Indian Statistical Institute, Kolkata

M. Tech. (Computer Science) Dissertation

Transfer Learning Using MMD

A dissertation submitted in partial fulfillment of the requirements
for the award of Master of Technology
in
Computer Science

Author:
Rahul Anand
Roll No: CS-1403

Supervisor:
Dr. N.R PAL
ECSU Unit, ISI

M.Tech(CS) DISSERTATION THESIS COMPLETION CERTIFICATE

Student: Rahul Anand (CS1403)

Topic: Transfer Learning Using MMD

Supervisor: Dr. N.R PAL

This is to certify that the thesis titled "***Transfer Learning Using Maximum Mean Discrepancy***" submitted by **Rahul Anand** in partial fulfillment for the award of the degree of Master of Technology is a bonafide record of work carried out by him under my supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission. The results contained in this thesis have not been submitted to any other university for the award of any degree or diploma.

Date: Dr.N.R PAL

Dedication

To my parents and my well wishers, without your help and encouragement it would not have been possible.

Acknowledgements

I would like to thank my dissertation supervisor Dr. N.R PAL for agreeing to guide me and for helping me to undertake work in the topic.

Last but not the least I am grateful to Prasenjit Da and all my lab mates and seniors of The computational Intelligence lab of Indian Statistical Institute , Kolkata for helping me though out the project with their valuable suggestions.

Abstract

In standard machine learning tasks, the domain of the data on which a classifier learns and the domain on which it predicts, are usually the same. However, this assumption may not always hold true. In such cases, we usually transfer the knowledge learnt from one domain to design a classifier on a related domain and this task of knowledge transfer is termed as *Transfer Learning*. There are many facets of Transfer Learning, many authors use some labeled data from the target domain (test domain) in addition to the labeled data from the source domain. There can be several different scenarios where knowledge may be transferred. In our method we are trying to minimize simultaneously the MMD (Maximum Mean Discrepancy) which measures the distance between means of two the domains after mapping the data in a higher dimension by some non-linear transformation and classifier objective function. In this work, our purpose is to transfer knowledge between a source domain and a target domain which lie in the same feature space but have different distributions. In the literature we could not find any work which optimizes the kernel parameters with respect to MMD. Here we first investigate how effective is optimization of kernel parameters and the minimization of MMD to achieve better performance in Transfer Learning.

Our investigation reveals that lower MMD does not necessarily mean better classifier performance on the target domain. It also suggests existence of multiple local minima with almost equal value of MMD. Then we moved to our main problem of building a classifier for the target domain. Various authors have used minimization of MMD using multiple kernels to find a suitable latent space. Typically, a convex combination of the kernels is used and weights of the combination are learnt to minimize the MMD. In our method, we cluster both source and target domain data into a predefined number of clusters and then we establish correspondence between source and target domain clusters using the *Hungarian Algorithm*. Finally, we minimize the MMD on corresponding pairs of cluster for obtaining a latent space that represents the source and target data in a better manner for solving our problem. Our method shows some improvement in performance when there are good cluster structure in the source and target domains. In our approach we do not use any label information from the target data.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

When we use data to design a system, it can be a forecasting system or a classification system, it is assumed that the test data and train data are generated from the same distribution. When this assumption fails we have to start the process from scratch. We have to again collect data and build the model. Collection and annotation of data are very expensive and time consuming in real world. There are application scenarios where plenty of un-annotated(unlabeled) data are available and data labeling is quite expensive. The domain in which we have labeled data will be called source domain and a related domain where we have a lot of unlabeled data will be called Target domain. It would be very useful if we can design a decision making system using labeled source domain data and unlabeled Target domain data and use the same in the target domain successfully. Note that the target domain data may be from a different application domain or from the same application domain but the data have a different distributional characteristics than that of the source domain labeled data. In such cases, *knowledge* *transfer or transfer learning* between domains would be desirable. Transfer Learning is very important in areas, where it is very difficult to label a large amount of data but to build a good model we need a good amount of label data. So Transfer Learning makes our life simple. Research on Transfer Learning has attracted more and more attention since 1995 in different names: Learning to learn, Life-long learning, Knowledge consolidation, Context sensitive learning, Knowledge-based inductive bias, Meta learning, and Incremental learning[14, 13, 12]. Among these, a closely related learning technique to transfer learning is the multi-task learning framework[11], which tries to learn multiple tasks simultaneously even when they are different. A typical approach for multi-task learning is to uncover the common(latent) features that can benefit each individual task. This philosophy is often used in Transfer Learning.

1.1 Category of Transfer Learning

In Transfer Learning, there are three main issues: (1) What to Transfer (2) How to Transfer (3) When to transfer:
"What to transfer" deals with which knowledge should be transferred across domains, There may be some features which are completely different across domains but we should look for the knowledge which is common across domains so that they may help in improving the performance. Then we should look for the algorithm which can transfer the knowledge which comes under "How to Transfer".

"When to Transfer" deals with situations in which knowledge transfer

should be performed. When we perform knowledge transfer, data in different domains must be related. If the data are not related and we try to transfer knowledge, it may lead to negative transfer, which may hamper the performance in term of accuracy.
Transfer Learning framework can be divided into several groups:

1. In the *inductive transfer learning* setting, the target task is different from the source task, no matter whether the source and target domains are the same or not. In this case, some labeled data in the target domain are required to *induce* an objective predictive model $f_T(.)$ for use in the target domain.

2. In the *transductive transfer learning* setting, the source and target domains are different.
   In this situation, no labeled data in the target domain is available while a lot of labeled data available in the source domain. In addition, according to different situations between the source and target domains, we can further categorize the *transductive transfer learning* setting in two cases:
   The features spaces between the source and target domains are different, $\mathcal{X}_s \neq \mathcal{X}_T$
   The feature space between the source and the target domains are the same, $\mathcal{X}_s = \mathcal{X}_T$, but the marginal probability distributions of the input data are different, $P(\mathcal{X}_s) \neq P(\mathcal{X}_s)$.

