A FRAMEWORK FOR TRANSFER LEARNING: MAXIMIZATION OF QUADRATIC MUTUAL INFORMATION TO CREATE DISCRIMINATIVE SUBSPACES

By

MOHAMMAD NAZMUL ALAM KHAN

Bachelor of Science in Computer Science & Engineering
Bangladesh University of Engineering & Technology
Dhaka, Bangladesh
2007

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2016

COPYRIGHT ©

By

MOHAMMAD NAZMUL ALAM KHAN

July, 2016

A FRAMEWORK FOR TRANSFER LEARNING: MAXIMIZATION OF QUADRATIC MUTUAL INFORMATION TO CREATE DISCRIMINATIVE SUBSPACES

Dissertation Approved:

Dr. Douglas Heisterkamp

Dissertation Advisor

Dr. Blayne Mayfield

Dr. Nohpill Park

Dr. Guoliang Fan

ACKNOWLEDGMENTS

The completion of PhD is a great achievement and a milestone in my life. When I first left my country and joined Computer Science department of OSU, it was a whole new experience and change of my career. Now that I am at the verge of a successful competition of my degree, firstly I will be ever grateful and thankful to the Almighty who blessed me with enthusiasm, perseverance, courage and all necessary calibers for which I have been able to complete this long journey of almost seven years. Secondly, my final dissertation topic is fully supervised by the chair of my PhD committee and my research advisor Dr. Douglas Heisterkamp. I feel lucky and honored to have the opportunity to work under his supervision. It was him who guided me to the proper direction and helped me stick to the desired goal. He is an open-minded, visionary, extra-ordinary mentor and sometimes I feel that I might not be quite able to fulfill his level of vision. I would like to express my heartiest gratitude and thanks to him. Also a major part of my PhD study was guided by Dr. Guoliang Fan, a person to whom I will be ever grateful for teaching me how to do research. He is a very caring mentor and supported me a lot, specially when I was about to give up. Last but not the least, I was very much thankful to my department (Computer Science) for providing me with continuous assistantship through out my study. Now I will take the opportunity to thank my family: my wife (Zahra Maria), my parents and also my in-laws. I admit this accomplishment was not very smooth, but I did not lose hope and it was because of my wife who always tries to motivate me to reach my goal. It is not possible to acknowledge enough to her in this limited space. I am also very much delighted that I did not fail my family members, specially my parents. I feel accomplished to fulfill

their dream. I was not very confident to leave them and start a whole new life long away from my home country. They inspired me, always kept me in their prayers and always beside me, whenever I faced any difficulty in my life. Lastly, I will always remember the support and advice from my in-laws and be grateful to them. I would also like to thank whole heartedly to all my friends, relatives and well-wishers who was beside me directly or indirectly during this toughest period of my life. I apologize not to mention everyone's name here because of space limitation.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: Mohammad Nazmul Alam Khan

Date of Degree: July, 2016

Title of Study: **A FRAMEWORK FOR TRANSFER LEARNING: MAXIMIZATION OF QUADRATIC MUTUAL INFORMATION TO CREATE DISCRIMINATIVE SUBSPACES**

Major Field: Computer Science

In the area of pattern recognition and computer vision, *Transfer learning* has become an emerging topic in recent years. It is motivated by the mechanism of human vision system that is capable of accumulating previous knowledge or experience to unveil a novel domain. Learning an effective model to solve a classification or recognition task in a new domain (dataset) requires sufficient data with ground truth information. Visual data are being generated in an enormous amount every moment with the advance of photo capturing devices. Most of these data remain unannotated. Manually collecting and annotating training data by human intervention is expensive and hence the learned model may suffer from performance bottleneck because of poor generalization and label scarcity. Also an existing trained model may become outdated if the distribution of training data differs from the distribution where the model is tested. Traditional machine learning methods generally assume that training and test data are sampled from the same distribution. This assumption is often challenged in real life scenario. Therefore, adapting an existing model or utilizing the knowledge of a label-rich domain becomes inevitable to overcome the issue of continuous evolving data distribution and the lack of label information in a novel domain. In other words, a knowledge transfer process is developed with a goal to minimize the distribution divergence between domains such that a classifier trained using source dataset can also generalize over target domain. In this thesis, we propose a novel framework for transfer learning by creating a common subspace based on maximization of non-parametric quadratic mutual information (QMI) between data and corresponding class labels. We extend the prior work of QMI in the context of knowledge transfer by introducing soft class assignment and instance weighting for data across domains. The proposed approach learns a class discriminative subspace by leveraging soft-labeling. Also by employing a suitable weighting scheme, the method identifies samples with underlying shared similarity across domains in order to maximize their impact on subspace learning. Variants of the proposed framework, parameter sensitivity, extensive experiments using benchmark datasets and also performance comparison with recent competitive methods are provided to prove the efficacy of our novel framework.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

Introduction

In computer vision, object class detection or classification has been studied in both supervised and unsupervised settings. Traditional machine learning techniques try to learn a model based on a training data set and it is assumed that i) training and test data follow the same distribution, and ii) they are in the same feature space [4, 5]. These assumptions are often challenged in real life scenario. One major prerequisite of building most of these models is the availability of abundant labeled training images which might create a bottleneck, as it involves manual labor to annotate data. Also distribution of data changes over time. It is quite usual if the dataset on which a model is trained vary significantly from the data distribution during testing time [6–9]. This may cause poor model performance in terms of classification accuracy on the test set. Nevertheless, with the advance of image capturing and sensory devices e.g. DSLR (Digital Single-Lens Reflex) camera, webcam, mobile camera etc., enormous amount of image data are being generated at every moment. Therefore, the need for transfer learning may arise when the data can be easily outdated. In this case, the labeled data obtained in one time period may not follow the same distribution in a later time period. Hence, one of the challenges for building a model is to cope up with the emerging and ever-changing nature of dataset. Moreover, it is also assumed in traditional machine learning approach that the availability of labeled training examples in a dataset will be sufficient to build a model which will not face any novel category/instance or any variation or mutation of data during future testing. As for example, a model which is trained using indoor images captured

from canonical viewpoints of objects or categories, might perform poorly in outdoor settings. An example setting is provided in Figure 1.1. Therefore, a learning technique should be able to adapt to the changes across data distributions without requiring to train the model from scratch every time a new challenge appears. An adaptation method is required to reuse or transfer previously achieved knowledge for dealing with unknown samples or variation in test data distribution. In general, this research area is widely known as "Transfer Learning".

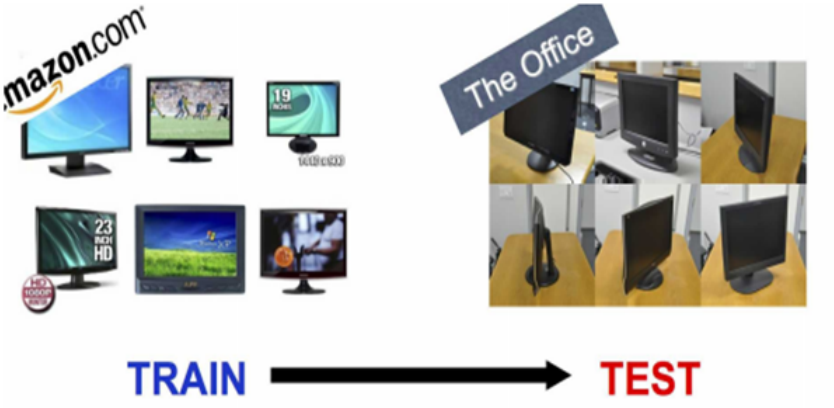

Figure 1.1: An example of transfer learning problem [1].

In our research, we will investigate some issues of transfer learning problem and propose an efficient framework to deal with them. In this article, we will refer *domain* and *dataset* interchangeably. It is worth mentioning that this area is also known as *domain adaption* in literature [1, 10]. In a typical setting, there is a *source* domain where abundant labeled data are available or a trained model with available training samples are available for a specific task like classification, regression etc. There is another domain referred to as *target* where any combination of the following three cases may arise,

1. labeled data are in short supply.

2. the calibration effort is very expensive.

3. the learning process is time consuming and costly.

In this situation, sharing or transferring previous knowledge/experience from the source domain might be helpful to build a model in the target domain. The goal is to overcome the difficulties of the target domain to accomplish the task of interest. We already mentioned why these three situations may occur. This is a very interesting and challenging topic in current machine learning and vision based research which has direct impact on real life scenario. Therefore, I have decided to explore some of the issues in this area that can be dealt from different perspective or might need more attention for improvement. This type of topic is also exercised in Natural Language Processing (NLP), Sentiment classification and many other areas. It is assumed that although the data distributions in different domains is different, they share some common characteristics in underlying low-dimensional manifold. The main challenge is to explore this underlying common structure or manifold across domains. Our research deals with the transfer learning problem and we propose an iterative framework to propagate knowledge from source to target domain in the form of class identity information. The idea is to learn a common feature subspace such that data sharing underlying similarity (across domains) can be projected in close proximity on the learned subspace. This way, the divergence between source and target data can be minimized and then the labeled source data can be utilized to predict unlabeled target domain data.

There are three main questions that need to be answered in the transfer learning area [1],

1. **What to transfer?** Before using previous experience or knowledge, we should be aware of what information we should utilize to learn a model that is suffering from fewer training examples. This knowledge may come in various forms depending on the nature of the target task. Not all the information from an

available source domain might be effective in knowledge adaptation and this may affect the system performance. Identifying the source information as *useful* or *non-useful* might be a challenging task. In this research, we investigated this issue and provide a very simple but elegant approach to deal with this.

2. **How to transfer?** The next issue is how to efficiently apply the transferred knowledge in order to facilitate model building in the target domain. This question is about the methodology that can connect the source knowledge towards target domain so as to efficiently explore the unknown or unlabeled target domain data.

3. **When to transfer?** This issue deals with the appropriate choice of the source knowledge that has been selected as a candidate for transfer. As for example, a model built on indoor home object images or office supply images is not a suitable candidate for learning a model that will identify school or campus decor objects, as these two domains do not share any features or characteristics among them. It is already mentioned that the domain difference might occur due to variability in capturing devices, images captured in different time domain, different resolutions, different surrounding environments and also different canonical viewpoints. Whether the transferred knowledge is worthwhile in the target domain is another prime issue to address. In some cases, transferring knowledge might cause performance bottleneck if we fail to apply it appropriately and when needed.

In our research, we propose a framework that will deal with these three issues inherently. The motivation was to utilize the underlying common shared structure to learn a low-dimensional subspace and propagate the source information towards unlabeled target data gradually. Our formulation is a 'knowledge propagation'approach where 'knowledge'refers to discriminative identity information of the available source model.

1.1 Terminology

In this section, some definitions and notifications are introduced that will be used through out this article. This section is highly influenced by the work of Pan and Yang [1].

Domain: A domain ($\mathcal{D}$) or dataset is referred to as a collection of visual data. Each image is represented in a high-dimensional feature space $\mathcal{X}$ as a data point and a marginal distribution of data points is associated with a domain. In this research area, it is generally assumed that one domain follows a single marginal distribution. Thus a domain can be defined as $\mathcal{D} = \{\mathcal{X}, P(\boldsymbol{x})\}$. A domain might be referred to as *source* or *target* depending on the direction of knowledge transfer. Usually source domain is rich with label information i.e. each image is correctly annotated, whereas target domain suffers the availability of annotations and hence training a classification or regression model using only target domain data becomes impossible. Therefore, source domain data along with label information are utilized to explore the novel target domain in a transfer learning based approach.

Task: Given a specific domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task is referred to as learning a model $f(.)$ (classification, regression etc.) to annotate unknown data in a label space $\mathcal{Y}$. Therefore, a task $\mathcal{T}$ is defined as a tuple $\mathcal{T} = \{\mathcal{Y}, f(.)\}$. $f(.)$ is generally an objective predictive function that is leaned using the domain data $\boldsymbol{x} \in \mathcal{X}$ and corresponding label information $y \in \mathcal{Y}$. For a single domain, data are sampled as training and testing set where training set contains data from different object categories. The model is tested using the test data. Another approach is to partition the whole data set into three subsets: training, validation and testing set, where validation set is used to fine-tune the model parameter. This setting is used for various tasks, for example, object category detection, scene recognition, padestrian detection,

face recognition [11, 12] etc. It is worth noting that both training and testing data are sampled from the same marginal distribution, this setting might be challenged in real life scenario, e.g. if testing data come from a different distribution. Although the set of class labels is same across training and testing distributions, the marginal data distributions are different. This entails a bottleneck for a learned model to correctly identify the unknown test data.

In this research, we focus on this domain difference problem. It has been proven experimentally that trained model performs poorly if the testing data are sampled from a different distribution. Therefore, it will be impractical to deploy a trained model without considering variability of unknown data. This necessitates model adaptation such that divergence difference is minimized. There are mainly two approaches for this adaptation:

1. Adapting a trained model by developing an *adaptation* layer on top of the model which will minimize the dataset bias. One popular approach is to utilize a Convolutional Neural Network model [13]. Usually a CNN is trained to learn a hierarchical features for images where intermediate layers represent mid-level image features. These mid-level features posses the characteristics of shared common structure. Therefore these features can be utilized to minimize the divergence gap between two different distributions.

2. Utilizing a label-rich large dataset for transferring knowledge i.e. if the labeled source and unlabeled target domain data are available, then they can be projected in a common low-dimensional manifold subspace with a goal to utilizing their underlying common structure. On that projected space, the labeled source domain data can be used to predict the class identity of the unknown target domain data.

1.1.1 Definition: Transfer Learning

. Given a source domain $\mathcal{D}_s$ and learning task $\mathcal{T}_s$, a target domain $\mathcal{D}_t$ and learning task $\mathcal{T}_t$, transfer learning aims to help improve the learning of the target predictive function $f_t(.)$ in $\mathcal{D}_t$ using the knowledge of $\mathcal{T}_s$ in $\mathcal{D}_s$, where $\mathcal{D}_s \neq \mathcal{D}_t$ or $\mathcal{T}_s \neq \mathcal{T}_t$ [1].

According to the definition, as domain is defined as a pair $\mathcal{D} = \{\mathcal{X}, P(X)\}$, the condition $\mathcal{D}_s \neq \mathcal{D}_t$ implies that either $\mathcal{X}_s \neq \mathcal{X}_t$ or $P_s(X) \neq P_t(X)$. For example, in a task of document classification, there are both source document set and a target document set and either the term features are different between the two sets (e.g., they use different languages), or their marginal distributions are different. Similarly, a task is defined as a pair $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$. Thus, the condition $\mathcal{T}_s \neq \mathcal{T}_t$ implies that either $\mathcal{Y}_s \neq \mathcal{Y}_t$ or $P(Y_s|X_s) \neq P(Y_t|X_t)$ [1]. When the target and source domains are same, i.e. $\mathcal{D}_s = \mathcal{D}_t$ and their learning tasks are same, i.e., $\mathcal{T}_s = \mathcal{T}_t$, the learning problem becomes a traditional machine learning problem.

Given specific domains $\mathcal{D}_s$ and $\mathcal{D}_t$, when the tasks $\mathcal{T}_s$ and $\mathcal{T}_t$ are different, then either the label spaces between domains are different i.e. $\mathcal{Y}_s \neq \mathcal{Y}_t$ or the conditional probability distributions of theses domains are different i.e. $P(Y_s|X_s) \neq P(Y_t|X_t)$. In the document classification example, the former case corresponds to the situation where source domain has binary document classes whereas the target domain has 10 classes to classify the documents to. The latter case corresponds to the imbalanced situation in class distribution in source and target domains. In addition, the two domains are considered to be related if there exists some underlying shared structural similarity among the two domains.

1.2 Cases of Transfer Learning

Let source and target domains are represented by $\mathcal{D}_s$ and $\mathcal{D}_t$ respectively. In the same way, source and target domain tasks are represented as $\mathcal{T}_s$ and $\mathcal{T}_t$ respectively, their corresponding predictive functions are represented as $f_s(.)$ and $f_t(.)$ respectively and also their corresponding label spaces as $\mathcal{Y}_s$ and $\mathcal{Y}_t$ respectively. According to the definition of *domain*, the domain difference or distribution shift occurs (i.e $\mathcal{D}_s \neq \mathcal{D}_t$) when either $\boldsymbol{X}_s \neq \mathcal{X}_t$ or $P(\boldsymbol{x}_s) \neq P(\boldsymbol{x}_t)$. Therefore, transfer learning deals with the scenario where $\mathcal{D}_s \neq \mathcal{D}_t$ and tries to learn a predictive function $f_t(.)$ by using the knowledge in $\mathcal{D}_s$ and the source predictive function $f_s(.)$.

