 (1.21)	95.08 (0.20)	93.38 (0.66)	95.98 (0.80)
Decision Trees	94.32 (1.27)	97.61 (0.13)	94.49 (0.52)	97.27 (0.75)
Random Forest	93.80 (1.26)	97.78 (0.13)	94.12 (0.41)	96.13 (0.78)

is straightforward to estimate means, covariance matrices and mixing proportions. Implementation of the learning algorithms can be found in WEKA [77] using default parameters. Specifically, neural networks are invoked with one hidden layer, two internal nodes, a learning rate of 0.3, momentum of 0.2, and 500 epochs. Support Vector Machines are invoked with polynomial kernels of degrees 1, 2, and 3 respectively. Decision trees are invoked with a confidence factor of 0.25. Random Forests combine 10 decision trees on each run. For 2 features , we solve for 3.6 by invoking the $solvefunction$ in the $symbolicmathtoolbox$ in Matlab.

Our first set of experiments compute predictive accuracy when the model obtained on LMC is applied to the transformed target domain (when a shift is applied to apparent magnitude). Accuracy is obtained by creating bootstrap samples of the

target or test set (M33); each sample is generated by sampling with replacement. We generated 10 samples, and averaged our results after applying model $f(x|\theta)$ on each transformed sample. Table 4.1 shows our results when the data refers to the visible band. The first column corresponds to the learning algorithms. The second column corresponds to accuracy on the source domain (as reference only). The third column shows accuracy on the target domain when class labels are available. The next columns show accuracy on M33 when class labels are unavailable. The fourth column (labeled *Standard*) shows results using the values listed in Table 2 of [74]. We take into consideration that the apparent magnitudes of the OGLE LMC Cepheids are not corrected for the average amount of interstellar dust towards this galaxy, while [74] applies a correction for dust[3]. The fifth column (labeled $ProposedMethod$) shows results when δ is determined using maximum likelihood based on density estimation (Section 3.1).

Compared to previous work [74], we observe a significant increase in accuracy when δ is computed using the proposed method; this is true for all learning algorithms under consideration, which validates our proposed methodology. Furthermore, results for both show accuracies similar to those obtained on M33 when class labels are present, evidencing how an appropriate shift in magnitude renders the model trained on the source domain valid under the target domain. Our final estimation for δ for visible band is -6.24.

Table 4.2 displays results similar to the ones describe above, but using the infrared

[3]For reference, the adjustment values derived in[74] are δ = -6.45 $\pm$ 0.02 for the visible band, and -6.31 $\pm$ 0.02 for the infrared band.

Table 4.2: Classification Performance under Infrared Band. Numbers in parentheses represent standard deviations. The last column corresponds to proposed methodology.

Learning Algorithm	Accuracy on LMC (source domain)	Accuracy on M33 (target domain)		
		Cross Validation	Standard	Proposed Method
Neural Networks	96.89 (0.16)	97.05 (0.14)	94.25 (2.45)	96.99 (0.81)*
SVM Polynomial 1	97.00 (0.17)	93.18 (0.26)	96.80 (0.15)	95.50 (1.11)
SVM Polynomial 2	96.98 (0.17)	93.18 (0.26)	94.97 (0.58)	96.83 (0.79)*
SVM Polynomial 3	97.02 (0.17)	93.18 (0.26)	95.15 (0.82)	96.71 (0.78)*
Decision Trees	96.35 (0.17)	96.27 (0.17)	95.77 (0.49)	97.62 (1.08)*
Random Forest	96.47 (0.17)	95.78 (0.16)	95.62 (0.71)	96.86 (0.84)

band. Compared to previous work, we observe a general improvement in accuracy when is computed using maximum likelihood, but the improvement is less clear. Our final estimation for δ for infrared is -6.08 which is different from the one obtained using the visible band, because the effects of reddening due to interstellar dust are significantly reduced at infrared waves[78]. Overall, empirical results support our proposed methodology.

We further compare our results with the existing domain adaptation algorithms and present the results in Table 4.3 using the data only from the infrared band. It can be expected that the results will be similar in the visible band. In this table , the first column refers to the base learning algorithm. The rest of the columns show the classification accuracy on M33(target) data when different models were used. The second column refers to the classification accuracy when the model built on the source

Table 4.3: Comparison of Classification Performance under Infrared Band. Numbers in parentheses represent standard deviations.The last column corresponds to proposed methodology.

Learning Algorithm	Accuracy on M33 (target domain)			
	Source model	KMM	SA	Proposed Method
Neural Networks	92.70 (1.19)	92.93	76	96.99 (0.81)
SVM Polynomial 1	92.70 (1.19)	92.93	78	95.50 (1.11)
SVM Polynomial 2	92.70 (1.19)	92.93	77	96.83 (0.79)
SVM Polynomial 3	92.70 (1.19)	92.93	75.33	96.71 (0.78)
Decision Trees	92.90 (1.24)	94.28	77.67	97.62 (1.08)
Random Forest	92.90 (1.24)	94.35	72	96.86 (0.84)

is directly used on the target. The third column shows the accuracy on M33 data when we applied a well-known instance-based domain adaptation technique called *kernel mean matching*. The fourth column shows the accuracy on M33 data when we applied a well-known feature-based domain adaptation technique called *subspace alignment* method.The fifth column shows the accuracy on M33 data when we used our proposed method.

From this table we see that our proposed method achieves a higher accuracy than the existing domain adaptation methods, thus eradicating the need of building a new model on the target dataset and as a result decreasing the complexity of the problem.

Table 4.4: Initial values and final parameter estimates for fitting a linear mixture model on LMC source dataset

Paramter	Initial value	Final Value
β_1	[-3.259 16.407]'	$[-3.25916.407]'$
β_2	[-2.969 16.904]'	$[-2.92916.889]'$
α_1	0.1	0.164
α_2	0.1	0.214
π_1	0.404	0.384
π_2	0.596	0.606

4.1.3 Experimenting on Cepheid dataset with 2 features using Linear Mixture Model

We run experiment on the same datasets as we do in Section 4.1.2, but instead of fitting the data into a Gaussian Mixture model, we fit the data into a mixture of Linear model[80] with Gaussian Noise as described in following sections.

4.1.3.1 Fitting a Mixture of Linear Models

We use equation 3.8 as given in the section 3.2.1 to fit a mixture of linear models to the source data. The two components refer to the two classes: Fundamental and First-Overtone. We use the EM algorithm[79] to do parameter estimation. The EM algorithm stops when the change in Q-value in the E-step falls below 10^{-10}. Initial values and final parameter estimates are shown in Table 4.4.

Table 4.5: Classification Accuracy of M33 Dataset by fitting it into Linear Mixture Model with Gaussian noise.

Algorithm	Accuracy on M33 (target domain)	
	No Data Alignment	With Data Alignment
Neural Networks	92.70 (1.19)	97.90 (0.89)
SVM Polynomial 1	92.70 (1.19)	96.27 (0.86)
SVM Polynomial 2	92.70 (1.19)	96.10 (0.80)
SVM Polynomial 3	92.70 (1.19)	96.53 (0.88)
Decision Trees	92.90 (1.24)	98.07 (0.83)
Random Forest	92.90 (1.24)	98.03 (0.69)

4.1.3.2 Aligning M33 Data with LMC Data

We use the parameter values in Table 4.4 to find the shift between LMC and M33. Solving equation 3.10 yields a value of δ=-6.06. After shifting M33 data by δ we obtain the results shown in Table 4.5. Experimental results show how classification accuracy increases significantly after the source and target datasets are aligned.

We present a summary of the classification errors on M33 dataset for our proposed methods against the popular domain adaptation method, kernel mean matching(KMM), in figure 4.2. We observe that except SVM Polynomial with degree 1, all other base algorithms yield error lower than KMM as well as the standard shift for both the proposed methods.

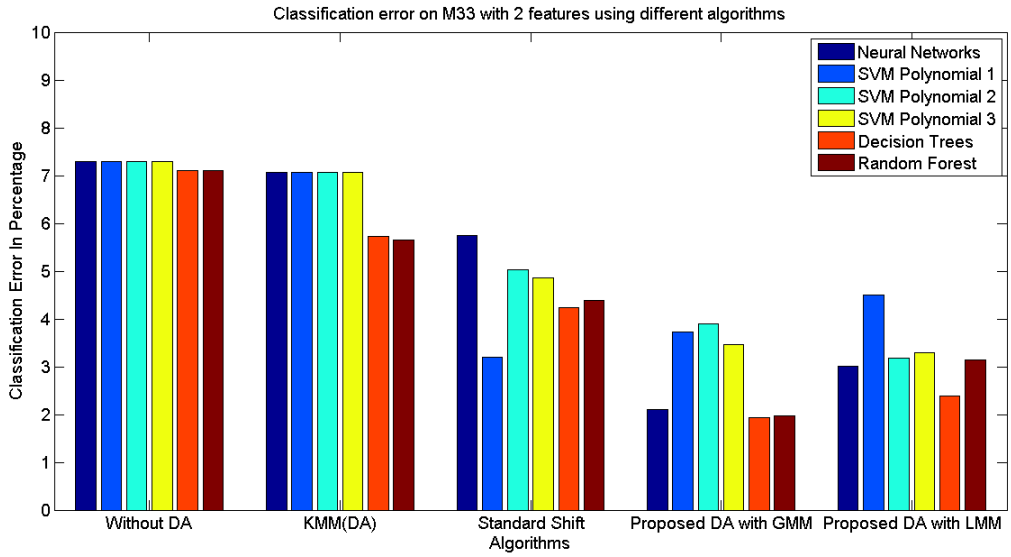

Figure 4.2: Summary of classification error (in percentage) of all the algorithms with and without domain adaptation using different base learning algorithms.The bar groups are in the following order : (1) 'Without DA': using the source classifier directly on the target (2) 'KMM(DA)': Using Kernel Mean Matching (3) 'Standard Shift': using standard shift (4)'Proposed DA with GMM': using proposed data alignment method using Gaussian Mixture Model (5) 'Proposed DA with LMM': using proposed data alignment method using Linear Mixture Model