3. In the *unsupervised transfer learning* setting, similar to *inductive trans-*

fer learning setting, the target task is different from but related to the source task. However, the *unsupervised transfer learning* focus on solving unsupervised learning tasks in the target domain, such as clustering, dimensionality reduction and density estimation. In this case, there are no labeled data is available in the both source and target domains for the training.

Here we shall be concentrating on *transductive transfer learning* and in this case feature sets are the same in both domains , i.e. , $\mathcal{X}_s = \mathcal{X}_T$ but the distribution characteristics are different, i.e., $P(\mathcal{X}_s) \neq P(\mathcal{X}_s)$. We shall not use any label of the target data during training.

1.2 Motivation

There are large number of areas in which Transfer Learning can be applied. Transfer Learning is very useful in applications where data get outdated very quickly. In this case data with same label may have different distribution during two different time periods. For instance WiFi localization problems, in which it is tried to estimate the location of the user based on previously collected WiFi data. It is cumbersome to calibrate the WiFi data for building localization because a user needs to label a large collection of WiFi signal data at each location.WiFi signal strength may vary with device,time or other factors. A model build at one time and used at another time may reduce its performance. To reduce the task of re-calibration, we can adapt the localization model trained at one time period(source domain) for a new time period(the target domain), or adapt localization model trained on a particular type of mobile device for a new mobile device [10].

Another example where Transfer Learning has been be applied is sentiment analysis. Here the task is to automatically classify the reviews of a product as positive or negative responses. For this first we have to collect a lot of reviews of the product and manually annotate (label) them. After this we need to train a classifier using the labeled data. If we design a classifier to analyze reviews for one product and use it on another product, it may not work because the distribution of review data is usually different for different products. So to maintain good classification performance, we need to collect a large amount of data for each product and label the data to train a classifier for each product. This labeling can be cumbersome. To make life simple, we might want to adapt a classification system that is trained on some product say, to help learn classification model for some other product say B. In such case Transfer Learning can play a vital role[1].

Transfer Learning has many other applications for example, it can be applied in Web Document classification. Here the goal is to classify a given Web document into several predefined categories. For a classification task on a newly created Web site where the data distribution may be different, there may be lack of labeled training data. So we cannot apply an existing web-page classifier directly on the new Websites. In such cases, it would be helpful if we could transfer the knowledge of an existing classifier into the new domain.

Recall that for transfer learning application areas, the feature distribution of training samples usually vary significantly across the domains and the training samples from different sources have different statistical properties.

Although large number of training data may be available in the source domain the performance of classifier build on the training data is likely to be poor on the target domain. So a challenging task in this whole process is how to reduce the distribution gap between source domain and target domain primarily projecting the data into new latent space. Intuitively, discovering a good feature representation across the domain is crucial. A good feature representation should be able to reduce the difference in distributions between domains as much as possible, while at the same time preserving important properties (such as geometric properties , statistical properties of the data[9]) so that the projected data can discriminate between classes.

1.3 Our work

Various authors have used Minimization of MMD using multiple kernels to find the suitable latent space[5]. Typically, a convex combination of the kernels is used and the weight of the combination is minimized. Here we first find the learning rule for minimizing the MMD with respect to the kernel parameters and study the effectiveness of the optimal kernel parameters on Transfer Learning. In the second part of our investigation we cluster both source and target data into predefined number of cluster and then apply MMD on corresponding pairs of clusters with a view to obtaining a latent space that represent the source and target data in a better manner for the purpose of solving our problem.

Chapter 2

Related works

Definition 1 (Transfer Learning) Given a source domain D_s and learning task T_s, a target domain D_T and learning task T_T, transfer learning aims to help in improving the learning of the target predictive function $f_t(.)$ in D_T using the knowledge in D_s and T_s where $D_s \neq D_T$ or $T_s \neq T_T$

In order to exploit the information of the data form both source domain and target domain, Daume [3] suggested a Feature Replication (FR) method to augment features of both source and target domains. Here they use three versions of each feature: a general version, a source version and a target specific version for transfer learning. Now these augmented features are used to construct a kernel function for support vector machine(SVM). Yang *et al*[17] proposed an Adaptive SVM to enhance the prediction performance of video concept detection, in which a new SVM classifier $f_T(\mathbf{x})$ is adapted from the existing classifier $f_A(\mathbf{x})$ trained on the source domain. Following this work cross domain SVM proposed by by Jiang *et al.* [9] used k-nearest neighbours from the target domain to define a weight for each source pattern and the SVM classifier was trained with re-weighted patterns. All these methods require some labeled patterns in the target domain and they do not use unlabeled Target domain data.

In a situation when there are very less labeled data for target domain or no labeled data then the classifier can be built only on the source domain data. There are several cross domain learning methods which cope with the inconsistency in data distribution between source and target domains and design classifiers using source data and unlabeled target data. Some of these methods re-weight the patterns from source domain with the help unlabeled patterns from the target domain [5].

We will briefly review the two main paradigms of cross-domain learning. The first one directly learns decision functions for the target domain based on the labeled source domain data by minimizing the mismatch of data distribution between the two domains. The second approach makes use of existing source domain classifiers based on the source domain data which is discussed in next two subsection.