Based on the definition of transfer learning, the following four cases are the possible scenarios where the transfer of previous knowledge is necessary,

$\mathcal{X}_s \neq \mathcal{X}_t$ In this case, feature encodings across domains are different i.e. the feature space where a model is trained is different from the feature space of the novel target domain. This difference might be the result of mismatch in dimensionality across domains or using different image representations. As for example, the same image or object can be represented as multiple feature representations, this line of research is also known as Heterogeneous Feature Adaptation (HFA) [14]. Usually a set of pivot features are detected across feature spaces to connect the two different feature distribution into a common shared feature set.

$P(\boldsymbol{x}_s) \neq P(\boldsymbol{x}_t)$ In this case, the marginal distributions of two different domains (i.e. training and testing) are different. This scenario is commonly known as *domain adaptation*, *distribution shift* or *dataset bias* [10, 15, 15, 16]. Our research is focused on this issue with a goal to minimize the shift between this two marginal distributions. Here images from source and target domains are represented with the same feature encoding and also the label space are

same across domains. The goal is to identify the target domain data with a classification model that is mainly trained with the source domain data. This also minimizes the necessity of label information in target domain and adapts an already existing model to cope up with continuous evolving data set.

$\mathcal{Y}_s \neq \mathcal{Y}_t$ This scenario is possible when the class data distribution is imbalanced. In other words, if the source domain consists of p number of classes and the target domain consists of q number of classes, where $p \neq q$. Here the label space is different across domains. In the literature, this scenario is also known as *zero-shot learning* [17] or *one-shot learning* [18]. In zero-shot learning, no training data is available for the novel category in the target domain, whereas in one-shot learning, only one labeled training datum is available during training phase. Usually auxiliary information is harnessed to extract the underlying features existing in the novel category. This setting will be considered as a future extension of our current research so that the current proposed algorithm can handle the scenario of facing novel category that was not present during the training phase.

1.3 Settings of transfer learning

Based on the information available in source and target domain, several settings of transfer learning is possible, among them three common settings are: *inductive* transfer learning, *transductive* transfer learning and *unsupervised* transfer learning. Table 1.1 summarizes these three different scenarios. This table has been quoted from [1]. Here is a brief description of each of these settings.

Inductive transfer learning: In this setting, $\mathcal{T}_s \neq \mathcal{T}_t$ i.e. the predictive target function $f_t(.)$ is different from the one in the source domain. Therefore, source model can't be directly utilized for the target domain task, rather the infor-

Table 1.1: Different settings of Transfer Learning. The table is adopted from [1]

Transfer Learning Setting	**Related Areas**	**Source Domain Labels**	**Target Domain Labels**
Inductive TL	Multi-task Learning	Available	Available
	Self-taught Learning	Unavailable	Available
Transductive TL	Domain Adaptation, Sample Selection Bias, Co-variate Shift	Available	Unavailable
Unsupervised TL		Unavailable	Unavailable

mation used in training a source model can be used as an auxiliary data for the target model. Although the target domain might not suffer from labeled data, researchers have been utilizing related auxiliary data from a different domain to boost up the robustness or performance of the model. One example of this setting is to use semantic feature space for the corresponding object or image dataset and build a connection between image feature space and semantic feature space to enhance the quality of the trained model [19]. Based on the availability of the labled data in the source domain, this inductive setting can be further categorized into following two settings:

a. Labeled data in the source domain are available in plenty of amount. This scenario is often referred to as *multi-task* learning [20]. A significant difference between multi-task learning and inductive transfer learning is that multi-task learning tries to optimize the function for both source and target task whereas inductive learning tries to learn a predictive function using target domain data along with utilizing the information available from the source domain data.

b. No labeled data are available in the source domain, this scenario is widely known as *self-taught* learning [21]. Raina *et al.* first proposed a framework to utilize the knowledge of unlabeled source data that come from the same distribution as the labeled target domain data. Also the label distribu-

tion $P(y)$ might be different across domains. This implies that unlabeled data might not be directly employed to learn a predictive model for the target domain, rather this information might act as a connecting auxiliary knowledge to help learn the target model.

Transductive transfer learning: In this setting, $\mathcal{T}_s = \mathcal{T}_t$ but $\mathcal{D}_s \neq \mathcal{D}_t$ i.e. the tasks across domains are similar whereas the source and target domains are different either in terms of feature space or marginal probability distribution of data. Usually, target domain suffers from lack of labeled data causing difficulty to build a classification or regression model. This requires exploiting related source domain where plenty of labeled data are available. The term 'transfer learning'is used for this setting in general. Based on the type of distribution of source and target domains, this setting is further categorized into following two sub-settings:

a. Here domain difference implies $P(\boldsymbol{x}_s) \neq P(\boldsymbol{x}_t)$ although the images are represented in the same feature space i.e. $\mathcal{X}_s = \mathcal{X}_t$. Also the two domains have the same label space. This area is widely known as *domain adaptation*. In this research, we will mainly focus on this issue. Two different settings in domain adaptation (DA) are usually considered: i)unsupervised domain adaptation [7, 8, 22, 23], where no labeled data available in target domain; and ii) semi-supervised domain adaptation [24, 25], where only a few labeled data are available in target domain along with abundant labeled data of source domain. Some other related topics in the literature that deal with domain adaptation include domain adaptation for knowledge transfer in text classification [26] , sample selection bias [27] or covariate shift [28].

b. This scenario happens when the target domain includes a novel category

data which was not present in the source domain during training phase. This implies that label space is different across domains i.e. $\mathcal{Y}_s \neq \mathcal{Y}_t$. The goal is to identify the features that have similar characteristics distribution among available source categories and the novel ones. Often a set of auxiliary information from web search is used in training to enhance the feature space for fitting novel category in the target domain [17, 29, 30].

Unsupervised transfer learning: This setting is similar to inductive transfer learning except that this is an unsupervised learning method i.e. the task in target domain includes clustering, dimensionality reduction, density estimation etc [31,32]. In this case, no labeled data are needed in the target domain, hence source data are used in an unsupervised way to help train a model dedicated for the target task.

Our research will be focused on second setting of *transductive* case where $P(X_s) \neq P(X_t)$ and $P(Y_s|X_s) \neq P(Y_t|X_t)$. The label space will be same across domains i.e. $\mathcal{Y}_s = \mathcal{Y}_t$. Nevertheless, in our research, we consider the tasks across domains be same i.e. mainly focusing on classification or object recognition task across domains.

1.4 History of transfer learning

Broadly, three different types of approaches are found in the literature to deal with the above three cases of transfer learning settings. As our focus is on the trunsductive scenario, we performed our literature review based on that case only. Researchers have proposed a vast number of approaches to deal with the transfer learning issue, they can be roughly categorized as the following three approaches,

1. Instance based transfer.

2. Feature representation (subspace learning) based transfer.

3. Model parameter transfer.

The first context can be referred to as *instance-based transfer learning* (or instance transfer/ instance reweighting) approach [33–37], which assumes that a certain portion of the source domain can be potential candidate for knowledge transfer, hence source domain data are weighted based on their relevance with target domain data. In this scheme, the impact of non-relevant data are minimized by down-weighting them as they share the least common characteristics with target data. A second case can be referred to as *feature-representation transfer learning* approach [38, 39]. This can be also referred to as *subspace learning* based approach. The idea here is to learn a common 'good'feature representation that can be used to encode both source and target domain image. Therefore the knowledge that will be transferred from source domain will be encoded in the feature encoding. This is intuitive as although the marginal distributions are different across domains, their underlying structure should be similar in a low-dimensional manifold. A third case can be referred to as *parameter-transfer* approach [13, 40] which reuse the parameter space learned using a label-rich source domain data. In this case the knowledge is transferred in the parameter space. A fine-tuning step might be necessary to align the parameter set to work for the target domain task, therefore available labeled data in the target domain is used to fine-tune this ''adaptation"layer.

1.4.1 Instance based transfer

This approach is based on correspondence relationship across domains using some pivot instances which are used to build a common shared model. Therefore, instance relationship should be known as prior information. In [19], a instance similarity has been applied along with some external semantic knowledge to propagate the source information for learning a target predictive function. Hoffman *et al.* propose a class invariant transformation function that projects the data points from one subspace to

another (source or target) with the help of instant constraints. This method uses both low-dimensional manifold structure and instance constraints to learn a feature space. A different approach is applied by Lim who tries to augment the source training data by for each class by borrowing example samples from other classes and learns a transformation function with the augmented set [41]. In [42, 43], authors propose to learn a similarity function using both source and target domain data and source label information both in linear and kernel form. This similarity function is plugged in various established classification models such that labeled source data are used for classifier training and unlabeled data are used for testing. Similar type of approach has been proposed by Donahue *et al.* [24] who tried to learn a smoothness regularizer to plug into an existing classification model. This smoothing function is learned with the help of instance relationship across domains. A geometric relationship among instances in the target domain is utilized by [44] which is known as low-rank reconstruction. The idea is that a point with same neighborhood class points in target domain can be reconstructed by the same neighboring class points in the source domain.

1.4.2 Feature representation based transfer/ subspace learning

In this category, a common image representation is sought in order to minimize the distribution divergence across domains and hence source data can be used to predict the identity of unknown target domain data. One of the case is, if data in different domains are encoded with different image representations or they have different dimensionality across domains. This scenario is known as Heterogeneous Domain Adaptation (HDA) [45]. Another approach is based on Heterogeneous Feature Augmentation (HFA) [2]. Here source and target domain data are represented with different feature spaces and hence two different projection matrices P and Q are learned for projecting them into a common feature space. This augmented feature space takes into account the original feature set and learned feature set with a goal to

build a large feature vector that will incorporate both common and individual characteristics among source and target domains (see Figure 1.2). A Feature Replication (FR) based approach has been applied in [39] i.e. each data $\boldsymbol{x} \in \mathbb{R}^d$ is augmented by extending its dimension upto three times $\boldsymbol{x} \in \mathbb{R}^{3d}$. Therefore two different feature transformation function are learned to map the data as following, $\phi_s(x) = [x, x, 0]^T$ and $\phi_s(x) = [x, 0, x]^T$ where $\phi_s(.)$ and $\phi_t(.)$ are learned for source and target domain data respectively. This is also a feature augmentation based approach similar to [2] except that no projection or mapping was applied in this augmentation, only original raw feature space is used to augment the feature space. Although seems surprising, they provided promising results to support their framework.

Hoffman *et al.* proposes a domain-invariant feature representation with a goal to minimize the effect of distribution shift in the feature space [46]. As a part of the classification training process, they learn a feature mapping function that aligns the target domain features towards source domain. They propose a generic framework by optimizing feature space and classification jointly. Their method also supports large scale problem, heterogeneous domain adaptation and multi-class representation learning. Fei-fei *et al.* reused an old model built on unrelated categories to minimize the necessity of labeled images for the novel categories during training phase [47].The authors employed a Bayesian probabilistic framework to model the object categories along with prior information available in the source domain. Another line of work based on discriminative learning is widely used in NLP area [38]. A structural learning framework based on feature correspondence is proposed to establish a shared classification model across domains. Another issue of domain adaptation is the continuous evolving of data through out the time. Hoffman *et al.* deals with this issue where target data are considered to be samples from different subspaces and propose a novel framework for continuous manifold adaptation.

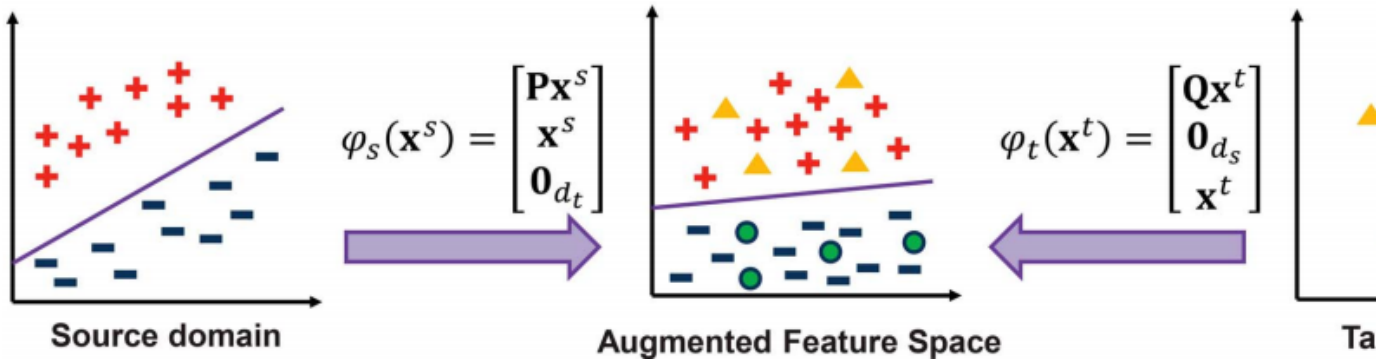

Figure 1.2: Samples from different domains are represented by different features, where red crosses, blue strips, orange triangles and green circles denote source positive samples, source negative samples, target positive samples and target negative samples, respectively. By using two projection matrices $\mathbf{P}$ and $\mathbf{Q}$, we transform the heterogenous samples from two domains into an augmented feature space. This picture is taken from [2].

Recently subspace learning based methods have gain popularity because of their simplicity and effectiveness in learning process [7, 22, 48]. In these cases, authors utilize both source and target training data to learn a common low-dimensional subspace with a goal to bring the two different distributions close together. Our work is somewhat related to this category, we propose a framework of subspace learning based on information theoretic perspective. Unlike others' approach, we not only used the marginal distribution of source and target data, but also take the advantage of class conditional distribution of source domain data to build a discriminative common subspace.

1.4.3 Parameter transfer approach

In this approach, two major lines of work are widely practiced: one is to reuse the parameter set of an existing trained model as a initializer of the training process for the target domain task model, another is to employ the model parameters as a regularization function in the objective function of target task model. Usually a model built using a large scale database is a suitable candidate for transfer learning using parameter transfer approach. Oquab *et al.* showed how a convolutional neural net-

work model trained with large image database can be efficiently applied to different tasks or different domains where labeled data is not sufficient to build a classification model. Mid-level image representations have been proved effective in transfer learning scenario as they possess the common shared features across domains, therefore, the authors propose to train an adaptation layer by reusing the previously learned layer. This can alleviate the necessity of labeled target data even if the source task is completely different from the target domain task. Tommasi *et al.* proposed a model adaptation framework based on both parameter transfer and instance weighting approach. Their work focuses on extending an SVM based model along with weighting previous knowledge in order to align the source domain with the target one. The work of Ayter *et al.* is also based on parameter transfer approach where they propose to transfer the training of a detector model to identify a novel category in the target domain [40]. The previously learned object template has been adapted to act as a regularizer for the training of novel categories. Some other notable research in this category include hierarchical classification model to incorporate object hierarchy structure [49], one shot learning utilizing class relevance metrics [50], adapting a Naive Bayes Nearest Neighbor classifier [51] etc.

1.5 Comparison methods

In next chapters, we propose our domain adaptation framework with exhaustive experimental evaluations. We have used several benchmark datasets to prove the efficacy of the proposed model. We show that the proposed framework surpasses state-of-the-art approaches in most of the cases by a significant margin. We also conduct detailed experiments by varying parameter setting and analyzed the effect of them on the proposed method. Here we will provide a brief description of each of the method that are used for comparison, thanks to the authors of these methods to publish their codes online. Same experimental protocol has been followed to test the performance

by each method along with ours for a fair comparison. According to DA setting, labeled source data and unlabeled target data are applied to learn a common subspace. Finally, the projected source data are utilized to annotate the unlabeled target domain data. In the following, the methods to be compared are described briefly.

1. Geodesic flow kernel [22]: This approach is based on finding a low-dimensional feature representation. The authors propose a kernel based method that model the underlying low-dimensional structure along the geodsic path from source to target domain. The model integrates an infinite number of subspaces which characterizes the domain shift in geometrical and structural properties along the path from source to target.

2. Subspace alignment [7]: In this method, source and target domain is represented by their corresponding eigen spaces. An optimization function is designed to align the source domain with the target one. This mapping function projects cross-domain data into a low-dimensional subspace such that a classifier trained using projected source data can be efficiently applied to projected unlabeled target data.

3. Transfer feature learning [23]: The authors propose a dimensionality reduction framework with a goal to minimize the distribution divergence between source and target domain data. Their method, known as Joint Distribution Adaptation, jointly optimizes both the marginal and conditional distribution between domains. Their optimization criterion is the non-parametric Maximum Mean Discrepency (MMD) that measures the difference between the sample means of source and target data. The MMD criterion is integrated with PCA formulation to learn a low-dimensional subspace.

Transfer jont matching [8]: The authors propose a cross-domain feature representation which is invariant to both the distributions across domains and also the

irrelevant instances. They employ a feature matching and instance reweighting based technique to identify closely distributed source samples with target ones. The idea is to assign higher weights to source samples that are more relevant to the target domain data in order to minimize the impact of irrelevant data existing in the source domain.