4.1.4 Light Curve Feature Characterization

Most previous work in light curve classification deals with features useful over a broad number of classes. As an example, identification of microlensing events requires features that can discriminate among several variable sources such as periodic variables, eruptive variables, and supernovae[70][71]. Typically, features are extracted from spectral analysis, and can be specialized or generic; the latter includes statistical tests, coefficients from the Fourier transform, and auto-correlation coefficients. In contrast, our study targets specialized geometric properties that serve

as better discriminators between the two types of pulsation mode in Cepheid variable stars (fundamental and first-overtone). We employ five features out of which three - $PhaseShift$, $Depth$, and $Slope$ - are calculated by fitting a polynomial curve along the points of one pulsation cycle. The other two features, $MeanMagnitude$ and $LogPeriod$, are directly available from the Optical Gravitational Lensing experiment (Section 4).

Figure 4.3 shows our features along typical light curves for the two different pulsation modes. $PhaseShift$ is the phase at which the light curve shows its highest magnitude (apparent brightness) in one phase cycle; $Depth$ is equal to the difference between the highest and lowest magnitudes in one complete pulsation cycle; $Slope$ is the angle of the line that connects the highest and lowest magnitudes. These features point to stark differences in the geometric shape of light curves between the two classes of interest, and thus appear more informative than other more typical features.

4.1.5 Experimenting on Cepheids Dataset with 5 features

We use the same dataset as in Section 4.1.2 but with the additional three new proposed features as mentioned in Section 4.1.4. As a pre-processing step, parameters for the mixture of Gaussians on the source domain are estimated directly through the data. We assume two Gaussians (one Gaussian for class fundamental and one for overtone). Since class labels are available, it is straightforward to estimate means,

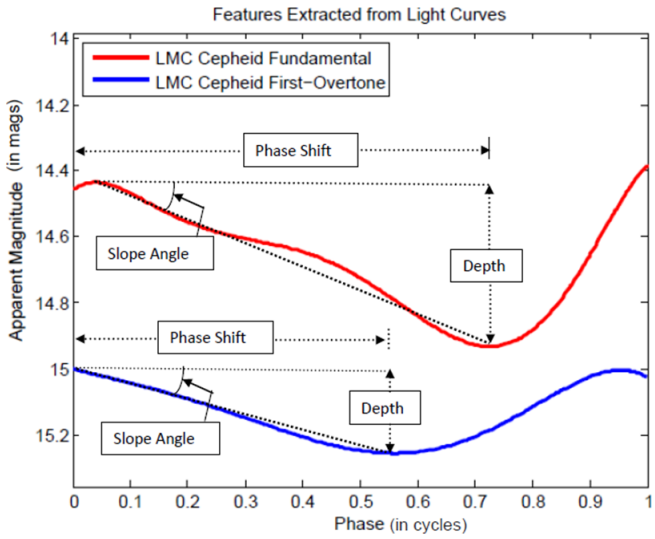

Figure 4.3: Features capture geometric properties of light curves that exhibit clear differences between the two classes of interest.

covariance matrices and mixing proportions. Implementation of the learning algorithms can be found in WEKA[77] using default parameters. Specifically, neural networks are invoked with one hidden layer, two internal nodes, a learning rate of 0.3, momentum of 0.2, and 500 epochs. Support Vector Machines are invoked with polynomial kernels of degrees 1, 2, and 3 respectively. Decision trees are invoked with a confidence factor of 0.25. Random Forests combine 10 decision trees on each run.

Our experiments compute predictive accuracy when the model obtained on LMC (source domain) is applied to the transformed target sample (where a shift Δ is added to each feature vector, Section 3.1). Accuracy is obtained by creating bootstrap samples of the target or test set(M33); each sample is generated by sampling with

Table 4.6: Classification Accuracy on M33 dataset with 5 Features with and without alignment.

Learning Algorithm	Accuracy on M33 (target domain)		
	No Data Alignment	With Data Alignment	
		Using Magnitude	Using All Features
Neural Networks	91.82 (1.42)	91.82 (1.42)	92.76 (1.09)
SVM Polynomial 1	91.82 (1.42)	91.82 (1.42)	95.21 (1.00)
SVM Polynomial 2	91.82 (1.42)	91.82 (1.42)	95.42 (0.89)
SVM Polynomial 3	91.82 (1.42)	91.82 (1.42)	94.91 (0.75)
Decision Trees	91.64 (1.60)	91.64 (1.60)	93.79 (1.00)
Random Forest	91.64 (1.42)	91.64 (1.42)	94.24 (0.89)

replacement. We generated 10 samples, and averaged our results after applying the training model on each transformed sample. Table 4.6 shows our results. The first column corresponds to the learning algorithms. The second column shows accuracy values on M33 with no data alignment. The third and fourth columns show accuracy on M33 with data alignment; we make a distinction between shifting along apparent magnitude only vs shifting over all features[4] (including our geometric features, section 4.1.4). Numbers in parentheses represent standard deviations.

Our results show a significant increase in accuracy when the data alignment step is applied to the target sample, and all features are used during data alignment ($p = 0.05$ level using a t-student distribution); this is true for all learning algorithms

[4]When shifting along all features we exclude the log period, as this is independent of the distance to galaxies.

under consideration. There is also an advantage when data alignment is effected over all features, as opposed to shifting along magnitude only, but the differences are not always significant. The mean increase in accuracy over all learning algorithms with M33 as the target domain is 0% when shifting along magnitude, but 2.57% when shifting along all features. Overall our results show the effectiveness of the data alignment step, and the geometric feature representation.

Additional observations can be drawn from Table 4.6. For example, accuracy gain is more clear when using Neural Networks, Decision Trees, and SVMs with high order polynomials; these algorithms tend to generate complex decision boundaries, which suggests a relatively complex class distribution. In addition, it is interesting to observe that the gain in accuracy between LMC and M33 is evident when shifting along all features, but there is no accuracy gain when shifting along magnitude.

Table 4.7: Comparison of Classification Performance under Infrared Band. Numbers in parentheses represent standard deviations.The last column corresponds to proposed methodology.

Learning Algorithm	Accuracy on M33 (target domain)			
	Source model	KMM	SA	Proposed Method
Neural Networks	91.82 (1.42)	96.03	68	92.76 (1.09)
SVM Polynomial 1	91.82 (1.42)	93.27	72	95.21 (1.00)
SVM Polynomial 2	91.82 (1.42)	93.94	69.33	95.42 (0.89)
SVM Polynomial 3	91.82 (1.42)	93.60	68.67	94.91 (0.75)
Decision Trees	91.64 (1.60)	94.63	74.67	93.79(1.00)
Random Forest	91.94 (1.42)	94.28	68.33	94.24 (0.89)

We show the classification error in percentage on M33 dataset with 5 features using source model, KMM and our proposed method in figure 4.4. We also include the error when KMM is used after aligning the target data with the source data using our proposed method.

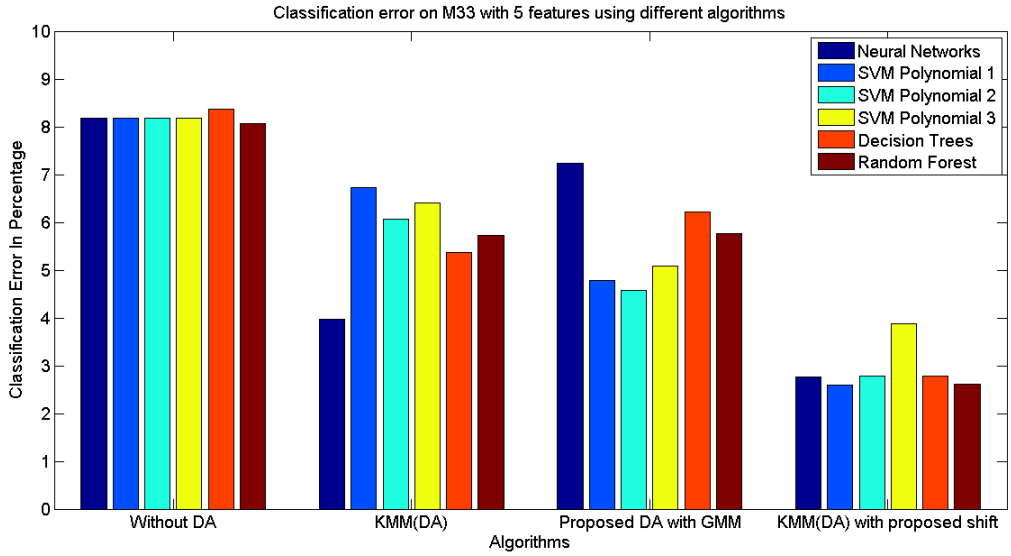

Figure 4.4: Summary of classification error (in percentage) of algorithms with and without domain adaptation using different base learning algorithms.The bar groups are in the following order : (1) 'Without DA': using the source classifier directly on the target (2) 'KMM(DA)': Using Kernel Mean Matching (3) 'Proposed DA with GMM': using proposed data alignment method using Gaussian Mixture Model (4) 'KMM(DA) with proposed shift': using proposed data alignment method and then using KMM on shifted data