2.1 Reduce the mismatch between distributions

In transfer learning it is very important to reduce the the mismatch between distribution in the source and target domains. In order to reduce the mismatch one of the ways is to estimate the distribution of each domains. But to avoid such a non-trivial task, Borgwardt [2] proposed an effective and non parametric criterion named as maximum mean discrepancy(MMD), to compare the data distribution based on the distance between the mean of samples from two domains in a kernel k induced Reproducing Kernel Hilbert space. Square of MMD is given by

$$\mathrm{DIST}_k(D^A, D^T) = \left\| \frac{1}{n_A} \sum_{i=1}^{n_A} \phi(\boldsymbol{x}_i^A) - \frac{1}{n_T} \sum_{i=1}^{n_T} \phi(\boldsymbol{x}_i^T) \right\|^2 . \tag{2.1}$$

Later we will justify the choice of this distance measure. In order to capture the higher statistics of the data, the samples used in (2.1) are transformed into higher dimensions or even infinite dimension through a non linear feature mapping $\phi(.)$ [5]. When $DIST_k(D^A, D^T)$ approaches to zero, the higher order moments of the data from the two domains become closer, so their data distribution become also close to each other[5]. For reducing the mismatch between he two distributions Huang et al. [8] suggested a two step approach called Kernel Mean Matching(KMM). The first step is used to diminish the mismatch between the means of the samples in RKHS from the two domains by re-weighting the samples $\phi(\mathbf{x}_i)$ in the auxiliary domain as $\boldsymbol{\beta_i}\phi(\boldsymbol{x}_i)$ In [8] $\boldsymbol{\beta}$ is learned by MMD in (2.1). Then in the second step authors learnt a decision function $f(\boldsymbol{x})=\boldsymbol{W}'\phi(\boldsymbol{x})+b$ that separates the patterns from two opposite classes in D^A using the loss function re-weighted by $\boldsymbol{\beta}_i$.

2.2 Modifying the Auxiliary Classifier

Instead of explicitly building a classifier for the target domain, some researchers make use of pre-learned classifiers built from the auxiliary data set to learn the target classifier. Yang et al.[17] gave an Adaptive SVM in which a new svm classifier $f^T(\boldsymbol{x})$ was adapted from auxiliary classifier $f^A(\boldsymbol{x})$ trained with the patterns from the auxiliary domain. In the adaptive SVM classifier, the target classifier $f^T(\boldsymbol{x})$ is adapted from the existing auxiliary classifier $f^A(\boldsymbol{x})$ as $f^T(\boldsymbol{x}){=}f^A(\boldsymbol{x}){+}\Delta f(\boldsymbol{x})$ where $\Delta f(\boldsymbol{x})$ is learned from the labeled target data. This method also assumed existence of some labeled data for the target domain.

2.3 Invariant and Variant features across domains

We can learn the invariant features across the two domains and once we are done with learning of invariant features we can build our model on the basis of these invariant features. Since these features in both source and target domains are not varying this may improve the accuracy of prediction for the target domain [15]. In [15] authors tried to find features which contribute to the distance between two domains most. Here authors take a diagonal matrix W where the diagonal elements define the feature weights. Let $W \in \mathbb{R}^{d\times d}$ be a diagonal weight matrix and $\boldsymbol{x} \in \mathbb{R}^d$ be a d dimensional sample vector. Authors in [15] form a convex optimization function which returns the weight matrix W whose diagonal elements indicate the strength of features. In determining the separation between the two domains they use a threshold. If a diagonal element is greater than the threshold then that feature is considered a variant feature across the two domains and if the diagonal element is less then the threshold then that feature is considered an invariant feature. Then a classifier is built on invariant features. To illustrate this with an example, let $\phi(\boldsymbol{x}) : \boldsymbol{X} \rightarrow \mathcal{H}$ be a polynomial kernel with degree, d. Then, $K^{'} = [k]_{(n_A+n_T)\times(n_A+n_T)}$, where $k_{ij} = (\boldsymbol{x_i}^{'} W \boldsymbol{x_j} + 1)^d$. The convex optimization problem is formulated as follows. [15]

$$\begin{gathered} W^* = \underset{w}{argmin} - trace(K'L) \\ \text{subject to } diag(W)^T * diag(W) \leq 1 \\ W > 0 \end{gathered}$$

where L = $\begin{bmatrix} [A]_{n_A\times n_A} & [B]_{n_A\times n_T} \\ [C]_{n_T\times n_A} & [D]_{n_T\times n_T} \end{bmatrix}$

$A_{ij} = \frac{1}{n_A^2}$, $B_{ij} = \frac{-1}{n_A\times n_T}$, $C_{ij} = \frac{-1}{n_A\times n_T}$, $D_{ij} = \frac{1}{n_T^2}$. Here $K^{'}$ is a matrix which contains the each data transformed by polynomial kernel. n_A and n_T are the number of point in the source domain and target domain respectively.

2.4 Simultaneous Reduction of MMD and learning of classifier

Another method followed by researchers to solve the problem to minimize the MMD and and learn the classifier simultaneously through multiple kernel learning method[4,11].In this method in order to reduce the distribution mismatch it is tried to bring MMD of the two distributions close in a higher dimension through a non linear feature mapping $\phi(.)$. Since data from each class may be distributed into several clusters, instead of minimizing the MMD of the entire data set, we cluster the source and target data into a predefined number of clusters and then establish a correspondence between clusters in the two domains and the sum of MMD's over all corresponding pairs of cluster is minimized. In our method instead of bringing only one mean of source and target close we are trying to bring multiple means close enough by clustering source and target data so that the chances of realizing a better latent space increases. This is also expected to better maintain the ”geometry” of the original data in the latent space. However, if the data do not have well defined clusters, this may not give any advantage.

Chapter 3

Proposed Method

Before moving on to the mathematical formulation of our problem, we first discuss the necessary mathematical background. We discuss the MMD and the method to calculate it. We will also discuss about the Hungarian matching algorithm. The necessary notations for the discussion are given below in Table 3.1

Table 3.1: Notations used in the chapter

Notations	**Meaning**
A	Source Domain
T	Target Domain
n_A	number of samples in Source Domain
n_T	number of samples in Target Domain
$\boldsymbol{X}_s$	labeled source data set
$\boldsymbol{X}_T$	unlabeled target data set