Transfer component analysis [9]: Here the authors also use MMD metric to learn transfer components in a reproducing kernel Hilbert space. These transfer components will act as connecting landmarks between source and target domains such that in the projected subspace the difference in domain distributions is minimized. The authors prove that this new feature representation is suitable to use in traditional machine learning methods in order to classify unknown target domain data.

Principal component analysis [52]: This method is not intended for domain adaptation but has been used as a benchmark method for our proposed algorithm. PCA is an unsupervised dimensionality reduction technique with a goal to preserve maximum data variance in low-dimensions. We will project both source and target data into a PCA subspace and evaluate the classification accuracy achieved by a classifier trained using projected source data. The performance evaluation will serve as a benchmark criterion for the need of domain adaptation or transfer learning.

A list of notations/symbols used in this thesis is provided in the Appendix section.

CHAPTER 2

Subspace Learning Based On Quadratic Mutual Information Induced With Soft-labeling

In this chapter, a domain adaptation framework based on maximization of mutual information induced with soft-labeling has been proposed. In this problem setting, a source domain dataset with corresponding class labels and an unlabeled target domain dataset are available. The goal is to learn a low-dimensional common feature subspace using both source and target domain data so that a classifier trained using projected source data can be applied to predict class labels for the unlabeled target domain data.

It is assumed that source and target data of a same class will be closely located in an ideal common subspace. A supervised technique based on maximization of non-parametric mutual information (MI) between data and corresponding class labels has been proved effective in learning such discriminative subspace [53–56]. Now to extend the prior work in domain adaptation setting, we induced soft assignment of class labels (probability that a point belongs to a class) into the objective function [53] of MI maximization. In the learned subspace, target data will be labeled *softly* using neighboring labeled source data. These two steps: i) finding MI-maximized subspace and ii) updating target data predictions will continue in an iterative fashion till converging to a final feature space. This iterative approach will aid to learn a domain-adaptive discriminative subspace utilizing both source and target domain data. In summary, our contributions in this work will be,

- utilizing class label distribution of source data along with all data across domains to develop a domain adaptation framework.
- extending supervised method of MI to support unlabeled target data by inducing soft-labeling and
- proposing an iterative approach of common subspace learning based on maximization of non-parametric MI with soft-labeling.

2.1 Mutual Information

According to information theoretic literature, *Mutual Information* (MI) is defined as a measures of independence between random variables [56, 57]. Assume a random variable X representing d-dimensional data points $\boldsymbol{x} \in \mathbb{R}^d$ and another discrete random variable C representing class labels from $c \in \{1, 2, \ldots, N_c\}$, where N_c is the total number of classes. Also let $p(\boldsymbol{x})$ is the marginal probability density function for the data samples, $P(c)$ is the class prior probability and $p(c)$ marginal distribution of class labels. Then MI is defined as,

$$\mathcal{I}(X, C) = H(C) - H(C|X). \tag{2.1}$$

where $H(C)$ denotes the Shanon's entropy or uncertainty of class labels [57] which is defined as,

$$H(C) = -\sum_{c} P(c) \log(P(c)). \tag{2.2}$$

and $H(C|X)$ is the class conditional entropy which is defined as,

$$H(C|X) = -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \left(\sum_{c} p(c|\boldsymbol{x}) \log(p(c|\boldsymbol{x})) \right) d\boldsymbol{x} \tag{2.3}$$

Therefore, $\mathcal{I}(X, C)$ can be written as,

$$\mathcal{I}(X, C) = -\sum_{c} P(c) \log P(c) + \int_{\boldsymbol{x}} p(\boldsymbol{x}) \left(\sum_{c} p(c|\boldsymbol{x}) \log p(c|\boldsymbol{x}) \right) \tag{2.4}$$

Using the identities, $p(c, \boldsymbol{x}) = p(c|\boldsymbol{x})p(\boldsymbol{x})$ and $P(c) = \int_{\boldsymbol{x}} p(c, \boldsymbol{x})d\boldsymbol{x}$, we can further simplify the above equation,

$$\begin{aligned}
\mathcal{I}(X, C) &= -\sum_c \int_{\boldsymbol{x}} p(c, \boldsymbol{x}) \log P(c) + \sum_c \int_{\boldsymbol{x}} p(c, \boldsymbol{x}) \log p(c|\boldsymbol{x}) \\
&= -\sum_c \int_{\boldsymbol{x}} p(c, \boldsymbol{x}) \log P(c) + \sum_c \int_{\boldsymbol{x}} p(c, \boldsymbol{x}) \log \frac{p(c, \boldsymbol{x})}{p(\boldsymbol{x})} \\
&= \sum_c \int_{\boldsymbol{x}} p(c, \boldsymbol{x}) \log \frac{p(c, \boldsymbol{x})}{P(c)p(\boldsymbol{x})} \\
&= \sum_c \int_{\boldsymbol{x}} p(\boldsymbol{x}, c) \log \frac{p(\boldsymbol{x}, c)}{P(c)p(\boldsymbol{x})} \\
&= KL\Big(p(\boldsymbol{x}, c), P(c)p(\boldsymbol{x})\Big) \qquad (2.5)
\end{aligned}$$

When $p(\boldsymbol{x}, c) = P(c)p(\boldsymbol{x})$, then the MI between C and X equals zero which essentially proves their independence of each other. MI can also be interpreted as the *Kullback-Leibler* divergence $KL(,)$ between $p(\boldsymbol{x}, c)$ and the product of two distributions $P(c)$ and $p(\boldsymbol{x})$ [54].

2.1.1 Estimating MI: non-parametric approach

Estimating MI is a non-trivial task. Histogram-based approach is suitable for low-dimensional data and performs poorly in high-dimansional case [54]. The sparse nature of high-dimensional data makes it difficult for histogram based estimation. To overcome this, a non-parametric estimation based on parzen window has been proposed [54] following the formulation of Renyi's non-parametric entropy [58]. Torkkola proposed a non-parametric quadratic measure of MI estimation, named as Quadratic Mutual Information (QMI). His proposed derivation of MI is inspired by Kapur [59]. Kapur argues that the third axiom of Shanon's entropy can be relaxed in certain circumstances. He tries to establish that instead of evaluating entropy of a distribution, we often focus on finding a distribution d_m that minimizes/maximizes that entropy. In this situation, the axioms for deriving the divergence measure (MI) can be relaxed and that optimization process can generate the desired distribution d_m. This theory

leverages the options of a large number of entropy measure, one such outcome is Renyi's entropy.

Following the above relaxation approach, Kapur also presented several non-parametric estimates of MI. Among those measures presented in [59], one is the following,

$$D^{\alpha}(P_1, P_2) = \frac{1}{\alpha(\alpha-1)} \sum_{i=1}^{n} [p_i^{\alpha} - \alpha p_i q_i^{\alpha-1} + (\alpha - 1) q_i^{\alpha}], \alpha \neq 0, \alpha \neq 1 \tag{2.6}$$

where P_1, P_2 are two discrete random variables. Now under the following conditions: $\alpha = 2$, ignore a constant and extend the discrete distributions to continuous ones as $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ respectively, the above equation of divergence measure becomes (following [54]),

$$D^2(f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) = \int_{\boldsymbol{x}} (f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}))^2 d\boldsymbol{x} \tag{2.7}$$

It is possible to substitute $f_1(\boldsymbol{x}) = p(\boldsymbol{x}, c)$ and $f_2(\boldsymbol{x}) = p(\boldsymbol{x}), P(c)$ to derive a quadratic divergence measure between a joint distribution $p(\boldsymbol{x}, c)$ and a product of two distributions $P(c)$ and $p(\boldsymbol{x})$. Torkkola [54] also justified that $D^2(,)$ eventually maximizes the lower bound of $KL(p(\boldsymbol{x}, c), P(c)p(\boldsymbol{x}))$, that results in maximization of MI. Therefore, using Equation (2.7), QMI can be formulated as follows,

$$\begin{aligned} \mathcal{I}(X, C) &= \sum_{c} \int_{\boldsymbol{x}} (p(\boldsymbol{x}, c) - p(\boldsymbol{x})P(c))^2 d\boldsymbol{x} \\ &= \sum_{c} \int_{\boldsymbol{x}} p(\boldsymbol{x}, c)^2 d\boldsymbol{x} + \sum_{c} \int_{\boldsymbol{x}} p(\boldsymbol{x})^2 P(c)^2 d\boldsymbol{x} - 2 \sum_{c} \int_{\boldsymbol{x}} p(\boldsymbol{x}, c) p(\boldsymbol{x}) P(c) d\boldsymbol{x} \end{aligned} \tag{2.8}$$

The probability distributions used in $\mathcal{I}(X, C)$ can be estimated by Parzen window method with a Gaussian kernel [60, 61]. A Gaussian function with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is defined as,

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}} e^{\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right)}$$

We need to find expressions for $P(c)$, $p(\boldsymbol{x}, c)$ and $p(\boldsymbol{x})$ to compute MI according to equation (2.8). We will derive these expressions using Parzen window based density estimator [60]. Parzen-windowing estimates a probability density function (PDF) $p(\boldsymbol{x})$ from which samples were derived. Each observation is examined underneath a window function and size of the window determines the influence of that observation towards other samples inside that window. In this way, each sample $\boldsymbol{x}$ contributes to the PDF estimate. While doing this, each sample is considered with equal importance. This assumption can be challenged in domain adaptation setting as we need to deal with two different distributions and our final feature space will be constructed with samples sharing similar underlying structures across two domains. Therefore, the approach will be not to assign equal importance (weight) for each sample. The closed form expression of MI presented in [53] is extended to a generalized version incorporating instance weighting and soft class assignment for target domain data. Hence, the main contribution to address domain adaptation problem using mutual information will be i) inducing soft class assignment into MI formulation for target domain data and ii) inducing instance weighting into QMI formulation to reduce the impact of unrelated samples (from both source and target domain) in learning a common subspace. Based on these attempts, we will derive closed-form expressions for,

a. Quadratic Mutual Information with Uniform Weighting of samples (UQMI)

b. Quadratic Mutual Information with Non-uniform Weighting of samples (WQMI).

In this chapter, the formulation of UQMI between data and corresponding class labels is presented. The goal is to learn a common subspace based on maximization of QMI between projected data and corresponding class labels. In next chapter, we will elaborately discuss about the WQMI approach. Each of this approach involves soft-labeling of samples i.e. instead of hard label annotation, each data point is associated

with a distribution of class labels.

2.1.2 QMI with uniform instance weighting (UQMI)

As stated earlier, we will now expand Equation (2.8) by substituting appropriate estimations of $P(c)$, $p(\boldsymbol{x}, c)$ and $p(\boldsymbol{x})$.

Evaluating $P(\boldsymbol{x})$: A Parzen window density estimation of $p(\boldsymbol{x})$ using IID drawn samples $\boldsymbol{x}_i$ is,

$$p(\boldsymbol{x}) = \sum_{i=1}^{n} P(\boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2\boldsymbol{I}\right)$$

where n is the cardinality of data set. As the samples are un-weighted or uniformly weighted, $P(\boldsymbol{x}_i)$ for each data sample can be expressed as $P(\boldsymbol{x}_i) = \frac{1}{n}$ and therefore $p(\boldsymbol{x})$ becomes,

$$p(\boldsymbol{x}) = \sum_{i=1}^{n} \frac{1}{n}\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2\boldsymbol{I}\right)$$

Evaluating $P(c)$: $P(c)$ is the prior class probability for class $c \in \{1, 2, \ldots, N_c\}$ which can be expressed as,

$$P(c) = \sum_{i=1}^{n} P(c \mid \boldsymbol{x}_i)P(\boldsymbol{x}_i)$$

Now $P(c|\boldsymbol{x}_i)$ is the probability that $\boldsymbol{x}_i$ belongs to a class c. Therefore, it indicates a sample's soft class labeling (probability of a datum being classified as class c). $P(c)$ represents each sample's class conditional probability, summed over all samples. We will denote this measure as S_c now on for notational convenience, that is,

$$P(c) = \sum_{i=1}^{n} P(c|\boldsymbol{x}_i)\frac{1}{n} = S_c$$

Evaluating $p(\boldsymbol{x}, c)$: A Parzen window estimate of the class data distribution $p(\boldsymbol{x}|c)$ is,

$$\begin{aligned} p(\boldsymbol{x} \mid c) &= \frac{1}{P(c)} \sum_{i=1}^{n} P(c, \boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2\boldsymbol{I}\right) \\ &= \frac{1}{P(c)} \sum_{i=1}^{n} P(c \mid \boldsymbol{x}_i)P(\boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2\boldsymbol{I}\right) \end{aligned}$$

$$= \frac{1}{S_c} \sum_{i}^{n} P(c|\boldsymbol{x}_i\,) \left(\frac{1}{n}\right) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i\,, \sigma^2 \boldsymbol{I}\right)$$

Finally, the estimate of $p(\boldsymbol{x}, c)$ i.e. joint pdf of $\boldsymbol{x}_i$ and c takes the following form,

$$\begin{aligned} p(\boldsymbol{x}, c) &= p(\boldsymbol{x} \mid c)\, P(c) \\ &= \left(\frac{1}{S_c} \sum_{i=1}^{n} P(c|\boldsymbol{x}_i\,) \left(\frac{1}{n}\right) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i\,, \sigma^2 \boldsymbol{I}\right) \right) S_c \\ &= \sum_{i=1}^{n} P(c|\boldsymbol{x}_i\,) \left(\frac{1}{n}\right) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i\,, \sigma^2 \boldsymbol{I}\right) \\ &= \frac{1}{n} \sum_{i=1}^{n} P(c|\boldsymbol{x}_i\,) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i\,, \sigma^2 \boldsymbol{I}\right) \end{aligned}$$

Using the expressions for $p(\boldsymbol{x})$, $P(c)$ and $p(\boldsymbol{x}, c)$ derived above, a closed form for Equation (2.8) will be formulated. As Gaussian kernel is used, we can define a centralized kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ as $\boldsymbol{K} = \tilde{\boldsymbol{K}} - \boldsymbol{E}_n\, \tilde{\boldsymbol{K}} - \tilde{\boldsymbol{K}} \boldsymbol{E}_n + \boldsymbol{E}_n\, \tilde{\boldsymbol{K}} \boldsymbol{E}_n$, where $\tilde{\boldsymbol{K}}_{i,j} = \mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\,; \boldsymbol{0}, 2\sigma^2 \boldsymbol{I}\right)$ and $\boldsymbol{E}_n$ is an $n \times n$ matrix with all elements equal to $1/n$. Let $\boldsymbol{\Phi} \in \mathbb{R}^{n \times m}$ represents the mapped data points from raw feature space to a kernel Hilbert space using the mapping function $\psi : \mathcal{X} \to \mathcal{H}$, where m is the dimension of kernel space (which is usually equals to infinity). Therefore, $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$. Following [53], Equation (2.8) can be represented as,

$$\mathcal{I}(X, C) = \mathcal{V}_{in} + \mathcal{V}_{all} - 2\mathcal{V}_{btw} \tag{2.9}$$

where

$$\begin{aligned} \mathcal{V}_{in} &= \sum_{c} \int_{\boldsymbol{x}} p(\boldsymbol{x}, c)^2 d\boldsymbol{x} \\ &= \frac{1}{n^2} \sum_{c} \left(\sum_{i=1}^{n} P(c|\boldsymbol{x}_i\,) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i\,, \sigma^2 \boldsymbol{I}\right) \right)^2 \\ &= \frac{1}{n^2} \sum_{c} \sum_{i=1}^{n} \sum_{j=1}^{n} P(c|\boldsymbol{x}_i\,) P(c|\boldsymbol{x}_j\,) \mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\,; 0, 2\sigma^2 \boldsymbol{I}\right) \\ &= \frac{1}{n^2} \sum_{c} \sum_{i=1}^{n} \sum_{j=1}^{n} P(c|\boldsymbol{x}_i\,) P(c|\boldsymbol{x}_j\,) \boldsymbol{K}_{i,j} \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{n^2} \sum_c \boldsymbol{z}_c{}^T \boldsymbol{K} \boldsymbol{z}_c \\
&= \frac{1}{n^2} \sum_c \mathrm{tr}\{ \boldsymbol{K} \boldsymbol{z}_c \, \boldsymbol{z}_c{}^T \} \\
&= \frac{1}{n^2} \mathrm{tr} \left\{ \boldsymbol{\Phi} \boldsymbol{\Phi}^{\mathrm{T}} \sum_c \boldsymbol{z}_c \, \boldsymbol{z}_c^T \right\} \\
&= \frac{1}{n^2} \mathrm{tr} \left\{ \boldsymbol{\Phi}^{\mathrm{T}} \left(\sum_c \boldsymbol{z}_c \, \boldsymbol{z}_c^T \right) \boldsymbol{\Phi} \right\}
\end{aligned}
$$