Similar to Table 4.3, we further compare our results with the existing domain adaptation algorithms and present the results in Table 4.7 using the data only from the infrared band. In this table, the first column refers to the base learning algorithm. The rest of the columns show the classification accuracy on M33(target) data when

different models were used. The second column refers to the classification accuracy when the model built on the source is directly used on the target. The third column shows the accuracy on M33 data when we applied a well KMM. The fourth column shows the accuracy on M33 data when we apply subspace alignment method. The fifth column shows the accuracy on M33 data when we used our proposed method.

We use the same methods on two other pairs of datasets. Like the previous pair LMC as the source and M33 as the source, we use data from Small Magellanic Clouds (SMC) as the source and M33 as the target and apply our proposed method. We show the results in Table 4.8. For 2 features we get δ=-5.70 for the SMC-M33 pair. As SMC is very close to LMC and show similar characteristics, we do not shift SMC against LMC with 2 features. We use our domain alignment method on the pair LMC and SMC only with 5 features, taking LMC as the source and SMC as the target. We present the results in Table 4.9

Table 4.8: Classification Performance under Infrared Band using SMC as the source dataset. Numbers in parentheses represent standard deviations.

Learning Algorithm	Accuracy on M33 (target domain)			
	Using 2 Features		Using 5 Features	
	Source model	Proposed Method	Source model	Proposed Method
Neural Networks	92.00 (0.28)	98.00 (0.13)	92.33 (0.28)	91.00 (0.29)
SVM Polynomial 1	92.00 (0.28)	98.00 (0.14)	92.33 (0.28)	91.33 (0.29)
SVM Polynomial 2	92.00 (0.28)	97.33 (0.16)	92.67 (0.27)	94.00 (0.24)
SVM Polynomial 3	92.00 (0.28)	97.67 (0.15)	92.33 (0.28)	94.00 (0.24)
Decision Trees	92.00 (0.28)	97.00 (0.17)	91.67 (0.29)	94.00 (0.24)
Random Forest	92.00 (0.28)	97.67 (0.15)	92.67 (0.26)	94.67 (0.21)

Table 4.9: Classification Performance under Infrared Band using LMC as the source dataset and SMC as the target dataset. Numbers in parentheses represent standard deviations.

Learning Algorithm	Accuracy on SMC (target domain)	
	Using 5 Features	
	Source model	Proposed Method
Neural Networks	91.31(0.29)	99.41(0.17)
SVM Polynomial 1	98.01(0.19)	98.67(0.19)
SVM Polynomial 2	97.42(0.10)	98.96(0.19)
SVM Polynomial 3	97.95(0.10)	99.18(0.18)
Decision Trees	96.58(0.27)	91.64(1.60)
Random Forest	98.23(0.22)	99.21(0.25)

We show the classification error of cross validation on all three datasets in figure consisting of 2 features and 5 features , thus proving the effectiveness of using three proposed features. We show the classification error on M33 dataset using SMC as the source dataset with 2 features and 5 features in figure 4.6 and figure 4.7 respectively. We also show the classification error on SMC dataset using LMC as the source dataset with 5 features in figure 4.8.

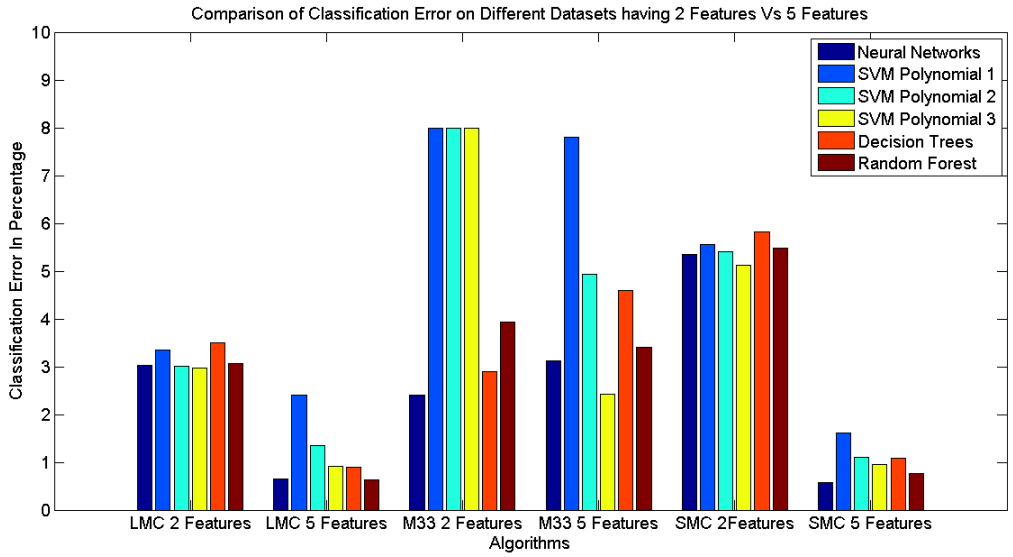

Figure 4.5: Classification error (in percentage) of algorithms on all three datasets : LMC, M33 and SMC.The bar groups are in the following order : (1) and (2) Error on LMC dataset having 2 features vs 5 features (3) and (4) Error on M33 dataset having 2 features vs 5 features (5) and (6) Error on SMC dataset having 2 features vs 5 features

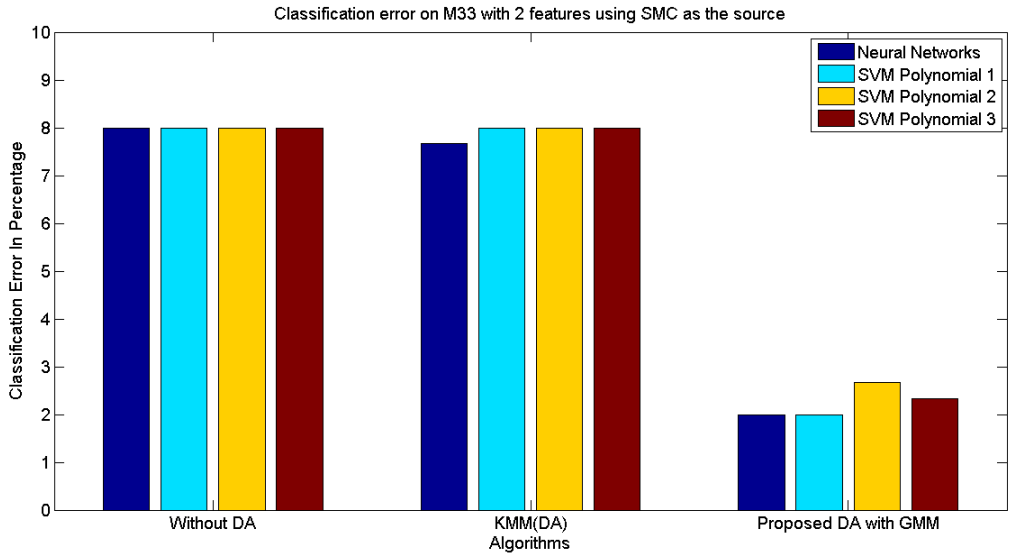

Figure 4.6: Classification error (in percentage) of algorithms on M33 dataset with 2 Features. The bar groups are in the following order : (1) Error using source model directly on target without domain adaptation (2)Error using domain adaptation algorithm kernel mean matching (3) Using proposed algorithm with gaussian Mixture model

4.2 Experiments For Novel Domain Adaptation using Active Learning

In this section we present the experimental results for the novel domain adaptation method using active learning as described in 3.3.