3.1 MMD and Hilbert Space

Hilbert Space $\mathcal{H}$: A Hilbert space is a vector space on which an inner product is defined, along with an additional technical condition, the space must contain the limit of all Cauchy sequences of functions.
Cauchy Sequence: A sequence $x = (x_1, x_2....., x_n)$ is a Cauchy sequence if $\forall\ \epsilon > 0\ \exists\ \mathrm{N} \in \mathbb{N}$ such that $n, m > N \Rightarrow d(x_n, x_m) < \epsilon$.
Kernel Let $\mathcal{X}$ be a nonempty set, function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if $\exists$ $\mathbb{R}$-Hilbert space $\mathcal{H}$ and a map $\phi : \mathcal{X} \to \mathcal{H}$ such that $\forall \boldsymbol{x}, \boldsymbol{x'} \in \mathcal{X}\ k(\boldsymbol{x}, \boldsymbol{x'}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x'}) \rangle$.
Reproducing Kernel: Let $\mathcal{H}$ be a Hilbert space on $\mathcal{X}$, $k(.,.)$: $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *reproducing kernel* of Hilbert space $\mathcal{H}$ if $\forall f \in \mathcal{H}$, $\forall x \in \mathcal{X}$, the following two condition is satisfied:

1. The feature map of every point is in the feature space: $\forall \boldsymbol{x} \in \mathcal{X}, k(., \boldsymbol{x}) \in \mathcal{H}$
2. The Reproducing property : $\forall \boldsymbol{x} \in \mathcal{X}, \forall f \in \mathcal{H}\ f(\boldsymbol{x}) = \langle k(\boldsymbol{x}, .), f(.) \rangle_{\mathcal{H}}$

RKHS is a Hilbert space with a reproducing kernel. The space of such function is known as Reproducing Kernel Hilbert Space(RKHS).
Two defining features of **RKHS**

1. The feature map of every point is in the feature space: $\forall \boldsymbol{x} \in \mathcal{X}, k(., \boldsymbol{x}) \in \mathcal{H}$
2. The Reproducing property : $\forall \boldsymbol{x} \in \mathcal{X}, \forall f \in \mathcal{H}, f(\boldsymbol{x}) = \langle k(\boldsymbol{x}, .), f(.) \rangle$

In particular $k(\boldsymbol{x}, \boldsymbol{y}) = \langle k(., \boldsymbol{x}), k(., \boldsymbol{y}) \rangle$

3.2 MMD

Our target is to minimize the mismatch between two distributions, source and target in a latent space. In order to avoid the estimation of density for target and source domains, we calculate the MMD(Maximum Mean Discrepancy), which measures the distance between the means of the two distributions in higher dimensions. So if we can reduce MMD in some projected space, it will lead to reduction in mismatch between two distributions in the projected space.

Suppose, we have two data sets generated from distributions p and q, then MMD is defined by

$$\mathcal{D}(p, q, \mathbb{F}) = \sup_{f \in \mathbb{F}} \{E_p f(\boldsymbol{x}) - E_q f(\boldsymbol{y})\} \tag{3.1}$$

where E_p is the expectation of the data transformed by function f under the distribution p.

Theorem1

$\mathcal{D}(p, q, \mathbb{F}) = 0$ iff $p = q$ where $\mathbb{F} = \{f | \ \|f\|_{\mathcal{H}} \leq 1\}$ is a unit ball in Reproducing Kernel Hilbert space,provided that $\mathcal{H}$ is universal.

In RKHS function evaluation is given by $f(\boldsymbol{x}) = \langle K(\boldsymbol{x}, .), f \rangle$

Now the mean is

$$\begin{aligned} E_p[f(\boldsymbol{x})] &= E[\langle K(\boldsymbol{x}, .), f\rangle] \\ &= \langle E[K(\boldsymbol{x}, .)], f\rangle \\ &= \langle \frac{1}{m}\sum_{k=1}^{m} K(\boldsymbol{x}_k, .), f\rangle \\ &= \langle \mu_{\boldsymbol{x}}, f\rangle \end{aligned}$$

so equation(3.1) can be written as [7]

$$\begin{aligned} \mathcal{D}(p, q, \mathbb{F}) &= \sup_{\|f\|\leq 1} [\langle \mu_x, f\rangle - \langle \mu_y, f\rangle] \\ &= \sup_{\|f\|\leq 1} \langle \mu_p - \mu_q, f\rangle \\ &= ||\mu_p - \mu_q||_{\mathcal{H}} \end{aligned}$$

3.3 MMD and Kernel

Using the principal of RKHS, now we will show how the square of MMD can be written in the form of a kernel. We have already seen

$$\mathcal{D}(p, q, \mathbb{F}) = ||\mu_p - \mu_q||_{\mathcal{H}}$$

Now we will show how the square of MMD estimated from finite sample can be written in the form of kernel.

$$\begin{aligned}
&\|\tfrac{1}{n_A}\textstyle\sum_{i=1}^{n_A}\phi(\boldsymbol{x}_i^A)-\tfrac{1}{n_T}\sum_{j=1}^{n_T}\phi(\boldsymbol{y}_j^T)\|^2\\
&=\langle\tfrac{1}{n_A}\textstyle\sum_{i=1}^{n_A}\phi(\boldsymbol{x}_i^A)-\tfrac{1}{n_T}\sum_{j=1}^{n_T}\phi(\boldsymbol{y}_j^T),\\
&\tfrac{1}{n_A}\textstyle\sum_{i=1}^{n_A}\phi(\boldsymbol{x}_i^A)-\tfrac{1}{n_T}\sum_{j=1}^{n_T}\phi(\boldsymbol{y}_j^T)\rangle\\
&=\tfrac{1}{n_A^2}\langle\textstyle\sum_{i=1}^{n_A}\phi(\boldsymbol{x}_i^A),\sum_{i=1}^{n_A}\phi(\boldsymbol{x}_i^A)\rangle\\
&-\tfrac{2}{n_A*n_T}\langle\textstyle\sum_{i=1}^{n_A}\phi(\boldsymbol{x}_i^A),\sum_{j=1}^{n_T}\phi(\boldsymbol{y}_j^T)\rangle\\
&+\tfrac{1}{n_T^2}\langle\textstyle\sum_{j=1}^{n_T}\phi(\boldsymbol{y}_j^T),\sum_{j=1}^{n_T}\phi(\boldsymbol{y}_j^T)\rangle\\
&=\tfrac{1}{n_A^2}\textstyle\sum_{i=1}^{n_A}\sum_{j=i}^{n_A}K(\boldsymbol{x}_i^A,\boldsymbol{x}_j^A)\\
&+\tfrac{1}{n_T^2}\textstyle\sum_{i=1}^{n_T}\sum_{j=i}^{n_T}K(\boldsymbol{y}_i^T,\boldsymbol{y}_j^T)\\
&-\tfrac{2}{n_T*n_A}\textstyle\sum_{i=1}^{n_A}\sum_{j=1}^{n_T}K(\boldsymbol{x}_i^A,\boldsymbol{y}_j^T)\\
&=trace(K*L)
\end{aligned}$$

where L is same as defined in previous chapter in section 2.3.