Here $\boldsymbol{z}_c = [P(c|\boldsymbol{x}_1), P(c|\boldsymbol{x}_2), \ldots, P(c|\boldsymbol{x}_n)]^T \in \mathbb{R}^{n \times 1}$. $\mathcal{V}_{in}$ essentially represents within class potential [54] i.e. interactions between pairs of samples inside a class, summed over all classes. Therefore, maximizing $\mathcal{I}(X, C)$ will result in maximizing interactions between every pair of samples within a class c irrespective of their originating domains. Hence, this term will play a vital role in bridging data distributions of a specific class from two domains i.e. minimizing divergence in class conditional distributions (alternatively, divergence between $P_s(c|\boldsymbol{x}_s)$ and $P_t(c|\boldsymbol{x}_t)$ will be reduced between source and target domains).

$$
\begin{aligned}
\mathcal{V}_{all} &= \sum_c \int_{\boldsymbol{x}} p(\boldsymbol{x})^2 P(c)^2 d\boldsymbol{x} \\
&= \sum_c \left(\sum_{i=1}^{n} \frac{1}{n} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I}\right) \right)^2 S_c^2 \\
&= \sum_c S_c^2 \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{n^2} \mathcal{N}\left(\boldsymbol{x}_i; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{x}_j; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I}\right) \\
&= \sum_c S_c^2 \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{n^2} \mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j; \boldsymbol{0}, 2\sigma^2 \boldsymbol{I}\right) \\
&= \frac{1}{n^2} \left(\sum_c S_c^2 \right) \sum_{i=1}^{n} \sum_{j=1}^{j=1} \boldsymbol{K}_{i,j} \\
&= \frac{1}{n^2} \left(\sum_c S_c^2 \right) \boldsymbol{1}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{1} \\
&= \frac{1}{n^2} \left(\sum_c S_c^2 \right) \mathrm{tr}\{ \boldsymbol{K} \boldsymbol{1} \boldsymbol{1}^{\mathrm{T}} \}
\end{aligned}
$$

$$= \frac{1}{n^2}\left(\sum_c S_c^2\right) \mathrm{tr}\big\{\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\,\mathbf{1}\mathbf{1}^{\mathrm{T}}\big\}$$
$$= \frac{1}{n^2}\left(\sum_c S_c^2\right) \mathrm{tr}\big\{\boldsymbol{\Phi}^{\mathrm{T}}\,\mathbf{1}\mathbf{1}^{\mathrm{T}}\,\boldsymbol{\Phi}\big\}$$

where $\mathbf{1}= [1, 1, \ldots, 1]^T \in \mathbb{R}^{n\times 1}$. $\mathcal{V}_{all}$ will control interactions between every pair of samples weighted by squared sum of class prior probability, irrespective of their class labels and originating domains which will eventually ensure interactions between source and target domain data in our domain adaptaion setting.

$$\begin{aligned}
\mathcal{V}_{btw} &= \sum_c \int_{\boldsymbol{x}} p(\boldsymbol{x}, c)P(c)p(\boldsymbol{x})d\boldsymbol{x} \\
&= \sum_c \left(\frac{1}{n}\sum_{i=1}^{n} P(c|\boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2\boldsymbol{I}\right)\right) S_c \left(\sum_{j=1}^{n}\frac{1}{n}\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_j, \sigma^2\boldsymbol{I}\right)\right) \\
&= \sum_c S_c \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{n^2}P(c|\boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2\boldsymbol{I}\right)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_j, \sigma^2\boldsymbol{I}\right) \\
&= \sum_c S_c \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{n^2}P(c|\boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j; \mathbf{0}, 2\sigma^2\boldsymbol{I}\right) \\
&= \sum_c S_c \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{n^2}P(c|\boldsymbol{x}_i)\boldsymbol{K}_{i,j} \\
&= \frac{1}{n^2}\sum_c S_c\, \boldsymbol{z}_c{}^T\boldsymbol{K}\cdot\mathbf{1} \\
&= \frac{1}{n^2}\sum_c \mathrm{tr}\big\{S_c\left(\boldsymbol{K}\cdot\mathbf{1}\cdot\boldsymbol{z}_c^T\right)\big\} \\
&= \frac{1}{n^2}\mathrm{tr}\left\{\sum_c S_c\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\cdot\mathbf{1}\cdot\boldsymbol{z}_c^T\right)\right\} \\
&= \frac{1}{n^2}\mathrm{tr}\left\{\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\cdot\mathbf{1}\cdot\sum_c S_c\boldsymbol{z}_c^T\right\} \\
&= \frac{1}{n^2}\mathrm{tr}\left\{\boldsymbol{\Phi}^{\mathrm{T}}\left(\mathbf{1}\cdot\sum_c S_c\boldsymbol{z}_c^T\right)\boldsymbol{\Phi}\right\}
\end{aligned}$$

$\mathcal{V}_{btw}$ essentially represents interactions between class data points against all points. In other words, it corresponds to interactions of a specific class c and all the available samples weighted by the prior of class c. According to Equation (2.9), minimizing this

term will eventually maximize MI between data points and corresponding class labels.

Substituting the expressions for $\mathcal{V}_{in}$, $\mathcal{V}_{all}$ and $\mathcal{V}_{btw}$ into Equation (2.9), the closed form expression of QMI will be,

$$\begin{aligned}\mathcal{I}(X,C) &= \frac{1}{n^2}\mathrm{tr}\Big\{\mathbf{\Phi}^{\mathrm{T}}\left(\sum_c \boldsymbol{z}_c\,\boldsymbol{z}_c^T\right)\mathbf{\Phi}\Big\} + \frac{1}{n^2}\left(\sum_c S_c^2\right)\mathrm{tr}\Big\{\mathbf{\Phi}^{\mathrm{T}}\,\mathbf{1}\mathbf{1}^{\mathrm{T}}\,\mathbf{\Phi}\Big\} - \frac{2}{n^2}\mathrm{tr}\Big\{\mathbf{\Phi}^{\mathrm{T}}\left(\mathbf{1}\cdot\sum_c S_c\boldsymbol{z}_c^T\right)\mathbf{\Phi}\Big\} \\ &= \mathrm{tr}\Big\{\mathbf{\Phi}^{\mathrm{T}}\left((\frac{1}{n^2})\left(\sum_c \boldsymbol{z}_c\,\boldsymbol{z}_c^T\right) + \left(\sum_c \frac{S_c^2}{n^2}\right)\mathbf{1}\mathbf{1}^{\mathrm{T}} - 2\cdot\mathbf{1}\left(\sum_c \frac{S_c}{n^2}\right)\boldsymbol{z}_c^T\right)\mathbf{\Phi}\Big\} \\ &= \mathrm{tr}\Big\{\mathbf{\Phi}^{\mathrm{T}}\,\boldsymbol{M}\mathbf{\Phi}\Big\}. \end{aligned} \tag{2.10}$$

where

$$\boldsymbol{M} = (\frac{1}{n^2})\left(\sum_c \boldsymbol{z}_c\,\boldsymbol{z}_c^T\right) + \left(\sum_c \frac{S_c^2}{n^2}\right)\mathbf{1}\mathbf{1}^{\mathrm{T}} - 2\cdot\mathbf{1}\left(\sum_c \frac{S_c}{n^2}\right)\boldsymbol{z}_c^T. \tag{2.11}$$

$\boldsymbol{M}$ matrix is the core of the proposed subspace learning procedure. It essentially characterizes the subspace which is optimized based on maximization of mutual information. The soft-labeling has been induced to original QMI formulation of [53] by the vector $\boldsymbol{z}_c = [P(c|\boldsymbol{x}_1), P(c|\boldsymbol{x}_2), \ldots, P(c|\boldsymbol{x}_n)]^T \in \mathbb{R}^{n\times 1}$, we refer to this formulation as QMI-S. Now maximizing QMI will eventually boils down to a trace optimization problem. In the next section, we will provide the objective function for learning a feature subspace cast as a trace ratio optimization approach.

2.1.3 Subspace Learning By Maximizing QMI-S

Now we can describe the objective function of the proposed algorithm for subspace learning based on the maximization of QMI between data and corresponding class labels, where instead of using hard labeling for each data point, a class label distribution (soft labeling) is used. Our formulation is inspired by the approach in [53]. Given a set of labeled data $\boldsymbol{X}_s \in \mathbb{R}^{n\times d}$ from source domain ($\mathcal{D}_s$) and unlabeled data $\boldsymbol{X}_t \in \mathbb{R}^{n\times d}$ from target domain ($\mathcal{D}_t$), our input data matrix consists of $\boldsymbol{X} = [\boldsymbol{X}_s, \boldsymbol{X}_t] \in \mathbb{R}^{n\times d}$, where $n = n_s + n_t$ and n_s and n_t represent source and target data size respectively. Note that both $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$ are sampled from the same d-dimensional feature space.

Computing QMI requires labeled data, whereas our target domain data is unlabeled. To overcome this issue, soft assignment of class labels for target data points are used. Each of these label distributions are initialized with uniform distribution of class labels (described in Section 2.1.6) and updated gradually in an iterative fashion.

As stated earlier, MI between X and C is measured in kernel space using mapped data points $\boldsymbol{\Phi}$. Therefore, data are first transformed into kernel space. Kernel matrix $\boldsymbol{K}$ will be computed using kernel trick, $\boldsymbol{K}(\boldsymbol{x}_i\,, \boldsymbol{x}_j\,) =< \phi(\boldsymbol{x}_i\,), \phi(\boldsymbol{x}_j\,) >$, where $\phi(\boldsymbol{x})$ is a mapped datum from original feature space to a reproducing kernel Hilbert space using the mapping function $\psi : \mathcal{X} \rightarrow \mathcal{H}$. We have mainly used Gaussian kernel in our thesis. By kernelizing the data matrix $\boldsymbol{X}$, the goal is to find a k-dimensional feature subspace ($k << m$) using a linear transformation such that the QMI between projected data and corresponding class labels (according to Equation (2.10)) is maximized. In other words, we need to find a projection matrix $\boldsymbol{W} \in \mathbb{R}^{m \times k}$ for this transformation. $\boldsymbol{W}$ will transform the input data from kernel space to a low-dimensional feature space. Let $\boldsymbol{w}_i$ is a projection vector from $\boldsymbol{W}$ with $\boldsymbol{w}_i \in \mathbb{R}^m$. We can impose a constraint on $\boldsymbol{w}_i$'s to be in the range of $\boldsymbol{\Phi}$ i.e. they will span the kernel feature space. Therefore, $\boldsymbol{w}_i$ can be expressed as a linear combination of $\phi(\boldsymbol{x})$ i.e. $\boldsymbol{w}_i = \sum_{j=1}^{n} a_{i,j}\phi(\boldsymbol{x}_j\,) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{a}_i\,$, where $\boldsymbol{a_i} \in \mathbb{R}^{n \times 1}$ is a coefficient vector. Projection matrix $\boldsymbol{W}$ can be constructed by arranging each $\boldsymbol{w}_i$ vector in columns such that $\boldsymbol{W} = [\boldsymbol{w}_1\,, \boldsymbol{w}_2\,, \ldots, \boldsymbol{w}_n\,] = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{A}$, where $\boldsymbol{A} = \{\boldsymbol{a}_i\,\}_{i=1}^{n} \in \mathbb{R}^{n \times k}$. Each column of $\boldsymbol{A}$ is $\boldsymbol{a}_i\,$, that is, $\boldsymbol{A}_i = \boldsymbol{a}_i\,$. Using $\boldsymbol{W}$, the projected data, $\boldsymbol{X}_P \in \mathbb{R}^{n \times k}$ can be obtained by $\boldsymbol{X}_P = \boldsymbol{\Phi}\boldsymbol{W}$. Our goal is to maximize QMI in the projected space. Therefore, the QMI between data and corresponding class distributions in the learned space will be following,

$$\begin{aligned}\mathcal{I}_p(\boldsymbol{X}_p\,, C) &= \mathrm{tr}\big\{(\boldsymbol{X}_p\,)^T\boldsymbol{M}\boldsymbol{X}_p\,\big\} \\ &= \mathrm{tr}\big\{(\boldsymbol{\Phi}\boldsymbol{W})^T\boldsymbol{M}(\boldsymbol{\Phi}\boldsymbol{W})\big\}\end{aligned}$$

$$
\begin{aligned}
&= \mathrm{tr}\big\{\boldsymbol{W}^T \boldsymbol{\Phi}^T \boldsymbol{M} \boldsymbol{\Phi} \boldsymbol{W}\big\} \\
&= \mathrm{tr}\big\{\boldsymbol{A}^T \boldsymbol{\Phi}\boldsymbol{\Phi}^T \boldsymbol{M} \boldsymbol{\Phi}\boldsymbol{\Phi}^T \boldsymbol{A}\big\} \\
&= \mathrm{tr}\big\{\boldsymbol{A}^{\mathrm{T}} \boldsymbol{K}\boldsymbol{M}\boldsymbol{K}\boldsymbol{A}\big\} \qquad (2.12)
\end{aligned}
$$

Finding an optimal $\boldsymbol{W}$ now boils down to finding $\boldsymbol{A}$ with a goal to maximizing QMI. Hence the objective function for learning the QMI-maximized subspace can be represented as,

$$
\boldsymbol{A}^* = \arg \max_{\boldsymbol{A}^{\mathrm{T}} \boldsymbol{K}\boldsymbol{A} = \boldsymbol{I}} \mathrm{tr}\big\{\boldsymbol{A}^{\mathrm{T}} \boldsymbol{K}\boldsymbol{M}\boldsymbol{K}\boldsymbol{A}\big\} \qquad (2.13)
$$

The constraint in Equation (2.13) is derived from the orthogonality of the projection matrix $\boldsymbol{W}$ [53]. Here $\boldsymbol{I}$ represents identity matrix. This constraint can be deduced as, $\boldsymbol{W}^{\mathrm{T}} \boldsymbol{W} = \boldsymbol{I} \Rightarrow (\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{A})^{\mathrm{T}} (\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{A}) = \boldsymbol{I} \Rightarrow \boldsymbol{A}^{\mathrm{T}} \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{A} = \boldsymbol{I} \Rightarrow \boldsymbol{A}^{\mathrm{T}} \boldsymbol{K}\boldsymbol{K}\boldsymbol{A} = \boldsymbol{I}$. We will focus on casting the optimization problem in Equation (2.13) into a trace ratio optimization problem. In the next section, a brief overview on trace ratio optimization is provided.

2.1.4 QMI-S as a trace ratio problem

In the area of machine learning and pattern recognition, dimensionality reduction methods have been practiced and widely used. There exists a good number of supervised and unsupervised dimensionality reduction techniques such as Linear Discriminant Analysis (LDA) [62], Principal Component Analysis (PCA) [52], Local Linear Embedding (LLE) [63], ISOMAP [64] and so on. Almost all of these methods can be generalized as a *trace ratio* optimization problem which has been an active research topic in this area. This unified approach involves searching for a transformation matrix $\boldsymbol{W}$ that maximizes or minimizes a trace ratio of the form $\mathrm{tr}\big\{\boldsymbol{W}^{\mathrm{T}} \boldsymbol{S}_a \boldsymbol{W}\big\} / \mathrm{tr}\big\{\boldsymbol{W}^{\mathrm{T}} \boldsymbol{S}_b \boldsymbol{W}\big\}$ where $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ are symmetric positive definite matrices which are derived from the corresponding dimensionality reduction method.

Therefore, the trace ratio problem will take the following form,

$$\boldsymbol{A}^* = \arg \max_{\boldsymbol{W}^{\mathrm{T}} \boldsymbol{W} = \boldsymbol{I}} \frac{\mathrm{tr}\{\boldsymbol{W}^{\mathrm{T}} \boldsymbol{S}_a \boldsymbol{W}\}}{\mathrm{tr}\{\boldsymbol{W}^{\mathrm{T}} \boldsymbol{S}_a \boldsymbol{W}\}}$$

This optimization problem suffers from an optimal closed form solution. Researchers have come up with an approximate solution by casting the above problem into an alternative ratio trace problem [65] defines as follows,

$$\boldsymbol{A}^* = \arg \max_{\boldsymbol{W}^{\mathrm{T}} \boldsymbol{W} = \boldsymbol{I}} \mathrm{tr}\{(\boldsymbol{W}^{\mathrm{T}} \boldsymbol{S}_a \boldsymbol{W})^{-1} (\boldsymbol{W}^{\mathrm{T}} \boldsymbol{S}_a \boldsymbol{W})\}$$

The ratio trace problem can be solved by generalized eigen value decomposition (GEVD) method as $\boldsymbol{S}_a \boldsymbol{u}_l = \beta_l \boldsymbol{S}_b \boldsymbol{u}_l$, where u_l is the eigen vector corresponding to l-th largest/smallest eigen value. The projection matrix $\boldsymbol{W}$ will be formed by the desired number of eigen vectors.