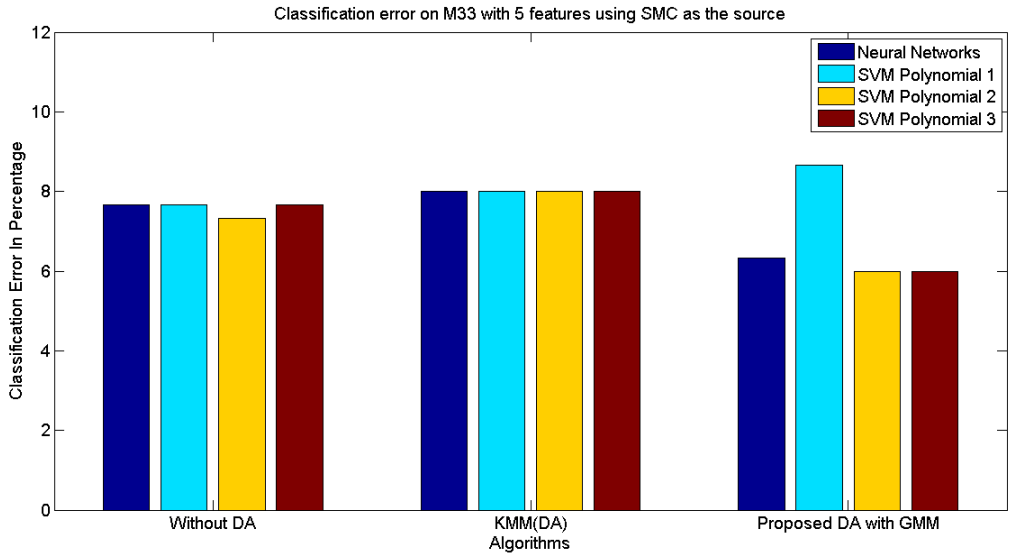

Figure 4.7: Classification error (in percentage) of algorithms on M33 dataset on 5 features. The bar groups are in the following order : (1) Error using source model directly on target without domain adaptation (2)Error using domain adaptation algorithm kernel mean matching (3) Using proposed algorithm with Gaussian Mixture model

4.2.1 Supernova and Landmine Datasets

4.2.1.1 Supernova Datasets

The supernova datasets used in this research have been provided by Supernova Photometric Classification Challenge [83]. It is a simulated dataset but at par with the expected real world dataset. It consists of supernova light curves simulated according to Dark Energy Survey specifications with the help of SNANA light curve simulator[84]. The dataset has been divided into training set and test set which we treat as our source and target datasets respectively. The source dataset consists of

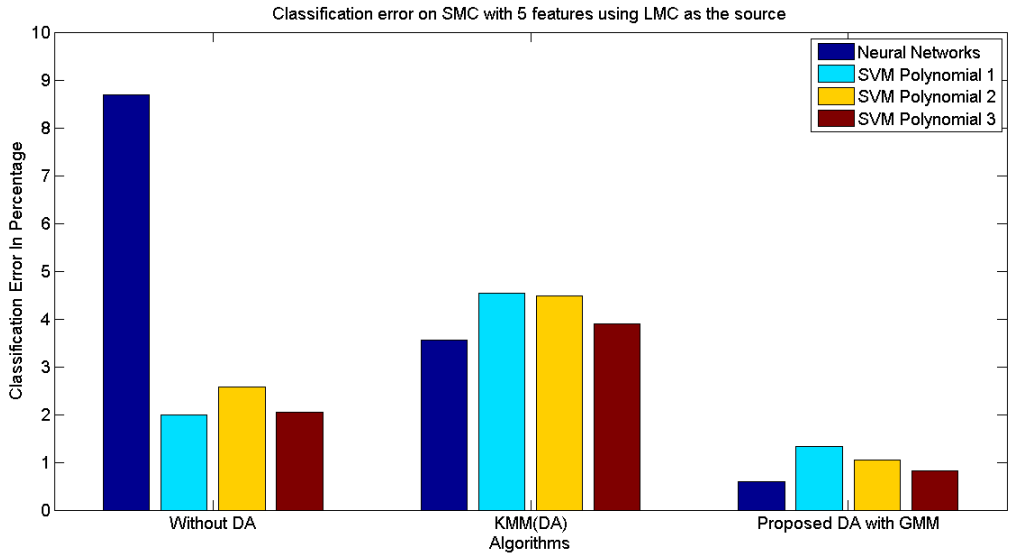

Figure 4.8: Classification error (in percentage) of algorithms on SMC dataset on 5 features. The bar groups are in the following order : (1) Error using source model directly on target without domain adaptation (2)Error using domain adaptation algorithm kernel mean matching (3) Using proposed algorithm with Gaussian Mixture model

718 light curves obtained using spectroscopic method, whereas the target dataset consists of 11946 photometric samples. The three classes of this dataset are supernova Type Ia, Ib and Ic. We use these datasets to classify supernova Type Ia versus the other two classes, Type Ib and Type Ic. The number of features in the original dataset is 108. However we used a low dimensional versions of the original dataset. The original data is reduced using Kernel Principal Component Analysis[86] to 20 features for 2-class classification problem.

4.2.1.2 Landmine Datasets

The landmine datasets that have been used are referred in [85]. There are 29 datasets each of which consists data from different mine fields. Each instance in the dataset has 9 features extracted from the radar images of the mine fields and belong to either of the classes 0 or 1. Here 1 represents the presence of landmine and 0 denotes its absence. The goal is to build a classifier to detect the landmines(class-1) from the clutter(class-0).

Among these 29 data sets, datasets 1 to 15 correspond to regions that are relatively highly foliated and datasets 16 to 29 correspond to regions that are bare earth or desert. Thus they are suitable for being used as domain adaptation methods. However all of them are unbalanced datasets. So, for our research we combine datasets 1 to 5 to produce our source dataset and use dataset 24 as our test dataset.

4.2.2 Estimating the Prior of θ using Domain Adaptation

To determine the prior of the parameter θ as described in section 3.3.2, 100 bootstrap samples of the source dataset were generated by uniform random sampling without replacement. Each sample contained 80% of the original dataset. In our experiments, θ refers to the number of hidden nodes of a neural network. We build models using neural network on each of the sampled datasets using a range of values of θ starting from 2 to 50. We obtain the values for θ for which we get the highest accuracy on each of the sample dataset. Figure 4.9 and 4.10 shows the histogram and the prior

distribution of θ for Supernova 2-class and Landmine source datasets.

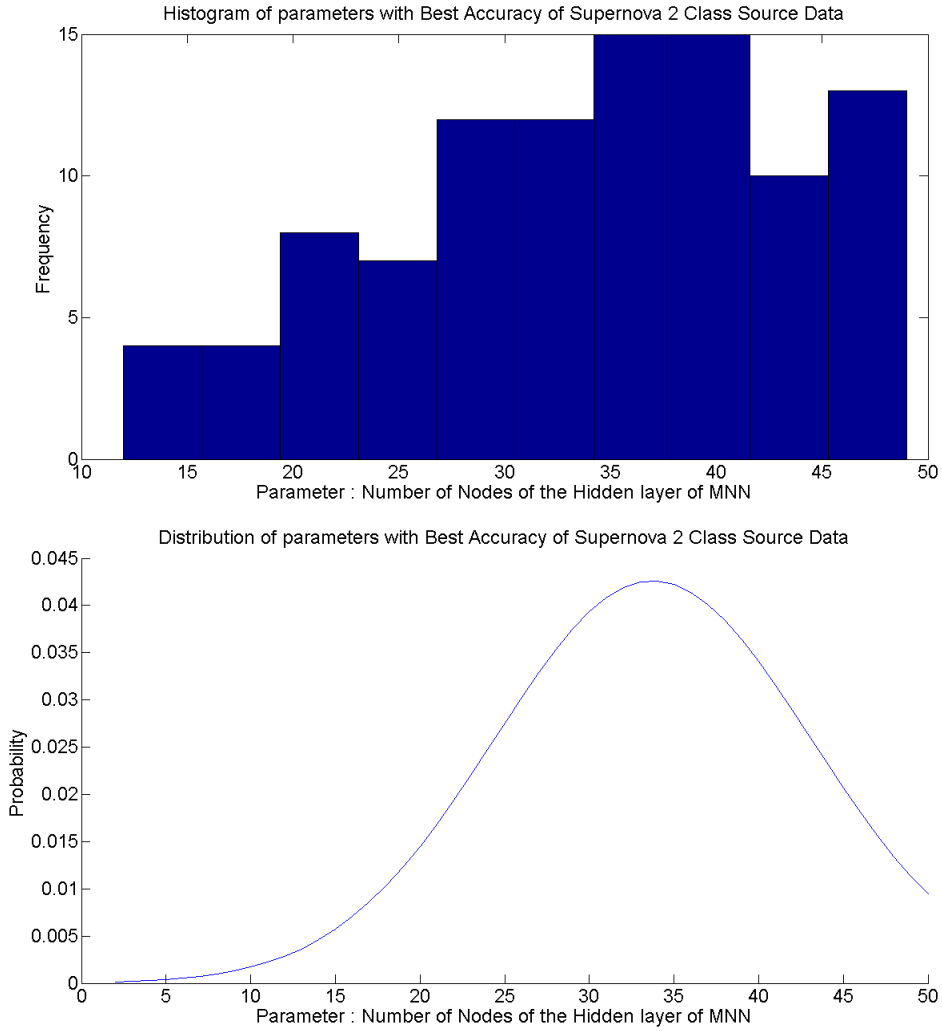

Figure 4.9: The histogram and and the prior distribution of θ for Supernova 2-class source dataset. Here θ refers to the number of hidden nodes of a neural network

The mean of the distribution denoted by μ_{tr} gives the optimal value θ^*_{tr} of the parameter θ in the source. From our experiments, we obtain μ_{tr}=33.75 and $\sigma_{tr} = 9.35$ for the Supernova source dataset, and μ_{tr}=23.59 and $\sigma_{tr} = 13.58$ for the Landmine