As we have discussed previously instead of minimizing the MMD of the source and Target data, We first cluster the source and Target data into a predefined number of clusters. Then we establish correspondence between clusters in the two domains and the sum of MMD's over all corresponding pairs of cluster is minimized. If the source and target data have well defined cluster structure this is likely to find better kernels. Once we find the clusters in $\boldsymbol{X}_s$ and $\boldsymbol{X}_T$, We need to establish a correspondence between clusters in the two domains. For correspondence we are using *Hungarian algorithm.* We shall learn the classifier and reduce the MMD(Maximum Mean Discrepancy) simultaneously [5].

But before that we want to investigate a more basic problem. Several authors have used multiple kernels with different predefined values of the kernel parameters and then made a convex combination of those kernels. In order to minimize MMD, they learnt the weights of the linear combination. But they did not optimize the kernel parameters. Here we first see how effective is the optimization of kernel parameters to minimize MMD to achieve better performance in transfer learning.

3.4 optimization of MMD with repect to the kernel parameters

We tried to optimize the MMD with respect to all the kernel parameters. We used three kernels, RBF kernel $\exp(-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2)$, inverse kernel $\frac{1}{\gamma\|\boldsymbol{x}-\boldsymbol{y}\|+1}$, and square inverse kernel $\frac{1}{\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2+1}$. We also learned the behaviour of kernel parameters with the iterations. We did this experiment by selecting different set of $\boldsymbol{d}$ and we are not optimizing $\boldsymbol{d}$. We are studying the behaviour with a fixed $\boldsymbol{d}$. The optimization function is as follows:

$$J(\boldsymbol{\gamma}) = \min_{\boldsymbol{\gamma}} \sum_{m=1}^{M} trace(\boldsymbol{d}_m K_m L). \tag{3.2}$$

We use gradient descent method for the optimization.

$$\boldsymbol{\gamma}_{t+1}^{i} = \boldsymbol{\gamma}_{t}^{i} - \eta \frac{d(J(\boldsymbol{\gamma^i}))}{d(\gamma^i)}$$

Algorithm 1 MMD With Kernel Parameter

```
1: initialize kernel parameters $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2 ... \gamma_l\}$ for each kernel, l = Number of kernels
2: for $i \leftarrow 1$ to $T_{max}$ do
3:     $\gamma^i_{t+1} = \gamma^i_t - \frac{d(J(\boldsymbol{\gamma^i}))}{d(\gamma^i)}$
4: end for
5: return
```

In Table 3.2 the initial vale of γ are given in order of RBF kernel,inverse kernel and square inverse kernel respectively.

Table 3.2: Result after 100 iteration on webcam_dslr data set

d	Final MMD	initial γ	final γ	Training Accuracy	Target Acc.
1/6,2/6,3/6	2.37×10^{-4}	0.4,0.2,0.1	.64, 1.8×10^{-5} 1.33×10^{-29}	99.7%	4.43%
1/6,2/6,3/6	.0031	0.2,0.3,0.4	.6168, 8.5×10^{-5} .9045	99%	14.43%
1/3,1/3,1/3	.0025	0.2,0.3,0.4	.6998, 8.5×10^{-5} .8038	100%	12.83%
1/3,1/3,1/3	.0032	0.3,0.4,0.2	.7071, .0047 .6769	99%	14.45%

In all four cases, the source accuracies are about 100% but Table 3.2 reveals that a lower value of MMD does not necessarily yield a better accuracy on target data. This may raise a question about effectiveness of minimum MMD as a criterion for domain adaptation.

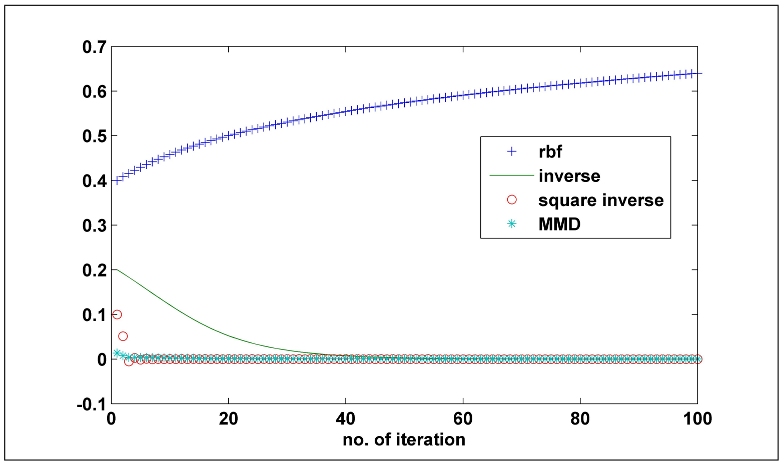

Figure 3.1: Variation of $\boldsymbol{\gamma}$ and MMD with $\boldsymbol{d}$=[1/6,2/6,3/6] and initial $\boldsymbol{\gamma}$=[0.4,0.2,0.1]