Researchers have argued that the above approximation approach is often far from optimal and tried to find a better solution. Besides GEVD approach, a number of other non-linear iterative techniques to the trace ratio problem has also been proposed in the literature [65–67]. In our work, the GEVD based solution is utilized because of its simplicity and fast implementation for the QMI-S optimization problem. Later in the thesis, we will incorporate a non-linear approach based on Newton's method to solve the trace ratio problem in the proposed domain adaptation framework.

2.1.5 Solving QMI-S objective function

In order to cast the QMI-S optimization as a trace ratio optimization problem discussed above, the following two constraints have been applied to Equation (2.13) according to [53],

i. **Constraint 1:** Often the class data happen to be imbalanced in either domains which makes the $\boldsymbol{M}$ matrix non-symmetric. To cast Equation (2.13) into a trace

ratio problem, $\boldsymbol{M}$ needs to be symmetric. For a real valued square matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$, the identity $\boldsymbol{x}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{M'} \boldsymbol{x}$ holds, where $\boldsymbol{M'} = \frac{\boldsymbol{M}+\boldsymbol{M}^{\mathrm{T}}}{2}$ is a symmetric form of $\boldsymbol{M}$. Therefore, $\boldsymbol{M}$ can be symetricized as $\boldsymbol{M'} = \frac{\boldsymbol{M}+\boldsymbol{M}^{\mathrm{T}}}{2}$ and $\boldsymbol{M'}$ will be used instead of $\boldsymbol{M}$ from now on.

ii. **Constraint 2:** To make the optimization problem in Equation (2.13) well-posed, an additional constraint has been imposed, which is, to ensure the embedding data vectors have unit covariance. In other words, $(1/n)(\boldsymbol{\Phi W})^{\mathrm{T}} (\boldsymbol{\Phi W}) = \boldsymbol{I} \Rightarrow (\boldsymbol{\Phi \Phi}^{\mathrm{T}} \boldsymbol{A})^{\mathrm{T}} (\boldsymbol{\Phi \Phi}^{\mathrm{T}} \boldsymbol{A}) = n\boldsymbol{I} \Rightarrow (\boldsymbol{KA})^{\mathrm{T}} (\boldsymbol{KA}) = n\boldsymbol{I} \Rightarrow \boldsymbol{A}^{\mathrm{T}} \boldsymbol{KKA} = n\boldsymbol{I}$. This is to ensure that the embedding data points are different from each other, a much similar approach used in PCA.

Applying the above two modifications in to Equation (2.13), the final objective function for subspace learning will take the following form,

$$\boldsymbol{A}^* = \arg \max_{\boldsymbol{A}^{\mathrm{T}} \boldsymbol{KA} = \boldsymbol{I}} \frac{\mathrm{tr}\{\boldsymbol{A}^{\mathrm{T}} \boldsymbol{KM'KA}\}}{\mathrm{tr}\{\boldsymbol{A}^{\mathrm{T}} \boldsymbol{KKA}\}} \tag{2.14}$$

The above equation is in the form of trace ratio optimization problem and hence an approximate solution can be generated by generalized eigen value decomposition as follows,

$$\boldsymbol{KM'KU} = \boldsymbol{KKU\Lambda}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix of eigen values in decreasing order and $\boldsymbol{U}$ is a matrix with colums as the corresponding eigen vectors. Also to satisfy the constraint $\boldsymbol{A}^{\mathrm{T}} \boldsymbol{KA} = \boldsymbol{I}$, each eigen vector $\boldsymbol{u}$ or column of $\boldsymbol{U}$ is divided by $\sqrt{\boldsymbol{uKu}}$ to get a normalized eigen vector. The columns of $\boldsymbol{U}$ will constitute the optimal $\boldsymbol{A}^*$. Finally, we can generate the projected data onto the QMI-S maximixed subspace by $\boldsymbol{X}_p = \boldsymbol{\Phi W} = \boldsymbol{\Phi \Phi}^{\mathrm{T}} \boldsymbol{A} = \boldsymbol{KA}^*$. As the rank of $\boldsymbol{M'}$ is $N_c - 1$, the first $N_c - 1$ number of columns in $\boldsymbol{U}$ will be used to create the projected data i.e. $\boldsymbol{A}^* = \{\mathbf{u}_l\}_{l=1}^{N_c-1}$. This approach for maximizing MI in the projected subspace has been proposed by Bouzas

et al [53].

Alternative solution: Besides the above mentioned method, a more robust approach for finding $\boldsymbol{A}^*$ is provided, which will give us numerical stability in precise computation of eigen decomposition. See Algorithm 1 for the detailed procedure. The projected data in QMI-S subspace is, $\boldsymbol{X}_p = \boldsymbol{K}\boldsymbol{A}^* = \boldsymbol{K}\boldsymbol{K}^{-\frac{1}{2}}\boldsymbol{V} = \boldsymbol{K}^{\frac{1}{2}}\boldsymbol{V}$. Therefore, the computation of $\boldsymbol{A}^*$ in obtaining projected data $\boldsymbol{X}_p$ is implicitly avoided .

Algorithm 1 Procedure for finding optimal $\boldsymbol{A}^*$

1: Re-write Equation (2.14) as, $\boldsymbol{K}\boldsymbol{M'}\boldsymbol{K}\boldsymbol{U} = \boldsymbol{K}\boldsymbol{K}\boldsymbol{U}\boldsymbol{\Lambda}$.

2: Substituting $\boldsymbol{K}\boldsymbol{U}$ with a new variable $\boldsymbol{V}$ *i.e.* $\boldsymbol{K}\boldsymbol{U} = \boldsymbol{V}$ and multiplying both sides of above equation by $\boldsymbol{K}^{-1}$, we get,

$$\boldsymbol{M'}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}.$$

3: Solve the above eigen value decomposition problem, where $\boldsymbol{\Lambda}$ is the diagonal matrix containing eigen values in descending order and $\boldsymbol{V}$ is the matrix of corresponding eigen vectors.

4: Select the $N_c - 1$ eigen vectors from $\boldsymbol{V}$ with the largest eigen values.

5: The optimal $\boldsymbol{A}^*$ that satisfies $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{A} = \boldsymbol{I}$ will be $\boldsymbol{K}^{-\frac{1}{2}}\boldsymbol{V}$. This can be verified as, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{A} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{K}^{-\frac{1}{2}}\boldsymbol{K}\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{V} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = I$, where $\boldsymbol{V}$ is orthogonal matrix.

2.1.6 Iterative update of soft-labeling and maximization of QMI-S

The proposed framework for domain adaptation is an iterative approach based on soft-labeling induced QMI maximization. At each iteration, target data predictions will be updated with the help of labeled source data and this will eventually update the $\boldsymbol{M}$ matrix for the next iteration. As stated earlier, input data consists of both source and target domain samples. The intension is to align these two distributions such that a classifier trained using projected source data can be applied to predict labels of target data. Therefore, the proposed iterative framework consists of two

main steps described below (see Figure 2.1).

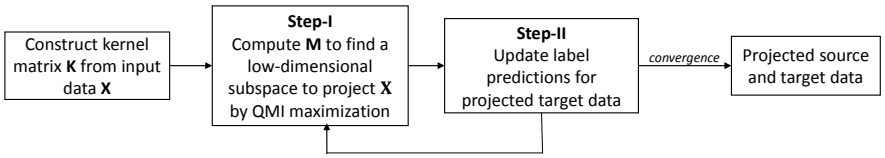

Figure 2.1: Proposed domain adaptation framework with two main steps.

The input to our proposed method are source and target samples, where source samples are associated with corresponding class labels (which are assumed to be known in the domain adaptation setting) and target samples are initialized with uniform uncertainty of class labels. Source and target data belong to the same d-dimensional feature space. Let $\boldsymbol{X}_s \in \mathbb{R}^{n_s \times d}$ represent n_s samples from source domain and $\boldsymbol{X}_t$ represent n_t samples from target domain. Hence the data matrix consists of $\boldsymbol{X} = [\boldsymbol{X}_s\,; \boldsymbol{X}_t\,] \in \mathbb{R}^{n \times d}$, where $n = n_s + n_t$. As an initialization of the proposed iterative algorithm, each target sample is assigned a discrete uniform distribution of class labels i.e for each target sample $\boldsymbol{x}_i \in \boldsymbol{X}_t$, the label distribution is,

$$p(c_j|\boldsymbol{x}_i\,) = \frac{1}{N_c},\ \text{for each } c_j \in \{1, 2, \ldots N_c\}.$$

On other hand, the source samples are annotated with true labels, representing as a label vector, $\boldsymbol{Y}_s = [y_1, y_2, \ldots, y_{n_s}]^T$. To be consistent with the target samples, each $\boldsymbol{x_i} \in \boldsymbol{X}_s$ can also be associated with a label distribution using the following representation,

$$p(c_j|\boldsymbol{x}_i\,) = \begin{cases} 1 & \text{if } c_j = y_i \\ 0 & \text{otherwise} \end{cases}$$

Therefore, for source data, the label distribution is skewed and for target data, it is flat. The expectation is to convert the flat label distribution biased towards its true label i.e. at the end of learning process, a skewed label distribution is expected for

each target sample. The two major steps involved in the subspace learning are,

Step I: After initialization of data with corresponding label distributions, this step involves in learning a low-dimensional subspace using a linear transformation. This transformation has been designed to maximize QMI-S between data and corresponding class labels in the projected space. The optimization objective has been provided in Equation 2.14.

Step II: Once we get the set of projected source and target samples, the target label predictions are updated. A K-nn classifier is trained using projected source data and applied to projected target data in order to obtain target label predictions. Usually, $K > 1$ is chosen for the classifier so that each target point is predicted with a probability distribution of class labels.

With this approach, the distribution divergence between source and target domain is eventually minimized in the projected subspace. According to Equation (2.14), at the beginning of this iterative process, $\boldsymbol{X}$ is mapped from its original feature space $\mathcal{X}$ to a kernel space using Gaussian kernel. The contribution of this kernel mapping in DA context is two-fold:

1. It will help capture the non-linear structure of data and
2. choosing sufficiently large kernel size (σ), kernel matrix $\boldsymbol{K}$ will help project the data (across domains) with similar underlying structure to a close proximity in the learned subspace.

The proposed methodology is summarized in Algorithm 2.

***Convergence* criterion:** The proposed algorithm will reach convergence when sub-

space change in two successive iterations will be negligible. The subspace is defined by the basis vectors $\boldsymbol{V}$ according to Algorithm 1. Hence, this change of two k-dimensional subspaces in successive iterations can be approximated as a subspace distance on a Grassmannian [68]. One such distance metric measures the principal angle θ between $\boldsymbol{V}_i$ and $\boldsymbol{V}_{i+1}$ of iterations i and $i+1$ respectively [68, 69]. A *convergence* threshold ϵ is set and the algorithm terminates when $\theta \leq \epsilon$. At this state, class predictions for target data become stabilized and exhibit no changes in following iterations.

Algorithm 2 Subspace learning based on iterative QMI-S.

1: **Input**: Data matrix $\boldsymbol{X}$=[$\boldsymbol{X}_s$;$\boldsymbol{X}_t$]$\in \mathbb{R}^{n\times d}$ where source data $\boldsymbol{X}_s \in \mathbb{R}^{n_s\times d}$ and target data $\boldsymbol{X}_t \in \mathbb{R}^{n_t\times d}$, source data labels $\boldsymbol{Y} = [y_1, y_2, \ldots, y_s]^T$.

2: **Output**: $\boldsymbol{X}_p \in \mathbb{R}^{n\times k}$, k-dimensional projected data.

3: **Initialization**: For $\boldsymbol{x}_i \in \boldsymbol{X}_t$, $P(c|\boldsymbol{x}_i) = \frac{1}{N_c}$ for each $c \in \{1, 2, \ldots N_c\}$. For $\boldsymbol{x}_i \in \boldsymbol{X}_s$, $P(c|\boldsymbol{x}_i) =$ 1 if $c = y_i$ and $P(c|\boldsymbol{x}_i) = 0$ otherwise, for each $c \in \{1, 2, \ldots N_c\}$.

4: Compute a centralized Gaussian kernel matrix, $\boldsymbol{K}\in \mathbb{R}^{n\times n}$.

5: **repeat**

Step-I:

6: Compute $\boldsymbol{M}$ matrix using Eq.(2.11).

7: Compute $\boldsymbol{M'}$ as $\boldsymbol{M'} = \frac{\boldsymbol{M}+\boldsymbol{M}^{\mathrm{T}}}{2}$.

8: Solve standard eigen problem, $\boldsymbol{M'V} = \boldsymbol{V\Lambda}$.

9: Compute projected data $\boldsymbol{X}_p = \boldsymbol{K}^{\frac{1}{2}}\boldsymbol{V}$.

Step-II:

10: Train a classifier f using projected source data $\boldsymbol{X}_p^s$ and apply it to update $P(c|\boldsymbol{x}_i^t)$ with soft-labeling.

11: **until** *convergence*.

2.1.7 Classification in target domain

Once the projected source and target data in the QMI-S optimized subspace are obtained, a classifier is trained with projected source data and corresponding class labels. The object classes are same across domains. Therefore, the classifier can be

trained with projected source data and tested against the target domain data for classification. In the experiment section, we will try with two different classifiers, i) K nearest neighbor classifier and ii) support vector machine (SVM).

2.2 Dataset and Experiments

The proposed method is implemented and tested against popular benchmark datasets. **Office** is a widely used image database for domain adaptation [42]. It contains three different domains (Amazon, DSLR, Webcam) of images captured with varied settings and image conditions. It contains a total of 4652 images with 31 image categories. A brief description of each domain is following,

Amazon: Images of Amazon domain are downloaded from online merchant site amazon.com. These images are usually captured in white background using studio lighting environment and typically posed in a canonical point of view. On an average, 90 images per category are collected from this site.

DSLR: It contains images captured with high-resolution DSLR camera in natural environment setting with resolution 42888×28848. On average, each object is captured 3 times from each viewpoint. This domain contains a total of 423 images.

Webcam: It contains images captured with low-resolution web camera. The resolution of each image is 640×480. Images also suffer from noise and other artifacts because of the environment. This domain contains a total of 795 images.

Sample images from three domains are shown in Figure 2.2. These three domains of images show significant variations from each other and provide a substantial ground for studying domain adaptation problem. It has been tested that if a classifier is trained using one domain (source) and applied to another domain (target), the classification accuracy is adversely affected. Additionally, we will use **Caltech-256** which is a standard dataset for object recognition [70]. It has 30,607 images of 256

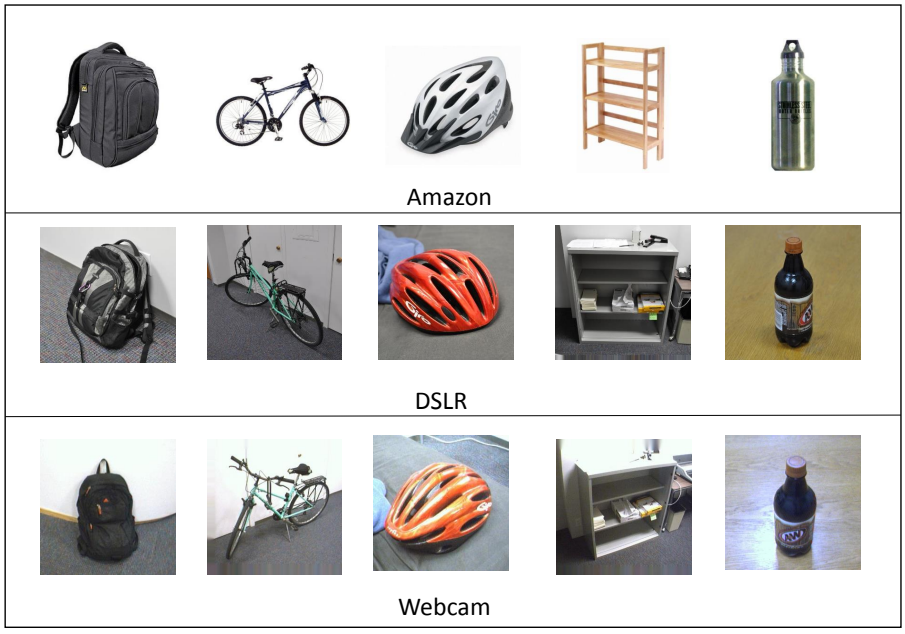

Figure 2.2: Sample images from three different domains.

categories. Therefore, experiments are conducted using these 4 domains with 10 common categories selected from each of them. The 10 common categories selected are Bike, BackPack, Calculator, Headphone, Keyboard, Laptop, Monitor, Mouse, Mug, Projector. From these 4 domains, a total of 12 domain adaptation sub-problems can be created, each of which involves one source($\mathcal{D}_s$) and one target($\mathcal{D}_t$) domain. The total number of samples from each domain is 958 (Amazon), 295 (Webcam), 157 (DSLR), 1123 (Caltech).