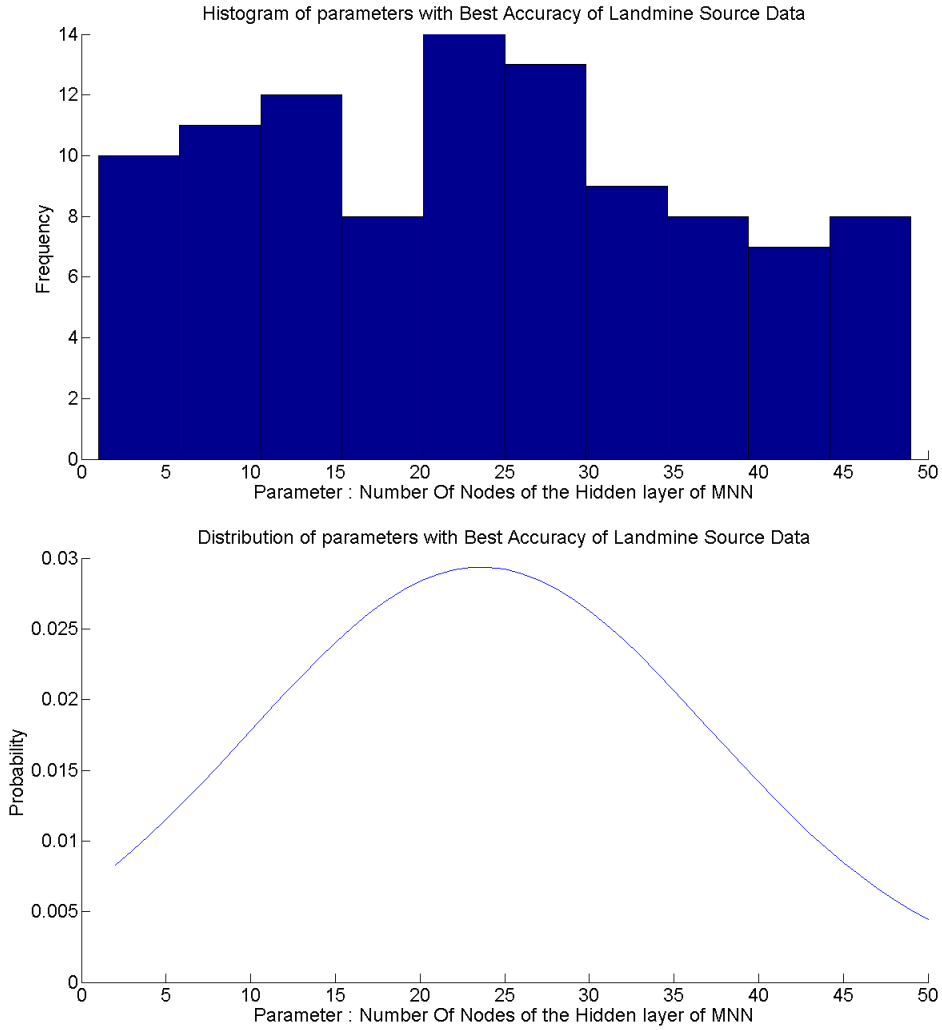

Figure 4.10: The histogram and and the prior distribution of θ for Landmine source dataset. Here θ refers to the number of hidden nodes of a neural network

source dataset. Here σ_{tr} refers to the standard deviation of the prior distribution.

4.2.3 Estimating the Likelihood of θ using Active learning

We use θ^*_{tr} obtained from the prior distribution to build models on the target dataset during active learning. We divide the target randomly in two parts: a pool of target instances, from which data will be queried, and a test set which will be unknown to the learning phase. For Supernova dataset the pool and the test dataset were of equal size, whereas the size of the pool of Landmine target dataset was 300 and the rest represented the test dataset. We generated 10 such pair of pools and test datasets from both the Supernova target dataset and the Landmine target datasets. We limit our budget to a minimum of 50 queries and increase it till 2000 for the Supernova target dataset.For Landmine dataset we limit our budget to a minimum of 50 queries and increase it till we use the entire pool of 300 instances.

After the active learning method stops, we build a model on the queried examples for each of the values of θ_{te} where θ^*_{tr}-$\sigma_{te} \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma_{te}$, where σ_{te}=σ_{tr}. For the Supernova 2-class dataset the value of θ_{te} varied from 24 to 43. For Landmine target dataset the value of θ_{te} varied from 10 to 37. We calculated the in-sample error E_{in} for each of the values of θ_{te} and thus calculated the likelihood of θ_{te} using the equation 3.17.

4.2.4 Estimating the Posterior of θ and Building the final model

We calculate the product of prior obtained in the section 4.2.2 and likelihood obtained in the section 4.2.3 to determine the best posterior probability of the parameter θ to get the optimal value θ^*_{te}. We then use θ^*_{te} to build the optimal model $f(x|\theta^*_{te})$ on the target domain. We present our results in the tables 4.10, 4.11, 4.12 and 4.13.

Table 4.10: Classification Accuracy on Supernova 2-class dataset using our proposed method of domain adaptation using active learning.Here we used θ^*_{tr} =33

Maximum Cost	Accuracy on Supernova 2-class target dataset		
	Using Prior $[\theta = 33]$	Using Posterior $[\theta \pm 1\sigma]$	Using Posterior $[\theta \pm 0.5\sigma]$
50	80.91 (0.66)	82.11 (0.31)	82.27 (0.45)
100	85.08 (0.55)	86.17 (0.35)	86.10 (0.37)
200	88.76 (0.37)	89.35 (0.37)	89.42 (0.37)
500	92.28 (0.18)	92.49 (0.18)	92.51 (0.19)
1000	93.41 (0.17)	93.57 (0.19)	93.57 (0.20)
2000	93.75 (0.22)	93.81 (0.22)	93.81 (0.23)

Table 4.11: Classification Accuracy on Supernova 2-class dataset using our proposed method of domain adaptation using active learning.Here we used θ^*_{tr} =34

Maximum Cost	Accuracy on Supernova 2-class target dataset		
	Using Prior $[\theta = 34]$	Using Posterior $[\theta \pm 1\sigma]$	Using Posterior $[\theta \pm 0.5\sigma]$
50	80.97 (1.17)	82.22 (0.89)	82.32 (0.75)
100	85.21 (0.29)	86.24 (0.37)	86.18 (0.32)
200	88.94 (0.25)	89.53 (0.27)	89.48 (0.20)
500	92.29 (0.16)	92.54 (0.17)	92.56 (0.19)
1000	93.48 (0.27)	93.59 (0.24)	93.60 (0.23)
2000	93.73 (0.22)	93.82 (0.21)	93.80 (0.23)

The first column in these tables refers to the maximum cost(or budget) assigned

Table 4.12: Classification Accuracy on Landmine dataset using our proposed method of domain adaptation using active learning.Here we used θ^*_{tr} =23

Maximum Cost	Accuracy on Landmine target dataset		
	Using Prior $[\theta = 23]$	Using Posterior $[\theta \pm 1\sigma]$	Using Posterior $[\theta \pm 0.5\sigma]$
50	90.92 (2.11)	91.13 (1.82)	90.79 (1.86)
100	91.88 (1.58)	91.99 (1.60)	91.80 (1.70)
150	92.30 (1.42)	92.40 (1.61)	92.39 (1.52)
200	92.69 (1.20)	92.80 (1.31)	92.78 (1.22)
250	93.06 (1.03)	93.12 (1.13)	93.12(1.07)
300	93.17 (1.08)	93.31 (1.03)	93.24 (1.06)

to the active learning method. The second column shows the classification accuracy of the target dataset using the best prior θ^*_{tr}. As $\theta^*_{tr} = 33.75$ for Supernova 2-class source dataset, we show results for both $\theta_{tr} = 33$ and $\theta_{tr} = 34$ in tables 4.10 and 4.11 respectively. Similarly for landmine dataset we show results for $\theta_{tr} = 23$ and $\theta_{tr} = 24$ in tables 4.12 and 4.13, since $\theta^*_{tr} = 23.59$ for Landmine source dataset. The third and fourth column corresponds to the accuracy on the target dataset using the proposed method as described in 3.3. The third column refers to the experiment when we use θ^*_{tr}-$\sigma_{te} \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma_{te}$ and the fourth column refers to the experiment where we use θ^*_{tr}-$0.5\sigma \leq \theta_{te} \leq \theta^*_{tr}$+$0.5\sigma$.We show the classification errors on Supernova and Landmine datasets in figures 4.11, 4.12, 4.13 and 4.14.