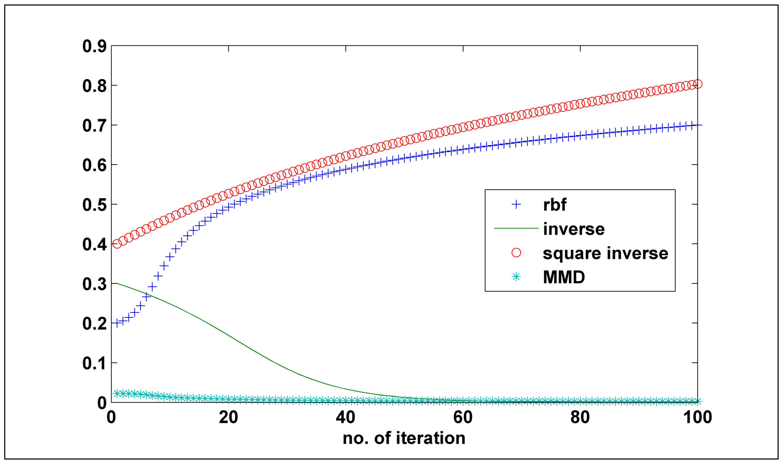

Figure 3.2: Variation of $\boldsymbol{\gamma}$ and MMD with $\boldsymbol{d}$=[1/6,2/6,3/6] and initial $\boldsymbol{\gamma}$=[0.2,0.3,0.4]

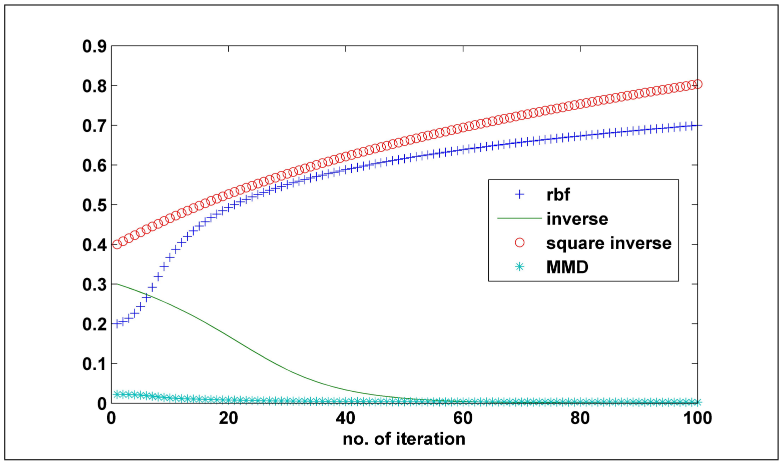

Figure 3.3: Variation of $\boldsymbol{\gamma}$ and MMD with $\boldsymbol{d}$=[1/3,1/3,1/3] and initial $\boldsymbol{\gamma}$=[0.2,0.3,0.4]

Figures 3.1 to 3.4 reveal that there are several distinct minimum of MMD which are almost equally good in terms of the MMD value. But their effectiveness in Transfer Learning is quite different.

3.5 Domain adaptation via clustering

There are a few domain adaptation methods which finds the kernel minimizing MMD. For real data sets, it may be possible that each class, both in the source and target domains might be represented by several clusters in the training data. If that happens, then instead of finding MMD between source and target data as a whole, it might be better to minimize the sum of MMDs between corresponding clusters in the source and target domains, In order to do this we need to cluster the data and establish a correspondence between clusters found in the source and target domains data. For establishing correspondence we are using the Hungarian Method [16]. The *Hungarian Method*

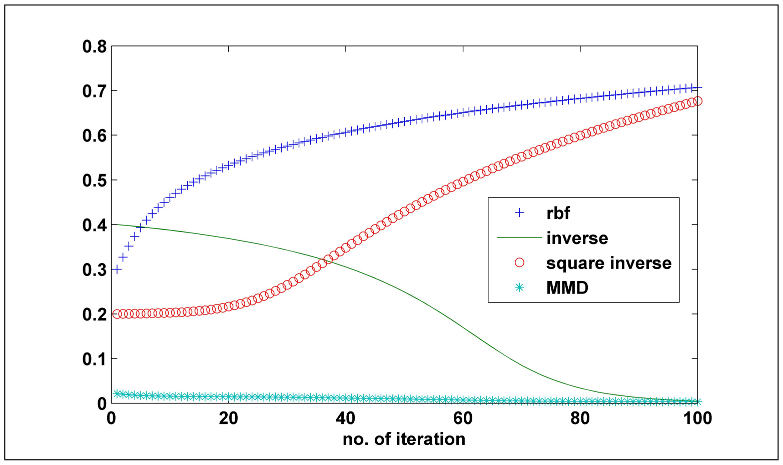

Figure 3.4: Variation of $\boldsymbol{\gamma}$ and MMD with $\boldsymbol{d}$=[1/3,1/3,1/3] and initial $\boldsymbol{\gamma}$=[0.3,0.4,0.2]

is an algorithm which can find an optimal assignment between two given sets of centroids with the given cost matrix. For this we make a $N \times N$ matrix. N is the number of clusters in the source and target data sets. The (i,j) entry of the matrix contains the distance between the i^{th} cluster in the source data and j^{th} cluster in the target data. The steps of the algorithm go as follows:

1. Subtract the smallest entry in each row from all the entries of its row.

2. Subtract the smallest entry in each column from all the entries of its column.

3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.

4. *Test for Optimality*:(1)If the minimum number of covering lines is n,an

optimal assignment of zeros is possible and we are finished.(2)If the minimum number of covering lines is less than n, an optimal assignment of zeros is not yet achieved. In that case, proceed to Step 5.

5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

Now we will get the proper assignment of source cluster to the target cluster.

3.6 Final Mathematical Formulation

We follow the same formulation as in [5], but we adapt it to clusters.
The optimization problem for Transfer Learning is formulated as

$$[k, f] = \underset{k,f}{argmin} \Omega(dist_k(D^A, D^T)) + \theta SVM_{k,f}(D) \tag{3.3}$$

where $\Omega(.)$ is monotonic increasing function and $\theta > 0$ is a trade-off parameter to balance the difference of data distribution from two domains and the structural risk functional $SVM_{k,f}(D)$ of SVM for labeled source domain patterns. We want to learn the decision function f and kernel k at the same time.