Data preparation: We will use the image representation published by Gong *et al.* [22]. SURF features are extracted from images and a codebook of 800 visual words is computed by K-means clustering on a subset of Amazon images. Each image is encoded as a 800-dimensional histogram. The histogram is normalized using z-scoring along each dimension to maintain zero mean and unit variance. We will follow the experimental protocol of [8, 23].

Parameter setup: Input data are whitened with PCA preserving 95% of the data variance. Following heuristic in the literature [6], σ of Gaussian kernel is set to median of the pair-wise distances of data in raw feature space. To update target data labels inside the **repeat** loop of Algorithm 2, we used K-nn classifier with K set to $log(n_s) + 1$ [71]. Also for *convergence*, $\epsilon = 1 \times 10^{-4}$ is used.

In next sections, experiments are conducted with different types of setup and analyzed the effect of each of them on the proposed itrative QMI-S algorithm. Each type of experiment has been abbreviated with letters ($\mathcal{A}, \mathcal{B}, \mathcal{C}$ etc.) and the result tables are organized using these letter references. This will ensure a comparative picture of each different settings along with their contributions to the framework.

2.2.1 Experiment $\mathcal{A}$

Here, the data and parameter are set up using the procedure mentioned above. After obtaining the projected data, the classification is conducted on target domain using a K-nn classifier with $K = 1$. Each subproblem is denoted as $source \rightarrow target$ meaning source classifier is applied to target data in QMI-S subspace. C, A, W and D represent Caltech-256, Amazon, Webcam and DSLR domains respectively. Also to prove the superiority of the iterative QMI-S method, another experiment is conducted where target data are assigned with hard class labels during label predictions i.e. each target point is assigned a unique class label in STEP-II of the proposed algorithm. The Table 2.1 shows the comparative results where hard labeling based approach is denoted as QMI-H. The result shows that soft label assignment is effective over hard class labeling. The reason is that iterative QMI-S smoothly updates target data labels, whereas QMI-H is aggressive and once a target point is falsely labeled, it is prone to stick with this label in following iterations. Nevertheless, applying Gaussian

Table 2.1: Classification accuracy(%) of target data for 12 different sub-problems. Each sub-problem is in the form of *source* $\rightarrow$ *target*, where C(Caltech-256), A(Amazon), W(Webcam) and D(DSLR) indicate four different domains.

Methods	$C \rightarrow A$	C $\rightarrow$ W	C $\rightarrow$ D	A $\rightarrow$ C	A $\rightarrow$ W	A $\rightarrow$ D	W $\rightarrow$ C	W $\rightarrow$ A	W $\rightarrow$ D	D $\rightarrow$ C	D $\rightarrow$ A	D $\rightarrow$ W	Avg
QMI-H	55.95	49.49	45.86	**42.12**	42.71	37.58	30.37	35.8	80.89	35.71	38.31	61.02	46.32
QMI-S	**57.72**	**55.93**	**48.41**	41.76	**46.44**	**38.85**	**30.72**	**36.74**	**83.44**	**38.38**	**42.48**	**77.63**	**49.88**

kernel with sufficiently large kernel size (large σ) helped to capture the underlying shared structure of similar data across domains.

Finally, we observe that the class label uncertainty of target data is greatly reduced, when the algorithm reaches *convergence*. As each target data is assigned soft-labeling (assigning class labels with probability), we can approximate the uncertainty measure as $u(\boldsymbol{x}_i) = max(P(c|\boldsymbol{x}_i))$ for each target datum. Higher $u(\boldsymbol{x}_i)$ will indicate lower uncertainty. We reported target data count(%) in projected subspace with $u(\boldsymbol{x}_i) \geq 0.6$ (see Figure 2.3) for each of the 12 sub-problems. Using this criterion, we find that at least 70% of the target data have achieved reduced label uncertainty (increased bias towards a class label) for all of the 12 cases.

In terms of computational cost, the proposed method mainly involves matrix computation and solving generalized eigen value decomposition which can be efficiently implemented using any good software package. The average iteration count of our iterative approach till *convergence* is 24.67, over all 12 sub-problems. Figure 2.4 shows the *convergence* behavior of the framework. As the algorithm proceeds, the distance between learned subspaces (defined in Section 2.1.5) in two successive iterations reaches below a minimum threshold.

Comparison with state-of-the-art methods: The proposed algorithm is compared with 7 different DA methods (Table 2.2). These can be categorized as follows,

- Without adaptation: **Origfeat** and **PCA** indicate the classification accuracy

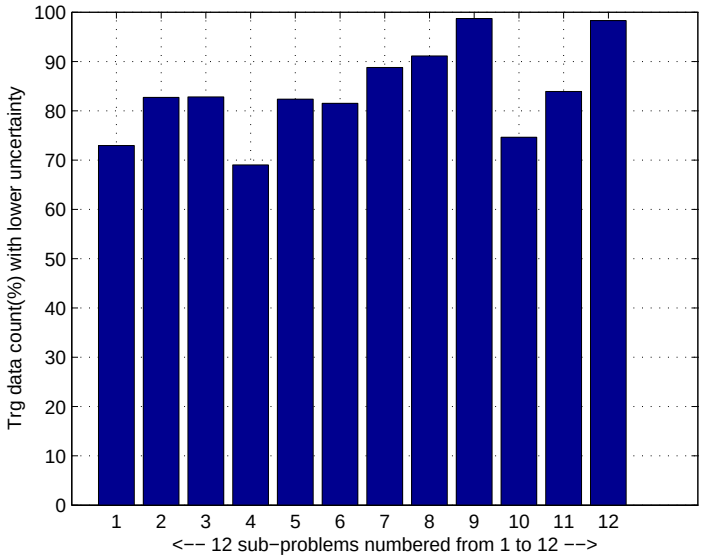

Figure 2.3: Number of target domain data (%) with reduced uncertainty of class labels, upon *convergence* of iterative QMI-S. Each bar represents one sub-problem (*source* $\rightarrow$ *target*) indexed by $\{1, 2, \ldots, 12\}$.

of target data in original 800-dimensional feature space and PCA subspace respectively. The classification in the original feature space and PCA subspace without adaptation is investigated in order to imply the necessity of developing DA methods.

- Adaptation based on subspace alignment: This category of methods focus on learning a shared low-dimensional subspace between source and target domain in order to minimize their divergence difference. In other words, they align the two domains of data such that a classifier trained on projected source data can also classify unlabeled target domain data. It includes geodesic flow kernel (**GFK**) [22], unsupervised subspace alignment (**SA**) [7], transfer component analysis (**TCA**) [9] and transfer feature learning (**TFL**) [23].

- Adaptation based on subspace alignment+instance re-weighting: This approach resembles our proposed framework which mainly focuses on finding similarly distributed source and target data in order to utilize them in subspace learning. It includes transfer joint matching (**TJM**) [8].

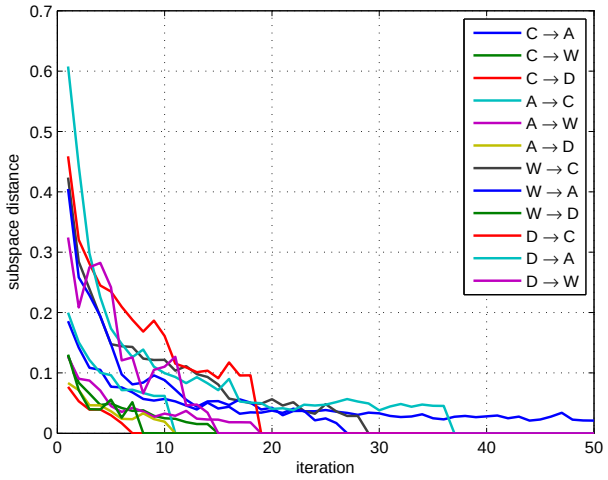

Figure 2.4: For each of 12 sub-problems, distance between 2 subspaces in succesive iterations being decreased through out the iterations. Proposed algorithm reaches *convergence* when subspace distance is negligible.

Table 2.2: Comparative results in terms of classification accuracy(%) of target data for 12 different sub-problems. Each sub-problem is in the form of *source* → *target*, where C(Caltech-256), A(Amazon), W(Webcam) and D(DSLR) indicate four different domains.

Methods	C → A	C → W	C → D	A → C	A → W	A → D	W → C	W → A	W → D	D → C	D → A	D → W	Avg
Origfeat	23.70	25.76	25.48	26.00	29.83	25.48	19.86	22.96	59.24	26.27	28.5	63.39	31.37
PCA	36.95	32.54	38.22	34.73	35.59	27.39	26.36	29.35	77.07	29.65	32.05	75.63	39.65
GFK	41.02	40.68	38.85	40.25	38.98	36.31	30.72	29.75	80.89	30.28	32.05	75.59	42.95
SA	42.07	32.2	45.86	39.8	37.63	36.94	28.76	34.34	88.54	32.5	34.24	88.47	45.11
TCA	45.82	30.51	35.67	40.07	35.25	34.39	29.92	28.81	85.99	32.06	31.42	86.44	43.03
TFL	44.78	41.69	45.22	39.36	37.97	39.49	**31.17**	32.78	89.17	31.52	33.09	**89.49**	46.31
TJM	46.76	38.98	44.59	39.45	42.03	**45.22**	30.19	29.96	**89.17**	31.43	32.78	85.42	46.33
QMI-H	55.95	49.49	45.86	**42.12**	42.71	37.58	30.37	35.8	80.89	35.71	38.31	61.02	46.32
QMI-S	**57.72**	**55.93**	**48.41**	41.76	**46.44**	38.85	30.72	**36.74**	83.44	**38.38**	**42.48**	77.63	**49.88**

For each of the 12 sub-problems with source-target combination (C→A, C→W etc.), the classification accuracy(%) of target domain data using our proposed iterative framework (Table 2.2) is reported. Iterative QMI-S shows improved performance compared to others and outperforms other methods in 8 out of 12 sub-problems. Also in terms of average accuracy over all 12 cases, our method is 3.61% ahead of the closest average accuracy obtained by **TJM**. Also ours approach outperforms **GFK** and **SA** by a large margin that have been considered state-of-the-art approaches so far. We believe, the enhanced performance comes from this fact that our learned

feature space exhibits better class separation along with reducing the distribution difference between source and target domains. Among these methods, only **TFL** utilized conditional label distribution of data, but they updated target data iteratively with hard labeling, which seems to be an aggressive approach compared to our soft-labeling solution. Nevertheless, applying Gaussian kernel with large kernel size (large σ) helped to capture the underlying shared structure of similar data. We used σ as median of the pair-wise distances of data in raw feature space. Small σ caused degraded performance as the kernel matrix might not capture accurate similarity across domains. We have also tried with much higher values of σ (e.g. median$\times 10^2$, median$\times 10^3$ etc.) and found little impact on learning subspace, which was verified by negligible change in classification performance with change of σ.

Similarity embedding on the projected space: Figure 2.5 shows the similarity embedding in the learned feature space for three different methods including ours. For better illustration, we took 1051 projected data samples from 5 different classes among which 584 data are from source domain with their true labels and 467 data from target domain with their predicted labels. Then we formed the similarity matrix $\mathcal{S}$ with 25-nearest neighbors in the projected space (see Figure 2.5). $\mathcal{S} \in \mathbb{R}^{n \times n}$ is constructed as follows,

$$\mathcal{S}(i,j) = \begin{cases} 1 & \text{if } \boldsymbol{x}_p(i) \in L_k(\boldsymbol{x}_p(j)) \wedge \boldsymbol{x}_p(j) \in L_k(\boldsymbol{x}_p(i)) \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

Here $L_k(\boldsymbol{x})$ denotes the set of K nearest neighbors of $\boldsymbol{x}$. The top-left and bottom-right sub-matrices of each sub-figure represent within-class similarity inside a domain (source or target). It is noticed that QMI-S method exhibits more compact block diagonal structure (Figure 2.5(c)) compared to **TJM** (Figure 2.5(b)) that has the closest accuracy with ours in this sub-problem. On the contrary, the top-right and bottom-left sub-matrices represent within-class similarity across domains. Here, also

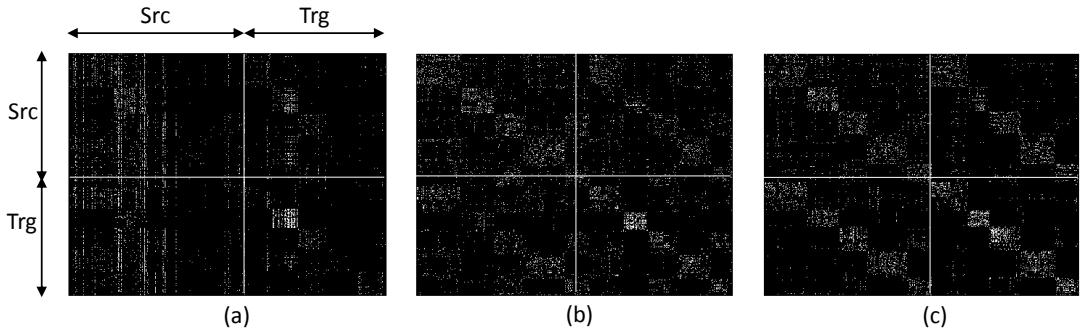

Figure 2.5: Assesing the quality of feature subspace by constructing similarity matrix using projected source and target domain data (from 5 different classes) for sub-problem $C \rightarrow A$ using (a) original feature space, (b) TJM and (c) iterative QMI-S.

Table 2.3: Classification accuracy(%) of target domain data for 12 different sub-problems using SVM classifier (QMI-S [$\mathcal{B}$]) and K-NN classifier (QMI-S [$\mathcal{A}$]), both trained using projected source data. Each sub-problem is in the form of *source* $\rightarrow$ *target*, where C(Caltech-256), A(Amazon), W(Webcam) and D(DSLR) indicate four different domains.

Methods	$C \rightarrow A$	C $\rightarrow$ W	C $\rightarrow$ D	A $\rightarrow$ C	A $\rightarrow$ W	A $\rightarrow$ D	W $\rightarrow$ C	W $\rightarrow$ A	W $\rightarrow$ D	D $\rightarrow$ C	D $\rightarrow$ A	D $\rightarrow$ W	Avg
QMI-S [$\mathcal{A}$]	57.72	55.93	**48.41**	41.76	**46.44**	38.85	30.72	**36.74**	**83.44**	**38.38**	**42.48**	**77.63**	49.88
QMI-S [$\mathcal{B}$]	**61.9**	**56.61**	**48.41**	**42.56**	45.76	**43.95**	**31.26**	36.53	**83.44**	38.29	41.96	76.61	**50.61**

we see better compact block diagonal structure generated by QMI-S method compared to **TJM**. It requires a careful look to realize that off-diagonal entries are more cluttered in **TJM** compared to our method. Figure 2.5(a) shows similarity matrix for **Origfeat** which involves no adaptation and hence evidently displays no pattern in the data similarity. The similarity embeddings for all 12 sub-problems are displayed in Figure 2.6.

2.2.2 Experiment $\mathcal{B}$

In this section, classification on the projected space is conducted using Support Vector Machine (SVM) classifier [72]. SVM has been proved to be a robust classifier in the area of classification or regression. It can also generate predicted labels along with a label distribution of each test datum. Therefore, SVM classifier can also be adopted to update label predictions inside the **repeat** loop of Algorithm 2 (line 11). In table 2.3, the classification accuracies for each of the 12 sub-problems have been provided. Using

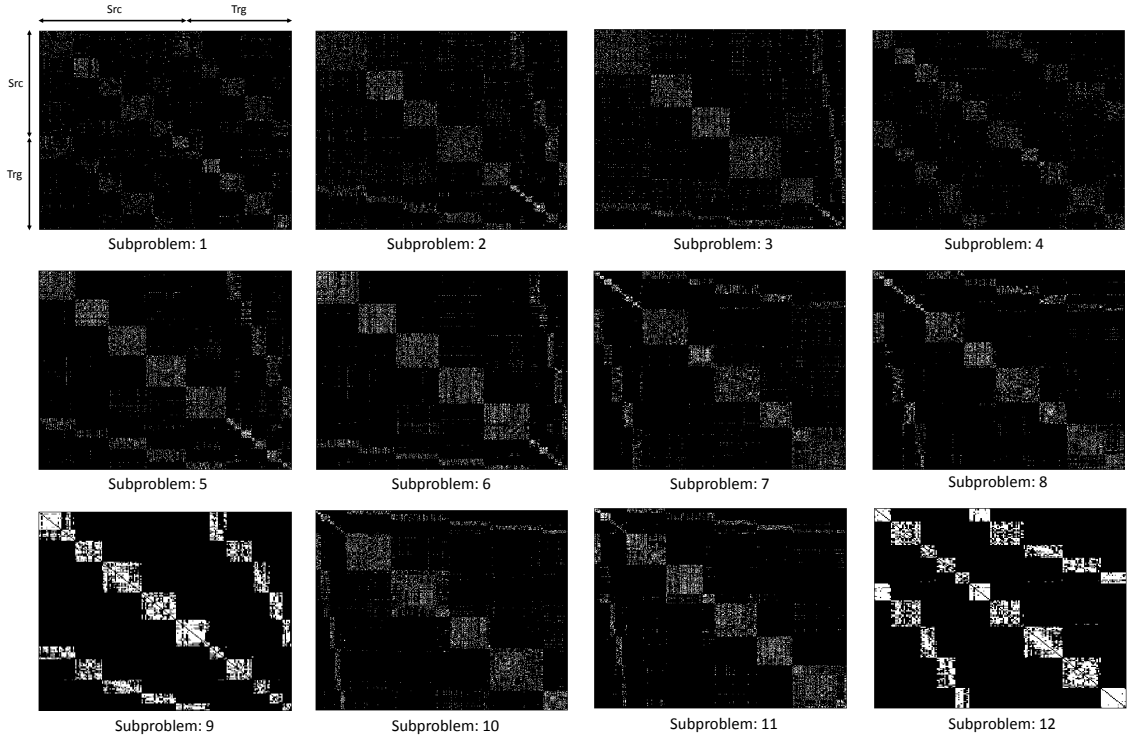

Figure 2.6: Similarity matrices $\mathcal{S}$ constructed with K nearest neighbors in learned subspaces for all 12 sub-problems. For each sub-problem, projected data (source and target) from 5 different classes are used to construct $\mathcal{S}$ according to Equation (2.15).