From these results we observe that using posterior probability of θ_{te} yields better classification accuracy on the target dataset than using the best prior θ^*_{tr}. We can also observe that even if we use half of the number of θ_{te} values to calculate the likelihood, we will still achieve a better accuracy than using the prior θ^*_{tr}. This leads us to an important conclusion: if the standard deviation σ_{tr} of the prior distribution $P_{tr}(\theta)$

Table 4.13: Classification Accuracy on Landmine target dataset using our proposed method of domain adaptation using active learning.Here we used θ^*_{tr} =24

Maximum Cost	Accuracy on Landmine target dataset		
	Using Prior $[\theta = 24]$	Using Posterior $[\theta \pm 1\sigma]$	Using Posterior $[\theta \pm 0.5\sigma]$
50	90.09 (2.59)	90.73 (1.97)	90.57 (2.06)
100	91.04 (1.60)	91.89 (1.58)	91.91 (1.72)
150	90.84 (1.58)	92.36 (1.72)	92.51 (1.56)
200	92.55 (1.52)	92.81 (1.19)	92.77 (1.29)
250	92.64 (1.10)	93.08(1.09)	93.11 (1.10)
300	92.71 (1.47)	93.26 (1.04)	93.27 (1.04)

is high, we can simple use the values of θ_{te} that lies within half a standard deviation of θ^*_{tr}. This will reduce the time and complexity of the algorithm considerably.

As the method is independent of the covariate shift assumption, it is very likely that this method will also work for datasets that show a covariate shift. Also, as the parameter θ is user defined, it can be chosen as per the requirement of the experiment. Thus we can conclude that our proposed method is most likely to yield better results in most scenarios of domain adaptation.

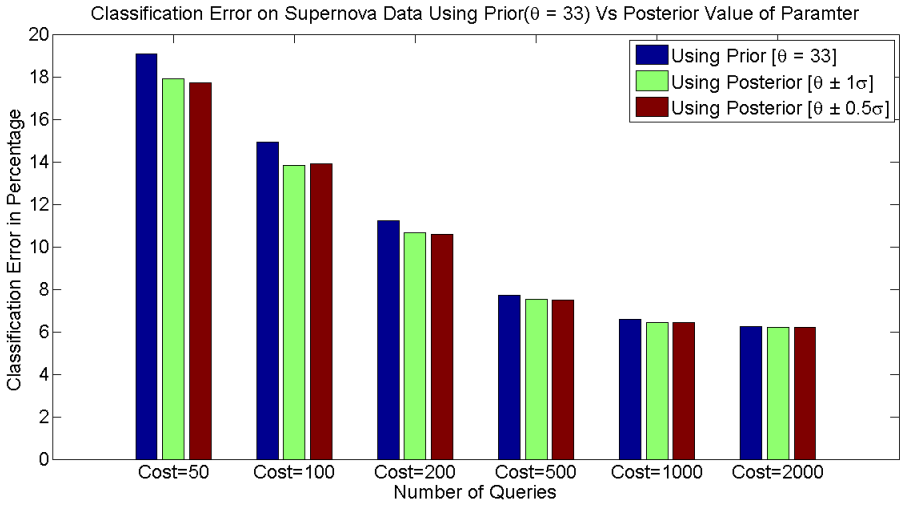

Figure 4.11: Classification error (in percentage) on Supernova 2-class dataset using prior θ_{te}=θ^*_{tr}=33 and posterior θ^*_{tr}-$\sigma \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma$

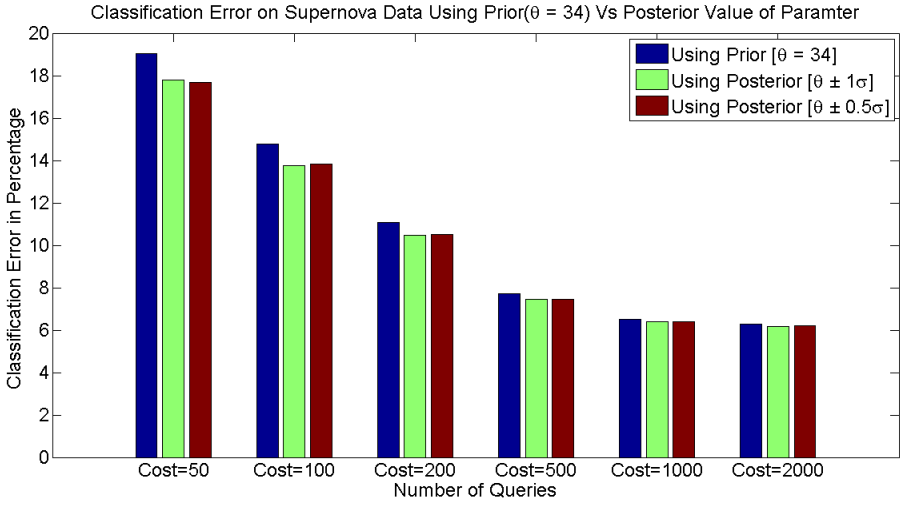

Figure 4.12: Classification error (in percentage) on Supernova 2-class dataset using prior θ_{te}=θ^*_{tr}=34 and posterior θ^*_{tr}-$\sigma \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma$

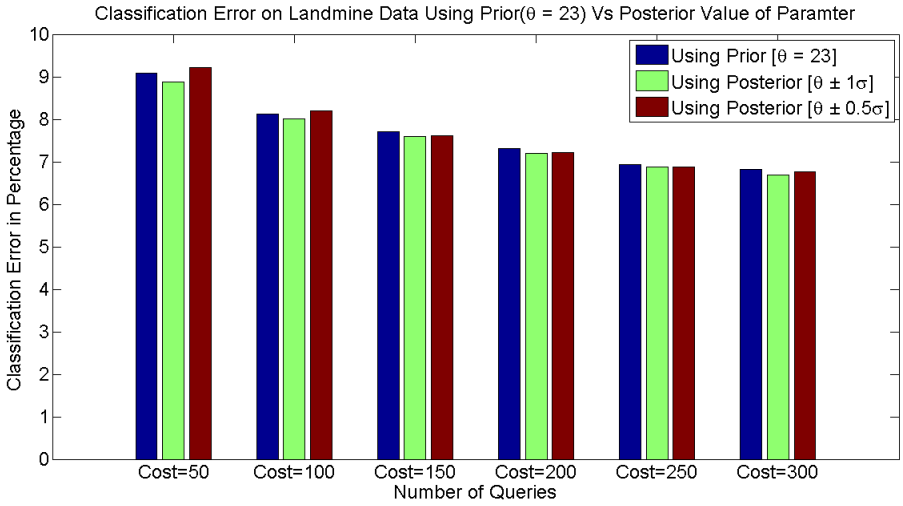

Figure 4.13: Classification error (in percentage) on Landmine dataset using prior θ_{te}=θ^*_{tr}=23 and posterior θ^*_{tr}-$\sigma \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma$

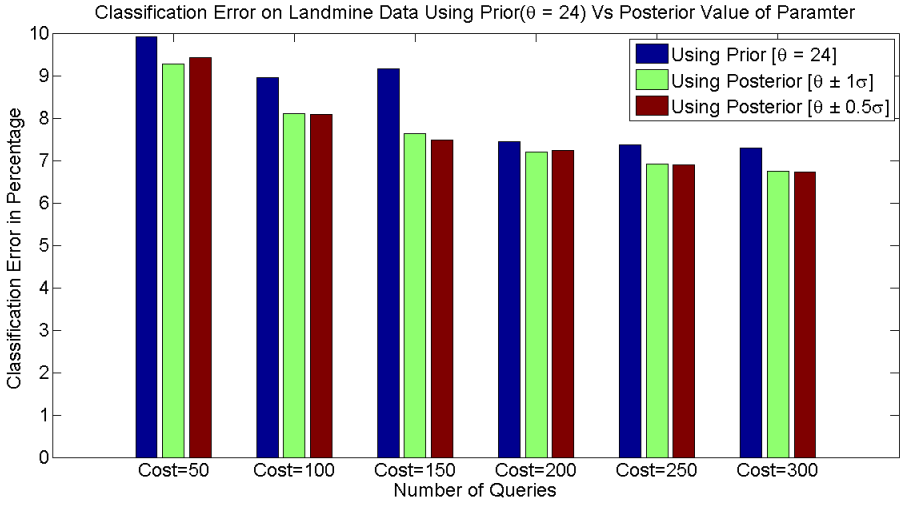

Figure 4.14: Classification error (in percentage) on Landmine dataset using prior θ_{te}=θ^*_{tr}=24 and posterior θ^*_{tr}-$\sigma \leq \theta_{te} \leq \theta^*_{tr}$+$\sigma$

Chapter 5

Conclusion and Future Work

5.1 Summary of Contributions

In this research we present two novel methods of domain adaptation. The first method is a new data-alignment method that can be applied to scenarios where the marginal distributions of the training and testing datasets differ by a linear shift. The second method that we proposed is more general in nature as it can be applied to all pairs of training and testing datasets whose marginal distributions are different irrespective of whether they have a covariate shift or not. However in this case we assume that the priors of the parameters of the models remain the same across the training(source) and testing(target) domains.

5.2 Conclusions and Limitations

From our experiments, we can conclude that our proposed data-alignment method, as described in 3.1, produces better classification accuracy than methods that use data without any alignment. We see that this method is well suited to classify the same type of stars across different galaxies. We also see that this method works better than the standard shift. The method is not only restricted to astronomy domain - it can be used in any other domains as well. One of the advantages of this method is that the source model can be directly used, removing the need of building a completely new model on the target domain. But the algorithm suffers from a limitation that the co-variate shift assumption must hold between the source and the target datasets and the change in prior distributions must be due to a linear shift, which restricts its usage to only certain types of datasets.