3.6.1 Multiple Kernel Learning

We are assuming the kernel function k as a linear combination of a set of a base kernels function k_m, i.e., $k = \sum_{m=1}^{M} \boldsymbol{d}_m k_m$
So our final optimization problem is as follows:

$$\underset{d\in\mathcal{M}}{min}\underset{\alpha\in\mathcal{A}}{max}\frac{1}{2}\left(\sum_{j=1}^{no.of cluster} tr(\sum_{m=1}^{M}\boldsymbol{d}_m k_m L)_j\right)^2 + \theta\left(\boldsymbol{\alpha}'\mathbf{1} - \frac{1}{2}(\boldsymbol{\alpha}\,\boldsymbol{.}\,\boldsymbol{y})'(\sum_{m=1}^{M}\boldsymbol{d}_m k_m^{L,L})(\boldsymbol{\alpha}\,\boldsymbol{.}\,\boldsymbol{y})\right) \tag{3.4}$$

where $\boldsymbol{d} = [d_1, ...d_m]$ and L represents the data with label and A=$\{\boldsymbol{\alpha} \in \mathcal{R}^n | \mathbf{c1} > 0, \boldsymbol{\alpha}'\boldsymbol{y} = 0\}$

We can write equation (3.4) as follows

$$\min\, h(\mathbf{d}) = \underset{\mathbf{d}}{min}\frac{1}{2}\mathbf{d}'\mathbf{p}\mathbf{p}'\mathbf{d} + \theta J(\mathbf{d}) \tag{3.5}$$

where $\mathbf{p}$ is a row vector of sum or max of the MMD computed over matching clusters for each kernel.
Equation(3.5) can be solved using Newton's optimization method[6]. The optimization algorithm is described as Algorithm 2.

Algorithm 2

```
Input: $(\boldsymbol{x}_1^A, y_1), (\boldsymbol{x}_2^A, y_2), ........., (\boldsymbol{x}_n^A, y_n)$, and $\boldsymbol{x}_i^A \in \boldsymbol{X}_s$
    $(\boldsymbol{x_1}, \boldsymbol{x_2}.......\boldsymbol{x_n})$ and $\boldsymbol{x}_i \in \boldsymbol{X}_T$
    M number of kernel
Output: $\boldsymbol{d}$ and classifier $f(.)$
 1: Normalize data
 2: Cluster source $(\boldsymbol{X}_s)$ and target $(\boldsymbol{X}_T)$ data separately using K-Means
    algorithm.
 3: Find the centroid of each cluster.
 4: Find the correspondence between source and target cluster using Hun-
    garian algorithm.
 5: for $i \leftarrow 1$ to M do
 6:     Find the sum of MMD over all corresponding pair of source and target
        cluster corresponding to kernel i
 7: end for
 8: initialize $\boldsymbol{d}$=$\frac{1}{M}$
 9: for $i \leftarrow 1$ to $T_{max}$ do
10:     solve for $\boldsymbol{\alpha}$ in the dual formation of SVM shown in eq(3.6) using LIB-
        SVM.
11:     update d for multiple base kernel: $\boldsymbol{d}_{t+1} = \boldsymbol{d}_t - 0.5 * \boldsymbol{g}_t$
12: end for
13: return
```

where $\boldsymbol{g} = \boldsymbol{d} + \theta(pp^T + \epsilon)^{-1}\nabla J(\boldsymbol{d})$

$\boldsymbol{p}$ =[MMD sum for kernel$_1$, MMD sum for kernel$_2$, MMD sum for kernel$_3$]

$$J(\boldsymbol{d}) = \max_{\alpha \in \mathcal{A}} (\boldsymbol{\alpha}' \mathbf{1}) - \frac{1}{2} (\boldsymbol{\alpha} \centerdot \boldsymbol{y})' (\sum_{m=1}^{M} \boldsymbol{d}_m k_m^{L,L}) (\boldsymbol{\alpha} \centerdot \boldsymbol{y}) \tag{3.6}$$

Algorithm 3

```
Input: $(\boldsymbol{x}_1^A, y_1), (\boldsymbol{x}_2^A, y_2), ........., (\boldsymbol{x}_n^A, y_n)$, and $\boldsymbol{x}_i^A \in \boldsymbol{X}_s$
    $(\boldsymbol{x_1}, \boldsymbol{x_2}.......\boldsymbol{x_n})$ and $\boldsymbol{x}_i \in \boldsymbol{X}_T$
    M number of kernel
Output: $\boldsymbol{d}$ and classifier $f(.)$
 1: Normalize data
 2: Cluster source ($\boldsymbol{X}_s$) and target ($\boldsymbol{X}_T$) data separately using K-Means
    algorithm.
 3: Find the centroid of each cluster.
 4: Find the correspondence between source and target cluster using Hun-
    garian algorithm.
 5: for $i \leftarrow 1$ to M do
 6:     Find the min of MMD over all corresponding pair of source and target
        cluster corresponding to kernel i
 7: end for
 8: initialize $\boldsymbol{d}=\frac{1}{M}$
 9: for $i \leftarrow 1$ to $T_{max}$ do
10:     solve for $\boldsymbol{\alpha}$ in the dual formation of SVM shown in eq(3.6) using LIB-
        SVM.
11:     update d for multiple base kernel: $\boldsymbol{d}_{t+1} = \boldsymbol{d}_t - 0.5 * \boldsymbol{g}_t$
12: end for
13: return
```

$\boldsymbol{p} =$[min. of MMD for kernel_1, min. of MMD for kernel_2, min. of MMD for kernel_3]
Algorithm 3 is shows an improvement over Algorithm 2, cluster pair with the minimum distance and with minimum MMD over all pairs, shows the improvement in accuracy. In this case after the correspondence between source and target clusters, we are getting the most similar cluster, the cluster pair with minimum MMD value is used in the optimization which increases the chances of better latent space since we are dealing with small MMD.