SVM classifier yields better classification accuracies compared to K-nn classifier. To train the SVM, linear kernel has been used. The same protocol is maintained for SVM training i.e. all source domain data are used for training and target domain data are used for testing. An efficient implementation of SVM is **libsvm** [73]. This software package is publicly available and used in this research work.

2.2.3 Experiment $\mathcal{C}$

Another investigation is conducted in the label prediction phase (Step-II) of Algorithm 2. It is worth noting that target data are predicted using neighboring source data in projected space. This entails a question that *what if a target domain datum is incorrectly classified?*. One safeguard to overcome this scenario is using *soft class labeling*. Instead of using hard label annotation for a target domain datum, a probability distribution of labels is used in prediction. Assigning target points with hard labels is an aggressive approach, whereas soft labeling will ensure a smooth update of the target predictions.

We experimented with the target prediction process by adding a uniform uncertainty to each prediction. This uncertainty is added initially to each prediction and gradually removed through out the iterations using a simulated annealing schedule [74]. An uniform uncertainty is defined as $U(c|\boldsymbol{x}) = \frac{1}{N_c}$ for each $c \in 1, 2, \ldots, N_c$. Therefore, each target prediction will adopt the following rule,

$$P_f(c|\boldsymbol{x}) = (1-\gamma)P(c|\boldsymbol{x}) + \gamma U(c|\boldsymbol{x}). \tag{2.16}$$

where $P(c|\boldsymbol{x})$ is the prediction using a classifier trained with projected source data (see Algorithm 2 line 11) and γ is a balancing co-efficient between classifier prediction and uniform uncertainty. $P_f(c|\boldsymbol{x})$ is the final prediction for each target point. Through out the iterations, γ will be faded away using a simulated annealing schedule. The intension behind this is, during the initial stage of the learning process, the

label predictions for the target data might be incorrectly biased and hence we induce a uniform uncertainty to each prediction in order to avoid misclassification of target data as much as possible. As the algorithm reaches towards *convergence*, the target predictions become more confident and correctly biased. Hence, we prefer to remove the uncertainty gradually from the prediction process (Equation (2.16)) by decreasing γ using a cooling schedule. A good number of simulated annealing schedules is available in the literature [74, 75]. The schedule used in this work is $\gamma = \gamma_0 exp(-bT)$, where γ_0 is the initial value of γ and T is the iteration count. In this experiment, $\gamma_0 = 0.5$ and $b = 0.4$ have been used. This value of γ_0 ensures an equal contribution of the two terms involved in the target prediction (Equation (2.16)). The coefficient b will control the decay speed of γ. This annealing of γ will follow a smooth decay as shown in Figure 2.7.

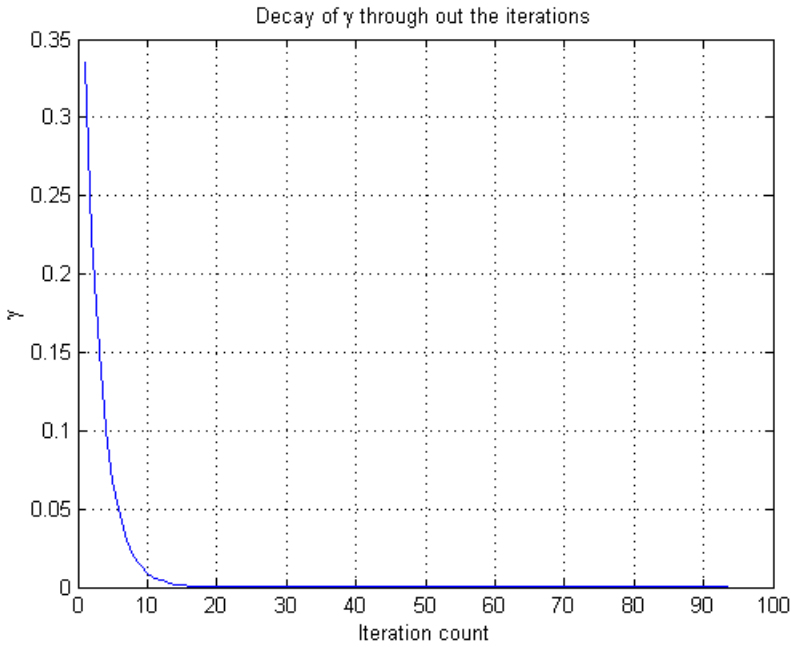

Figure 2.7: Simulated annealing schedule for γ with the increase of iteration count.

The uncertainty is removed gradually through out the iterations. Inducing a constant uncertainty (e.g. $\gamma = 0.5$) at every iteration may affect the discriminative nature

Table 2.4: Classification accuracy(%) of target data for 12 different sub-problems using K-nn classifier with K=1. Second row (QMI-S[$\mathcal{A} + du$]) represents accuracies where target predictions are obtained by adding a uniform uncertainty which is diminished through out the iterations (decayed uniform uncertainty). Third row (QMI-S[$\mathcal{A} + cu$]) represents accuracies where target predictions are obtained with constant uniform uncertainty ($\gamma = 0.5$).

Methods	$C \rightarrow A$	C → W	C → D	A → C	A → W	A → D	W → C	W → A	W → D	D → C	D → A	D → W	Avg
QMI-S [$\mathcal{A}$]	**57.72**	**55.93**	48.41	**41.76**	46.44	38.85	30.72	**36.74**	**83.44**	**38.38**	**42.48**	**77.63**	**49.88**
QMI-S [$\mathcal{A} + du$]	57.1	54.92	**49.04**	41.23	**46.78**	**40.76**	**31.17**	34.86	**83.44**	38.29	**42.48**	75.93	**49.67**
QMI-S[$\mathcal{A} + cu$]	41.23	35.93	38.85	37.13	31.53	32.48	21.10	19.31	64.33	16.47	14.61	45.42	33.20

of target data. Also for a correctly classified data point, the addition of uniform noise will only lessen its contribution to maximize QMI-S in the following iterations. Our goal was to achieve a biased label distribution for each target point whereas adding constant uniform noise will always somewhat flatten the label distribution. This eventually affects the discriminative subspace learning in following iterations. The uniform uncertainty can be thought of as a *safeguard* against the effect of misclassified data in earlier stages of the learning process. In Table 2.4, a comparative analysis is shown on how the introduction of uniform uncertainty will behave in our framework.

According to the results shown in Table 2.4, it is evident that inducing uncertainty helps in some of the cases, although the overall average accuracy is comparable to $\mathcal{A}$. This implies the power of soft-class labeling into the proposed subspace learning technique. The similar performances in classification accuracy with and without using uncertainty indicates that soft-labeling can efficiently handle the impact of initial misclassified target data. Another interesting observation from this experiment is that constant uniform uncertainty might be detrimental in learning a discriminative common subspace. At each iteration, each target datum is predicted using $P_f(c|\boldsymbol{x}) = 0.5 * P(c|\boldsymbol{x}) + 0.5 * U(c|\boldsymbol{x})$ i.e. a constant uncertainty is added with the label distribution predicted by the source classifier. The average accuracy in this case for all 12 subproblems is only 33.20%.

2.2.4 Experiment $\mathcal{D}$

Inducing exponential smoothing in target predictions:

According to Algorithm 2, target data predictions at each iteration are memoryless i.e. the label predictions for the target data at i-th iteration are ignorant of predictions at $(i-1)$-th iteration. Therefore, this label update is independent from one iteration to the next. It would be interesting to see if we update target label distribution P^i of each point at iteration i using a linear combination of P_i and P^{i-1} (label prediction at $(i-1)$-th iteration). Therefore, the label prediction for each datum at i-th iteration would be,

$$\begin{aligned} P^i(c|\boldsymbol{x}) &= \rho P^i(c|\boldsymbol{x}) + (1-\rho)P^{i-1}(c|\boldsymbol{x}) \\ &= \rho P^i(c|\boldsymbol{x}) + (1-\rho)(\rho P^{i-1}(c|\boldsymbol{x}) + (1-\rho)P^{i-2}(c|\boldsymbol{x})) \\ &= \rho P^i(c|\boldsymbol{x}) + \rho(1-\rho)P^{i-1}(c|\boldsymbol{x}) + (1-\rho)^2 P^{i-2}(c|\boldsymbol{x}) + \dots. \end{aligned}$$

Each prediction will combine the prediction at its current iteration and the one at the previous iteration and so forth. Therefore, an exponential growth in label prediction will take place for each target datum. The parameter ρ will control the depth level of the previous predictions in predicting current prediction. Higher value of ρ indicates that we are enforcing more importance to current predictions rather than previous ones, as only a small number of previous predictions will be involved in predicting the current one. On other hand, smaller value of ρ will cause the prediction process to traverse deep down the path of previous predictions. This process can be referred to as exponential smoothing of target label predictions.

In Table 2.5, we report the average accuracy over all 12 sub-problems along with the average loop count for *convergence* for different ρ values. It is worth noting that in terms of classification accuracy, there exists little difference than the original proposed method ($\mathcal{A}$). We believe using soft-labeling based approach has caused the

Table 2.5: Average classification accuracy over all 12 sub-problems for different values of smoothing parameter (ρ)

ρ	Avg. accuracy (%)	Avg. loop count
0.1	47.66	30
0.2	49.35	33.5
0.3	49.34	41.5
0.4	49.55	64.17
0.5	49.79	49.08
0.6	49.83	62.83
0.7	49.93	29.5
0.8	49.79	46.58
0.9	49.6	55.08

target predictions and hence the subspace learning process smoothly optimize. The only difference from the original method is the *convergence* time. Average *convergence* time changes based on the ρ values selected.

In this chapter, we proposed a domain adaptation algorithm based on soft-labeling induced quadratic mutual information. Unlike other subspace alignment methods, our goal was to utilize class conditional distribution of source domain to learn a common subspace with better class separation such that a classifier trained with projected source data can be applied to annotate target data. In next chapter, we propose to incorporate instance weighting into this framework in order to facilitate subspace learning using data samples with shared underlying similarity across domains along with minimizing the impact of unrelated source samples.

CHAPTER 3

Subspace Learning Based on Mutual Information Induced With Soft-labeling and Instance Weighting

In previous chapter, a domain adaptation framework based on soft-labeling induced mutual information (QMI-S) is proposed. In the projected subspace, target domain data are predicted using labeled source data. This process involves all the source and target domain data to learn the common subspace. In this chapter, we will address this question, *Are all source data important for transfer learning?*. It is already discussed in Introduction chapter that there is a tendency of *negative transfer* in learning process. Not all the source data are relevant for transfer learning i.e. not all of them share similarity with target domain data. These source points are considered to be similarly distributed with target domain data. Hence we can introduce instance weighting that will assign weights based on their importance or relevance in learning a common subspace.

In the literature, very few works have been involved to deal with this issue [6, 8]. Most of these works focus on finding closely distributed source data. We argue that both source and target data should be weighted based on their mutual relevance on the subspace learning process. Our idea is to learn a common subspace by maximizing QMI-S with weighted data samples. Source data will be weighted based on their distribution on the projected space. Source data that are closely distributed with respect to target data can be considered as highly resembled with target domain, hence they will gain higher weights than other source data. A nearest neighbor based approach

is employed to find this source data distribution. Our proposed approach is iterative and the weights are dynamically updated through out the iterations. This will ensure that similarly distributed samples (across domains) will be projected closely on the learned subspace. Thus source data whose underlying structures are not similar with target domain data are down weighted eventually. On the other hand, target data will be predicted using their neighboring points (source or target) and they are weighted based on their *confidence* in class label distribution. The *confidence* is measured using the max value of a sample's posterior class probability. Target points with higher *confidence* will be weighted higher than other target points.

We introduced instance weighting into the formulation of QMI-S. In previous chapter, a definition for soft-labeling induced mutual information is proposed based on [53]. Now the previous work is extended by inducing instance weighting into QMI-S formulation, we refer to this approach as weighted QMI-S or WQMI-S.

3.1 Weighted Quadratic Mutual Information with Soft Labeling (WQMI-S)

The definition of mutual information (MI) is provided in previous chapter. To review it, MI is defined as follows,

$$\begin{aligned}\mathcal{I}(X,C) &= \sum_c \int_{\boldsymbol{x}} (p(\boldsymbol{x},c) - p(\boldsymbol{x})P(c))^2 d\boldsymbol{x} \\ &= \sum_c \int_{\boldsymbol{x}} p(\boldsymbol{x},c)^2 d\boldsymbol{x} + \sum_c \int_{\boldsymbol{x}} p(\boldsymbol{x})^2 P(c)^2 d\boldsymbol{x} - 2\sum_c \int_{\boldsymbol{x}} p(\boldsymbol{x},c)p(\boldsymbol{x})P(c) d\boldsymbol{x}\end{aligned} \tag{3.1}$$

The probability distributions used in $\mathcal{I}(X,C)$ can be estimated by Parzen window method with a Gaussian kernel. $P(c)$, $p(\boldsymbol{x},c)$ and $p(\boldsymbol{x})$ are derived with Parzen window based density estimator [60] for weighted instances. In last chapter, QMI-S with uniform weighting or no weighting of instances is discussed. In this chapter, an ex-

pression involving non-uniform weighting of instances is formulated. This is a general formulation that can support both weighted and unweighted instances.

Finding $p(\boldsymbol{x})$: A Parzen window density estimation of $p(\boldsymbol{x})$ using IID drawn samples $\boldsymbol{x}_i$ is,

$$p(\boldsymbol{x}) = \sum_{i=1}^{n} P(\boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right)$$

where n is the cardinality of data set. With non-uniform weighting, $P(\boldsymbol{x}_i) = w_i$. Here w_i denotes individual weight of each data point. Hence, $p(\boldsymbol{x})$ becomes,

$$p(\boldsymbol{x}) = \sum_{i=1}^{n} w_i\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right)$$

Therefore WQMI is a generalized representation of MI between two random variables. The expression for UQMI can be obtained by choosing $w_i = \frac{1}{n}$ for all $\boldsymbol{x}_i$.