On the other hand, the domain adaptation method that uses active learning, as described in section 3.3, without building a model using the source instances or features directly, is much more general in nature and is more likely to yield optimal classifier on the target dataset in any domain adaptation scenarios. This method has a much broader scope of application and is not limited by the budget cost or the complexity of the model. Its only limitation is the number of executions needed in the experiment. One way to reduce this overhead is to restrict running the experiments with less number of parameter values both on source and on target based on the number of features of the dataset and the standard deviation of the marginal distribution of the parameter in question. This method can easily be incorporated

with any active learning algorithms combined with any domain adaptation scenarios.

5.3 Future work

Both of the methods proposed here can be expanded further to achieve better performance in complex scenarios. One way can be to use active learning with the data alignment method to detect the shift using very limited number of queries from the target dataset. Also the data alignment method can be used to classify the Cepheid stars across various other galaxies. Another future work may be to use this method to classify other type of starts whose light curves show similar properties.

In our research we used only one parameter for the second method. This can be expanded to a problem that searches for the optimal combination of values for a set of parameters. Also the method can be extended to other classes of models with a different set of parameters. Another aspect is to estimate an optimal range of the parameters both on source and target. Last but not he least, one direction of research will be to estimate the impact of the parameters on the classification accuracy.

Bibliography

[1] Ben-David, Shai, John Blitzer, Koby Crammer, and Fernando Pereira. "Analysis of representations for domain adaptation." Advances in neural information processing systems 19 (2007): 137.

[2] Ben-David, Shai, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. "A theory of learning from different domains." Machine learning 79, no. 1-2 (2010): 151-175.

[3] Quionero-Candela, Joaquin, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. Dataset shift in machine learning. The MIT Press, 2009.

[4] Blitzer, John, and Hal Daume III. Domain Adaptation, Tutorial.International conference in machine learning, 2010.

[5] Settles, Burr. "Active learning." Synthesis Lectures on Artificial Intelligence and Machine Learning 6, no. 1 (2012): 1-114.

[6] Shimodaira, Hidetoshi. "Improving predictive inference under covariate shift by weighting the log-likelihood function." Journal of statistical planning and inference 90, no. 2 (2000): 227-244.

[7] Bickel, Steffen, Michael Brckner, and Tobias Scheffer. "Discriminative learning for differing training and test distributions." In Proceedings of the 24th international conference on Machine learning, pp. 81-88. ACM, 2007.

[8] Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." IEEE Transactions on knowledge and data engineering 22, no. 10 (2010): 1345-1359.

[9] Sugiyama, Masashi, Shinichi Nakajima, Hisashi Kashima, Paul V. Buenau, and Motoaki Kawanabe. "Direct importance estimation with model selection and its application to covariate shift adaptation." In Advances in neural information processing systems, pp. 1433-1440. 2008.

[10] Huang, Jiayuan, Arthur Gretton, Karsten M. Borgwardt, Bernhard Schlkopf, and Alex J. Smola. "Correcting sample selection bias by unlabeled data." In Advances in neural information processing systems, pp. 601-608. 2006.

[11] Gretton, Arthur, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schlkopf. "Covariate shift by kernel mean matching." Dataset shift in machine learning 3, no. 4 (2009): 5.

[12] Saenko, Kate, Brian Kulis, Mario Fritz, and Trevor Darrell. "Adapting visual category models to new domains." In European conference on computer vision, pp. 213-226. Springer Berlin Heidelberg, 2010.

[13] Kulis, Brian, Kate Saenko, and Trevor Darrell. "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms." In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 1785-1792. IEEE, 2011.

[14] Blitzer, John, Ryan McDonald, and Fernando Pereira. "Domain adaptation with structural correspondence learning." In Proceedings of the 2006 conference on empirical methods in natural language processing, pp. 120-128. Association for Computational Linguistics, 2006.

[15] Fernando, Basura, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. "Unsupervised visual domain adaptation using subspace alignment." In Proceedings of the IEEE International Conference on Computer Vision, pp. 2960-2967. 2013.

[16] Bruzzone, Lorenzo, and Mattia Marconcini. "Domain adaptation problems: A DASVM classification technique and a circular validation strategy." IEEE transactions on pattern analysis and machine intelligence 32, no. 5 (2010): 770-787.

[17] Vilalta, Ricardo, Kinjal Dhar Gupta, and Lucas Macri. "A machine learning approach to Cepheid variable star classification using data alignment and maximum likelihood." Astronomy and Computing 2 (2013): 46-53.

[18] Vilalta, Ricardo, Kinjal Dhar Gupta, and Lucas Macri. "Domain Adaptation under Data Misalignment: An Application to Cepheid Variable Star Classification." In ICPR, pp. 3660-3665. 2014.

[19] Vilalta, Ricardo, Kinjal Dhar Gupta, and Ashish Mahabal. "Star Classification Under Data Variability: An Emerging Challenge in Astroinformatics." In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 241-244. Springer International Publishing, 2015.

[20] Ando, Rie Kubota, and Tong Zhang. "A framework for learning predictive structures from multiple tasks and unlabeled data." Journal of Machine Learning Research 6, no. Nov (2005): 1817-1853.

[21] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Domain adaptation for large-scale sentiment classification: A deep learning approach." In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 513-520. 2011.

[22] Blitzer, John, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. "Learning bounds for domain adaptation." In Advances in neural information processing systems, pp. 129-136. 2008.

[23] Argyriou, Andreas, Massimiliano Pontil, Yiming Ying, and Charles A. Micchelli. "A spectral regularization framework for multi-task structure learning." In Advances in neural information processing systems, pp. 25-32. 2007.

[24] Chen, Minmin, Kilian Q. Weinberger, and John Blitzer. "Co-training for domain adaptation." In Advances in neural information processing systems, pp. 2456-2464. 2011..

[25] Mansour, Yishay, Mehryar Mohri, and Afshin Rostamizadeh. "Domain adaptation with multiple sources." In Advances in neural information processing systems, pp. 1041-1048. 2009.

[26] Kumar, Abhishek, Avishek Saha, and Hal Daume. "Co-regularization based semi-supervised domain adaptation." In Advances in neural information processing systems, pp. 478-486. 2010.

[27] Lewis, David D., and William A. Gale. "A sequential algorithm for training text classifiers." In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 3-12. Springer-Verlag New York, Inc., 1994.

[28] Lewis, David D., and Jason Catlett. "Heterogeneous uncertainty sampling for supervised learning." In Proceedings of the eleventh international conference on machine learning, pp. 148-156. 1994.

[29] Scheffer, Tobias, Christian Decomain, and Stefan Wrobel. "Active hidden markov models for information extraction." In International Symposium on Intelligent Data Analysis, pp. 309-318. Springer Berlin Heidelberg, 2001.

[30] Japkowicz, Nathalie, and Shaju Stephen. "The class imbalance problem: A systematic study." Intelligent data analysis 6, no. 5 (2002): 429-449.

[31] Lin, Yi, Yoonkyung Lee, and Grace Wahba. "Support v ector machines for classification in nonstandard situations." Machine learning 46, no. 1-3 (2002): 191-202.

[32] Kubat, Miroslav, and Stan Matwin. "Addressing the curse of imbalanced training sets: one-sided selection." In ICML, vol. 97, pp. 179-186. 1997.

[33] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

[34] Zhu, Jingbo, and Eduard H. Hovy. "Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem." In EMNLP-CoNLL, vol. 7, pp. 783-790. 2007.

[35] Chan, Yee Seng, and Hwee Tou Ng. "Estimating class priors in domain adaptation for word sense disambiguation." In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 89-96. Association for Computational Linguistics, 2006.

[36] Chan, Yee Seng, and Hwee Tou Ng. "Word Sense Disambiguation with Distribution Estimation." In IJCAI, vol. 5, pp. 1010-5. 2005.

[37] Jiang, Jing. "A literature survey on domain adaptation of statistical classifiers." URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey (2008).

[38] Satpal, Sandeepkumar, and Sunita Sarawagi. "Domain adaptation of conditional probability models via feature subsetting." In European Conference on Principles of Data Mining and Knowledge Discovery, pp. 224-235. Springer Berlin Heidelberg, 2007.

[39] Daum III, Hal. "Frustratingly easy domain adaptation." arXiv preprint arXiv:0907.1815 (2009).

[40] Angluin, Dana. "Queries and concept learning." Machine learning 2, no. 4 (1988): 319-342.

[41] Cohn, David A., Zoubin Ghahramani, and Michael I. Jordan. "Active learning with statistical models." Journal of artificial intelligence research (1996).

[42] Settles, Burr. "Active learning literature survey." University of Wisconsin, Madison 52, no. 55-66 (2010): 11.

[43] King, Ross D., Kenneth E. Whelan, Ffion M. Jones, Philip GK Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. "Functional genomic hypothesis generation and experimentation by a robot scientist." Nature 427, no. 6971 (2004): 247-252.

[44] Atlas, Les E., David A. Cohn, Richard E. Ladner, Mohamed A. El-Sharkawi, Robert J. Marks, M. E. Aggoune, and D. C. Park. "Training Connectionist Networks with Queries and Selective Sampling." In NIPS, pp. 566-573. 1989.