Chapter 4

Results on different Dataset

4.1 Prepossessing of Data set

We normalized every Data set by following steps

1. Subtracted mean of a feature from its corresponding feature value.
2. Divided the data point with difference of maximum and minimum of the feature

4.2 Reuter Dataset

The Reuters-21578 dataset is a standard and widely used collection of hand-labeled articles pulled from Reuters magazine. It's a very well-known benchmark data which have been extensively used in the development of algorithms for the task of text categorization. In text classification we use the Reuters-21578 data set which has been used for transfer learning setting. The basic idea is to utilize the hierarchy of the data sets. The binary classification task is defined as classifying top categories. Each top category is split into two disjoint parts with different subcategories,one for training and the other for test. In this case, distributions between the training and test data may be very different.There are three classification problems: **orgs vs people, orgs vs places and people vs places** in transfer learning setting. As we are learning multiple kernels, we are using three kernels named RBF kernel, inverse kernel and inverse square kernel. For selecting the parameters of the kernels we are doing a grid search between .1 to 1. The value for which we get the minimum MMD is taken as the parameter for the final MMD calculation. We also try to find the relationship between the MMD and the kernel parameters. We have used the gradient descent technique to optimize the MMD with respect to kernel parameters.

4.3 Office Dataset

This dataset has 31 different object categories collected under three domain settings: images from Amazon, dslr camera, and webcam. There are 4652 images in total, with the object types belonging to backpack, bike, notebook, stapler etc. The Amazon domain has, on average, 90 instances for each category, whereas dslr and webcam have roughly around 30 instances for a category. The domain shift is caused by several factors including change in resolution, pose, lighting etc. There are 800 features corresponding to each data point. Since there are 31 classes in this Dataset, we are using *One Versus One Classification.* Basically we are making one classifier for each pair of classes then we tested whole test data with these classifiers by voting.

Table 4.1: Accuracy on Reuter Datasets

Dataset	without cluster	Sum of MMD	Min. of MMD
org vs people	74.2	73.6	74.2
org vs places	68.6	63.6	66.8
people vs places	56	55.5	55.8

Table 4.2: Accuracy on Office Datasets

Dataset	without cluster	Sum of MMD	Min.of MMD
webcam vs dslr	11.62	47.29	49.3
dslr vs webcam	9.17	30.78	32.16
amazon vs webcam	6.41	14.08	15.8

Tables 4.1 and 4.2 report the results when we use our algorithm, Reuter data sets did not exhibit good cluster structure and hence there was no improvement. For the office data sets, we experimented with 4 clusters. These data sets exhibited better clusters and hence improvement in performance is quite significant.

Chapter 5

Conclusion

Many authors have used MMD in conjunction with Multiple Kernels for domain adaptation in transfer learning. Some of these studies used convex combination of multiple kernels each with a predefined value of kernel parameters, but no effort was made to optimize the MMD with respect to kernel parameters. In this thesis, first we have investigated the effectiveness of tuning kernel parameters to optimize MMD in the context of domain adaptation in transfer learning. This investigation has revealed a bit surprising results - lower MMD does not necessarily mean better performance on target data in transfer learning. However, we note that we have minimized MMD independent of learning of any associated classifier.

In the second part of the thesis, we have studied the effectiveness of clustering the source or target domain data sets before optimizing MMD. In real life scenario, even each class in the source and target data might be represented by multiple clusters. In this case, instead of minimizing MMD of the entire data set as a whole, it might be better to minimize the sum of

MMDs between pairs of corresponding clusters. Of course, if the source and target data sets do not have cluster structures, such a method is not expected to show any improvement. But if the data sets exhibit good cluster structure, we may expect significant improvement in performance. Our experimental results with two benchmark data sets, Reuters and Office, indeed, verify our expectation.

In the literature, typically authors use a large number of kernels in a multiple kernel set up. For example, in [4], authors have used a convex combination of 52 kernels. In our study, due to limited computational resources, we have used only three kernels. We believe that use of more kernels is likely to improve the performance.

Bibliography

[1] Blitzer, J., Dredze, M., Pereira, F., et al. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL* (2007), vol. 7, pp. 440–447.

[2] Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics 22*, 14 (2006), e49–e57.

[3] Daumé III, H. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* (2009).

[4] Duan, L., Tsang, I. W., and Xu, D. Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 3 (2012), 465–479.

[5] Duan, L., Tsang, I. W., Xu, D., and Maybank, S. J. Domain transfer svm for video concept detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 1375–1381.

[6] Fischer, A. A special newton-type optimization method. *Optimization 24*, 3-4 (1992), 269–284.

[7] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research 13*, Mar (2012), 723–773.

[8] Huang, J., Gretton, A., Borgwardt, K. M., Schölkopf, B., and Smola, A. J. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems* (2006), pp. 601–608.

[9] Jiang, W., Zavesky, E., Chang, S.-F., and Loui, A. Cross-domain learning methods for high-level visual concept classification. In *2008 15th IEEE International Conference on Image Processing* (2008), IEEE, pp. 161–164.

[10] Mathew, K., Tan, C. E., and Issac, B. Evaluation of sound perception to identify candidate frequency for wireless networking. In *New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering*. Springer, 2015, pp. 349–359.

[11] MTL, M. L. Multi-task learning.

[12] Pan, S. J., and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering 22*, 10 (2010), 1345–1359.

[13] Roscoe, S. N. Incremental transfer effectiveness. *Human Factors: The Journal of the Human Factors and Ergonomics Society 13*, 6 (1971), 561–567.

[14] Thrun, S., and Pratt, L. *Learning to learn.* Springer Science & Business Media, 2012.

[15] Uguroglu, S., and Carbonell, J. Feature selection for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2011), Springer, pp. 430–442.

[16] University, H. The Assignment problem and Hungarian Algorithm. `http://www.math.harvard.edu/archive/20_spring_05/handouts/assignment_overheads.pdf`.

[17] Yang, J., Yan, R., and Hauptmann, A. G. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia* (2007), ACM, pp. 188–197.