[45] Cohn, David, Les Atlas, and Richard Ladner. "Improving generalization with active learning." Machine learning 15, no. 2 (1994): 201-221.

[46] Dagan, Ido, and Sean P. Engelson. "Committee-based sampling for training probabilistic classifiers." In Proceedings of the Twelfth International Conference on Machine Learning, pp. 150-157. The Morgan Kaufmann series in machine learning,(San Francisco, CA, USA), 1995.

[47] Krishnamurthy, Vikram. "Algorithms for optimal scheduling and management of hidden Markov model sensors." IEEE Transactions on Signal Processing 50, no. 6 (2002): 1382-1397.

[48] Yu, Hwanjo. "SVM selective sampling for ranking with application to data retrieval." In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pp. 354-363. ACM, 2005.

[49] Fujii, Atsushi, Takenobu Tokunaga, Kentaro Inui, and Hozumi Tanaka. "Selective sampling for example-based word sense disambiguation." Computational Linguistics 24, no. 4 (1998): 573-597.

[50] McCallumzy, Andrew Kachites, and Kamal Nigamy. "Employing EM and pool-based active learning for text classification." In Proc. International Conference on Machine Learning (ICML), pp. 359-367. 1998.

[51] Tong, Simon, and Daphne Koller. "Support vector machine active learning with applications to text classification." Journal of machine learning research 2, no. Nov (2001): 45-66.

[52] Hoi, Steven CH, Rong Jin, and Michael R. Lyu. "Large-scale text categorization by batch mode active learning." In Proceedings of the 15th international conference on World Wide Web, pp. 633-642. ACM, 2006.

[53] Thompson, Cynthia A., Mary Elaine Califf, and Raymond J. Mooney. "Active learning for natural language parsing and information extraction." In ICML, pp. 406-414. 1999.

[54] Settles, Burr, and Mark Craven. "An analysis of active learning strategies for sequence labeling tasks." In Proceedings of the conference on empirical methods in natural language processing, pp. 1070-1079. Association for Computational Linguistics, 2008.

[55] Zhang, Cha, and Tsuhan Chen. "An active learning framework for content-based information retrieval." IEEE transactions on multimedia 4, no. 2 (2002): 260-268.

[56] Yan, Rong, Jie Yang, and Alexander Hauptmann. "Automatically labeling video data using multi-class active learning." In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pp. 516-523. IEEE, 2003.

[57] Hauptmann, Alexander G., Wei-Hao Lin, Rong Yan, Jun Yang, and Ming-Yu Chen. "Extreme video retrieval: joint maximization of human and computer performance." In Proceedings of the 14th ACM international conference on Multimedia, pp. 385-394. ACM, 2006.

[58] Tur, Gokhan, Dilek Hakkani-Tr, and Robert E. Schapire. "Combining active and semi-supervised learning for spoken language understanding." Speech Communication 45, no. 2 (2005): 171-186.

[59] Liu, Ying. "Active learning with support vector machine applied to gene expression data for cancer classification." Journal of chemical information and computer sciences 44, no. 6 (2004): 1936-1941.

[60] Shannon, Claude Elwood. "A mathematical theory of communication." ACM SIGMOBILE Mobile Computing and Communications Review 5, no. 1 (2001): 3-55.

[61] Seung, H. Sebastian, Manfred Opper, and Haim Sompolinsky. "Query by committee." In Proceedings of the fifth annual workshop on Computational learning theory, pp. 287-294. ACM, 1992.

[62] Settles, Burr, Mark Craven, and Soumya Ray. "Multiple-instance active learning." In Advances in neural information processing systems, pp. 1289-1296. 2008.

[63] Roy, Nicholas, and Andrew McCallum. "Toward optimal active learning through monte carlo estimation of error reduction." ICML, Williamstown (2001): 441-448.

[64] Li, Shoushan, Yunxia Xue, Zhongqing Wang, and Guodong Zhou. "Active Learning for Cross-domain Sentiment Classification." In IJCAI. 2013.

[65] Persello, Claudio, and Lorenzo Bruzzone. "Active learning for domain adaptation in the supervised classification of remote sensing images." IEEE Transactions on Geoscience and Remote Sensing 50, no. 11 (2012): 4468-4483.

[66] Saha, Avishek, Piyush Rai, Hal Daum III, Suresh Venkatasubramanian, and Scott L. DuVall. "Active supervised domain adaptation." In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 97-112. Springer Berlin Heidelberg, 2011.

[67] Rai, Piyush, Avishek Saha, Hal Daum III, and Suresh Venkatasubramanian. "Domain adaptation meets active learning." In Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing, pp. 27-32. Association for Computational Linguistics, 2010.

[68] Cox, John P. "Theory of stellar pulsation." (1980).

[69] Persson, S. E., Barry F. Madore, W. Krzemiski, Wendy L. Freedman, M. Roth, and D. C. Murphy. "New Cepheid period-luminosity relations for the Large Magellanic Cloud: 92 near-infrared light curves." The Astronomical Journal 128, no. 5 (2004): 2239.

[70] Belokurov, Vasily, N. Wyn Evans, and Yann Le Du. "Light-curve classification in massive variability surveysI. Microlensing." Monthly Notices of the Royal Astronomical Society 341, no. 4 (2003): 1373-1384.

[71] Belokurov, Vasily, N. Wyn Evans, and Yann Le Du. "Light-curve classification in massive variability surveysII. Transients towards the Large Magellanic Cloud." Monthly Notices of the Royal Astronomical Society 352, no. 1 (2004): 233-242.

[72] Udalski, A. "The Optical Gravitational Lensing Experiment (OGLE): Bohdan's and Our Great Adventure." In The Variable Universe: A Celebration of Bohdan Paczynski, vol. 403, p. 110. 2009.

[73] Soszynski, I., R. Poleski, A. Udalski, M. K. Szymanski, M. Kubiak, G. Pietrzynski, L. Wyrzykowski, O. Szewczyk, and K. Ulaczyk. "The optical gravitational lensing experiment. the ogle-iii catalog of variable stars. i. classical cepheids in the large magellanic cloud." arXiv preprint arXiv:0808.2210 (2008).

[74] Pellerin, Anne, and Lucas M. Macri. "The M 33 Synoptic Stellar Survey. I. Cepheid Variables." The Astrophysical Journal Supplement Series 193, no. 2 (2011): 26.

[75] Stanek, K. Z., J. Kaluzny, M. Krockenberger, D. D. Sasselov, J. L. Tonry, and M. Mateo. "DIRECT Distances to Nearby Galaxies Using Detached Eclipsing Binaries and Cepheids. II. Variables in the Field M31ABased on observations collected at the Michigan-Dartmouth-MIT Observatory 1.3 m telescope and at the FL Whipple Observatory 1.2 m telescope." The Astronomical Journal 115, no. 5 (1998): 1894.

[76] Macri, L. M., K. Z. Stanek, D. D. Sasselov, M. Krockenberger, and J. Kaluzny. "DIRECT Distances to Nearby Galaxies Using Detached Eclipsing Binaries and Cepheids. VI. Variables in the Central Part of M33Based on observations collected at the Fred L. Whipple Observatory 1.2 m telescope and at the Michigan-Dartmouth-MIT 1.3 m telescope." The Astronomical Journal 121, no. 2 (2001): 870.

[77] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. "The WEKA data mining software: an update." ACM SIGKDD explorations newsletter 11, no. 1 (2009): 10-18.

[78] Madore, Barry F., and Wendy L. Freedman. "The Cepheid distance scale." Publications of the Astronomical Society of the Pacific 103, no. 667 (1991): 933.

[79] Faria, Susana, and Gilda Soromenho. "Fitting mixtures of linear regressions." Journal of Statistical Computation and Simulation 80, no. 2 (2010): 201-225.

[80] Gupta, Kinjal Dhar, Ricardo Vilalta, Vicken Asadourian, and Lucas Macri. "Adapting Predictive Models for Cepheid Variable Star Classification Using Linear Regression and Maximum Likelihood." Proceedings of the International Astronomical Union 10, no. S306 (2014): 319-321.

[81] Bartlett, Peter L., and Wolfgang Maass. "Vapnik Chervonenkis Dimension of Neural Nets." The handbook of brain theory and neural networks (2003): 1188-1192.

[82] Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin. Learning from data. Vol. 4. Singapore: AMLBook, 2012.

[83] Kessler, Richard, Alex Conley, Saurabh Jha, and Stephen Kuhlmann. "Supernova Photometric Classification Challenge." arXiv preprint arXiv:1001.5210 (2010).

[84] Ishida, Emille EO, and Rafael S. de Souza. "Kernel PCA for Type Ia supernovae photometric classification." Monthly Notices of the Royal Astronomical Society 430, no. 1 (2013): 509-532.

[85] Xue, Ya, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. "Multi-task learning for classification with Dirichlet process priors." Journal of Machine Learning Research 8, no. Jan (2007): 35-63.

[86] Schlkopf, Bernhard, Alexander Smola, and Klaus-Robert Mller. "Kernel principal component analysis." In International Conference on Artificial Neural Networks, pp. 583-588. Springer Berlin Heidelberg, 1997.