Finding $P(c)$: The prior class probability for class $c \in \{1, 2, \ldots, N_c\}$ using non-uniform weighting is,

$$P(c) = \sum_{i=1}^{n} P(c \mid \boldsymbol{x}_i)w_i \tag{3.2}$$

With hard class labeling, $P(c) = \sum_{i:P(c|\boldsymbol{x}_i)=1} w_i$, whereas using soft labeling (probability of a datum being classified as class c), $P(c) = \sum_{i=1}^{n} P(c|\boldsymbol{x}_i)w_i$. Hence, in case of non-uniform weighting and soft class labeling, $P(c)$ represents the sum of weighted class prior probabilities of class c, we will denote this as S_c for future use,

$$P(c) = \sum_{i=1}^{n} P(c|\boldsymbol{x}_i)w_i = S_c$$

Finding $p(\boldsymbol{x}, c)$: Parzen window estimate of the class data distribution $p(\boldsymbol{x}|c)$ is,

$$\begin{aligned} p(\boldsymbol{x} \mid c) &= \frac{1}{P(c)} \sum_{i=1}^{n} P(c, \boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right) \\ &= \frac{1}{P(c)} \sum_{i=1}^{n} P(c \mid \boldsymbol{x}_i)P(\boldsymbol{x}_i)\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right) \end{aligned}$$

With non-uniform weighting of samples and soft class labeling, $p(\boldsymbol{x}|c)$ takes the following form,

$$p(\boldsymbol{x}|c) = \frac{1}{S_c} \sum_{i}^{n} P(c|\boldsymbol{x}_i) w_i \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right)$$

Finally, the estimate of the joint pdf of $\boldsymbol{x}_i$ and c using non-uniform weighting of samples and soft class labeling will be,

$$\begin{aligned} p(\boldsymbol{x}, c) &= P(c) p(\boldsymbol{x} \mid c) \\ &= \sum_{i=1}^{n} P(c|\boldsymbol{x}_i) w_i \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right) \\ &= \sum_{i=1}^{n} z_{c,i} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\right) \end{aligned}$$

where $z_{c,i} = P(c|\boldsymbol{x}_i) w_i$. As discussed in previous chapter, a centralized Gaussian kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ is defined as $\boldsymbol{K} = \tilde{\boldsymbol{K}} - \boldsymbol{E}_n \tilde{\boldsymbol{K}} - \tilde{\boldsymbol{K}} \boldsymbol{E}_n + \boldsymbol{E}_n \tilde{\boldsymbol{K}} \boldsymbol{E}_n$, where $\tilde{\boldsymbol{K}}_{i,j} = \mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j ; \boldsymbol{0}, 2\sigma^2 \boldsymbol{I}\right)$ and $\boldsymbol{E}_n$ is an $n \times n$ matrix with all elements equal to $1/n$. Let $\boldsymbol{\Phi} \in \mathbb{R}^{n \times m}$ represents the projected data points from raw feature space to a kernel Hilbert space using the mapping function $\psi : \mathcal{X} \to \mathcal{H}$, where m is the dimension of kernel space (which is usually equals to infinity). Therefore, $\boldsymbol{K} = \boldsymbol{\Phi} \boldsymbol{\Phi}^{\mathrm{T}}$. Following [53], we can define $\mathcal{V}_{in}$, $\mathcal{V}_{all}$ and $\mathcal{V}_{btw}$ of Equation (2.9) as follows,

$$\begin{aligned} \mathcal{V}_{in} &= \sum_{c} \int_{\boldsymbol{x}} p(\boldsymbol{x}, c)^2 d\boldsymbol{x} \\ &= \sum_{c} \sum_{i=1}^{n} \sum_{j=1}^{n} z_{c,i} z_{c,j} \mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j ; 0, 2\sigma^2 \boldsymbol{I}\right) \\ &= \sum_{c} \boldsymbol{z}_c{}^T \boldsymbol{K} \boldsymbol{z}_c \\ &= \sum_{c} \mathrm{tr}\left\{ \boldsymbol{K} \boldsymbol{z}_c \boldsymbol{z}_c{}^T \right\} \\ &= \mathrm{tr}\left\{ \boldsymbol{\Phi} \boldsymbol{\Phi}^{\mathrm{T}} \sum_{c} \boldsymbol{z}_c \boldsymbol{z}_c^T \right\} \end{aligned}$$

where $\boldsymbol{z}_c = [z_{c,1}, z_{c,2}, \ldots, z_{c,n}]^T \in \mathbb{R}^{n \times 1}$.

$$\mathcal{V}_{all} = \sum_{c} \int_{\boldsymbol{x}} p(\boldsymbol{x})^2 P(c)^2 d\boldsymbol{x}$$

$$
\begin{aligned}
&= \sum_c S_c^2 \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j ; 0, 2\sigma^2 \boldsymbol{I}\right) \\
&= \left(\sum_c S_c^2\right) \sum_{i=1}^{n} \sum_{j=1}^{j=1} w_i w_j \boldsymbol{K}_{i,j} \\
&= \left(\sum_c S_c^2\right) \mathrm{tr}\{\boldsymbol{K}\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\} \\
&= \left(\sum_c S_c^2\right) \mathrm{tr}\{\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\} \\
&= \left(\sum_c S_c^2\right) \mathrm{tr}\{\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Phi}\}
\end{aligned}
$$

where $\boldsymbol{w}= [w_1, w_2, \ldots, w_n]^T \in \mathbb{R}^{n\times 1}$.

$$
\begin{aligned}
\mathcal{V}_{btw} &= \sum_c \int_{\boldsymbol{x}} p(\boldsymbol{x}, c) p(\boldsymbol{x}) P(c) d\boldsymbol{x} \\
&= \sum_c S_c \sum_{i}^{n} \sum_{j}^{n} z_{c,i} w_j \mathcal{N}\left(\boldsymbol{x}_i - \boldsymbol{x}_j ; \boldsymbol{0}, 2\sigma^2 \boldsymbol{I}\right) \\
&= \sum_c S_c \sum_{i}^{n} \sum_{j}^{n} z_{c,i} w_j \boldsymbol{K}_{i,j} \\
&= \sum_c s_c \, \boldsymbol{z}_c{}^T \boldsymbol{K}\boldsymbol{w} \\
&= \sum_c \mathrm{tr}\{S_c \boldsymbol{K}\boldsymbol{w}\boldsymbol{z}_c^T\} \\
&= \mathrm{tr}\left\{\sum_c S_c \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{w}\boldsymbol{z}_c^T\right\} \\
&= \mathrm{tr}\left\{\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{w}\sum_c S_c \boldsymbol{z}_c^T\right\} \\
&= \mathrm{tr}\left\{\boldsymbol{\Phi}^{\mathrm{T}}\left(\boldsymbol{w}\sum_c S_c \boldsymbol{z}_c^T\right)\boldsymbol{\Phi}\right\}
\end{aligned}
$$

Substituting the expressions for $\mathcal{V}_{in}$, $\mathcal{V}_{all}$ and $\mathcal{V}_{btw}$ into Equation (2.9), the closed form expression of WQMI-S takes the following form,

$$
\begin{aligned}
\mathcal{I}(X, C) &= \mathrm{tr}\left\{\boldsymbol{\Phi}^{\mathrm{T}}\left(\sum_c \boldsymbol{z}_c \boldsymbol{z}_c^T\right)\boldsymbol{\Phi}\right\} + \left(\sum_c S_c^2\right)\mathrm{tr}\{\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Phi}\} - 2\mathrm{tr}\left\{\boldsymbol{\Phi}^{\mathrm{T}}\left(\boldsymbol{w}\sum_c S_c \boldsymbol{z}_c^T\right)\boldsymbol{\Phi}\right\} \\
&= \mathrm{tr}\left\{\boldsymbol{\Phi}^{\mathrm{T}}\left(\sum_c \boldsymbol{z}_c \boldsymbol{z}_c^T + \left(\sum_c S_c^2\right)\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}} - 2\boldsymbol{w}\sum_c S_c \boldsymbol{z}_c^T\right)\boldsymbol{\Phi}\right\}
\end{aligned}
$$

$$= \mathrm{tr}\{\boldsymbol{\Phi}^{\mathrm{T}}\, \boldsymbol{M}\boldsymbol{\Phi}\} \tag{3.3}$$

where $\boldsymbol{M}=(\sum_c \boldsymbol{z}_c\, \boldsymbol{z}_c^T + (\sum_c S_c^2)\, \boldsymbol{w}\boldsymbol{w}^{\mathrm{T}} - 2\boldsymbol{w}\sum_c S_c \boldsymbol{z}_c^T)$ is the design matrix.

3.1.1 Subspace Learning By Maximizing WQMI-S

A low-dimensional subspace will be learned by maximizing WQMI-S. This subspace learning procedure is similar to one discussed in previous chapter. To recap, the objective function maximizing WQMI-S for learning a common subspace will be,

$$\boldsymbol{A}^* = \arg \max_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{A}=\boldsymbol{I}} \mathrm{tr}\{\boldsymbol{A}^{\mathrm{T}}\, \boldsymbol{K}\boldsymbol{M}\boldsymbol{K}\boldsymbol{A}\} \tag{3.4}$$

Here $\boldsymbol{A}^*$ can be obtained by applying Algorithm 1. Once we have the learned subspace, the projected data are computed as, $\boldsymbol{X}_p = \boldsymbol{\Phi}\boldsymbol{W} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\, \boldsymbol{A} = \boldsymbol{K}\boldsymbol{A}^*$.

3.1.2 Iterative update of instance weighting and soft-label prediction

As stated earlier, input data consists of both source and target samples which will be utilized for finding the desired subspace. The intension is to align these two distributions such that a classifier trained on source samples can also be applied in target domain. Therefore, our proposed framework consists of three main stages. They will be discussed elaborately in following paragraphs.

Like the iterative QMI-S based approach, source samples are initialized with corresponding class labels and target samples are initialized with uniform uncertainty of class labels. In other words, each target sample is assigned a discrete uniform distribution of class labels i.e for $\boldsymbol{x}_i \in \boldsymbol{X}_t$, the label distribution will be,

$$p(c_j|\boldsymbol{x}_i) = \frac{1}{N_c}, \text{ where } c_j \in \{1, 2, \ldots N_c\}.$$

On the contrary, source samples are annotated with corresponding ground truth labels, say they are represented as a label vector, $\boldsymbol{Y}_s = [y_1, y_2, \ldots, y_{n_s}]^T$. To be con-

sistent with the target samples, each $\boldsymbol{x_i} \in \boldsymbol{X}_s$ can also be associated with a label distribution using the following expression,

$$p(c_j|\boldsymbol{x}_i) = \begin{cases} 1 & \text{if } c_j = y_i \\ 0 & \text{otherwise} \end{cases}$$

Another prerequisite of the proposed method is the initialization of instance weighting. All the sample weights are represented as a weight vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]^T$ where $\sum_{i=1}^{n} w_i = 1$. The effects of instance weighting in DA framework are twofold,

Firstly, Not all the source data are useful for transfer learning i.e. some source samples might be *out-of-distribution* with the target data i.e. they share very little structural similarity with target domain data. By down-weighting those points, they are enforced to be less effective in the subspace learning process. Identifying those samples is a challenging task. It is reasonable to assume that those source samples will be mapped to distant places in an ideal common subspace with respect to target samples. The subspace is learned via an iterative fashion and all source samples contribute equally towards the learning process in the first iteration. While the algorithm proceeds towards *convergence*, *out-of-distribution* source samples are identified in the projected space and hence down-weighted through out the iterations. Figure 3.1 shows the final subspace learned using $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$ where some source points are distantly mapped with respect to target ones because of their underlying structure dissimilarity.

Secondly, Target data are initialized with zero weights as their labels are unknown resulting in high label uncertainty. Through out the iterations, target samples will obtain weights based on their *confidence* in class predictions. Although target samples are weighted gradually through out the iterations, this weight assignment is not uniform i.e. a target sample having higher *confidence* in class prediction will achieve higher weight than the one with lower *confidence*.

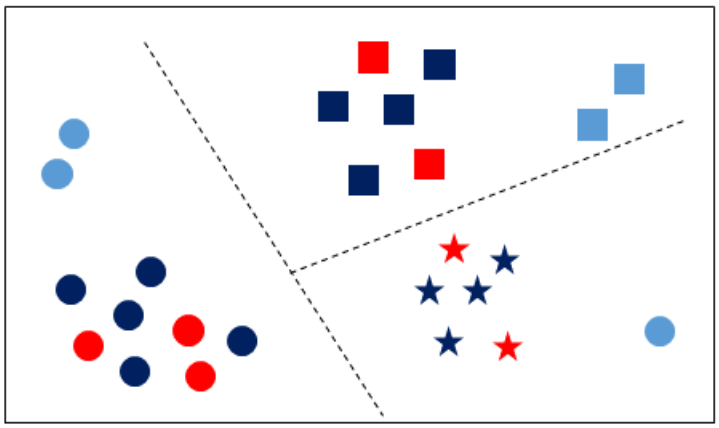

Figure 3.1: Final subspace learned from $\boldsymbol{X}_s$ (blue shapes) and $\boldsymbol{X}_t$ (red shapes). Circle, Rectangle and Star shapes represent three different object classes. *Out-of-distribution* source samples(light blue color) are distanlty located in the projected subspace.

This will ensure target samples having shared structural similarity with source samples to gain higher weights than those having less shared similarity.

Now a brief description for each stage of our proposed iterative framework shown in Figure 3.4 is provided as follows,

Step A: After initialization of data with corresponding label distributions and weights, this step involves in learning a low-dimensional subspace using a linear transformation. This transformation is designed to maximize quadratic mutual information QMI between data and corresponding class labels in the projected space. The optimization objective is provided in Equation 2.14. This subspace is different from the one discussed in previous chapter, as it involves both soft class assignment and weighted instances. In this case, the contribution of higher weighted instances in subspace learning will be more than the lower weighted ones, which is our expectation, as according to the hypothesis, not all source or target data might be equally useful to learn the common subspace.

Step B: Once the set of projected source and target samples is obtained, the target label predictions need to be updated. Initially, the label prediction for each target

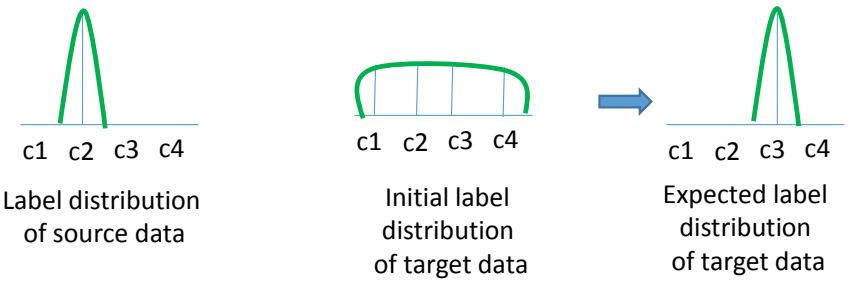

Figure 3.2: Class label distribution (with four classes) for each source and target domain datum.

sample is set to a uniform distribution of labels i.e. each target datum is assigned a uniform probability distribution of class labels. Through out the iterations, this uniform distribution is expected to be biased towards its true label (see Figure 3.2). Therefore, at each iteration, the label distributions are updated either using,

i. a classifier trained with projected source data and applied to projected target data to update their label distributions or

ii. a K nearest neighbor approach is applied to update the label prediction of each target sample (see Figure 3.3).

In the second approach, a target point is predicted by its neighboring source and target samples. The intuition is that projected source samples will influence nearby target points with their true labels. These target points (along with source ones) then eventually influence other unannotated nearby target points in predicting label distributions. This process will continue iteratively causing label information propagate through highly *confident* data points to lower ones. This scheme is somewhat similar to label propagation scheme [76] which is widely used in semi-supervised learning.

Step C: Apply a weighing scheme This is the last step of an iteration loop. This step plays a vital role in the subspace learning process. Through out the iterations, we will maintain this invariant, $\sum_{i=1}^{n} w_i = 1$. Also in the initialization step, equal weights are assigned to all source samples and zero weights to all target ones

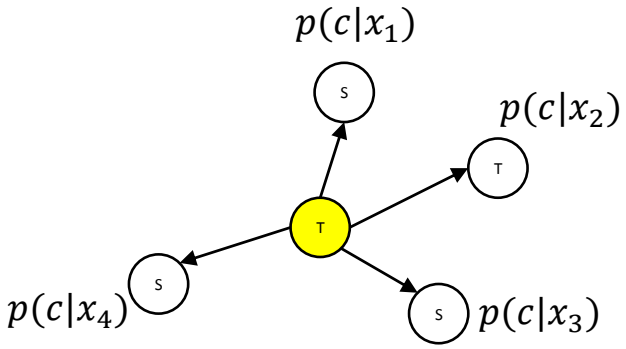

Figure 3.3: Weighted K nearest neighbor approach to update a label prediction of a target sample (marked as yellow). 'S'and 'T'represent source and target samples respectively. The updated label prediction will be $p(c|\boldsymbol{x}) = \sum_{i=1}^{4} w_i p(c|\boldsymbol{x}_i)$, where w_i is the corresponding weight of each neighboring sample.

such that $\sum_{i=1}^{n_s} w_i = 1$ and $\sum_{i=1}^{n_t} w_i = 0$. With this assignment, the initial WQMI-S subspace is dominated by source data and the learned projection matrix ($\boldsymbol{A}^*$) maps target data onto this subspace. In each following iteration, this process is repeated with updated instance weights resulting in a common domain adaptive WQMI-S subspace. Instance weighting is a generic approach and any appropriate weighting scheme can be employed based on the task at hand. One great advantage of our WQMI-S based subspace learning is that it can adapt a variety of weighting schemes. We will discuss 2 different weighting schemes in domain adaptation context. Generally, the motivation behind using a instance weighting approach is,

(i) To locate relevant samples across domain to assign them higher importance than others in learning a common feature space.

(ii) To overcome the imbalance nature of the dataset, e.g. imbalance in source and target domain data size, imbalance in class data distribution etc.

The overall framework is summarized in Figure 3.4. The weight vector $\boldsymbol{w}$ is induced into the original QMI formulation of [53]. If all the entries of $\boldsymbol{w}$ is equal i.e. all instances are assigned constant equal weights, then WQMI-S boils down to QMI-S approach discussed in previous chapter.

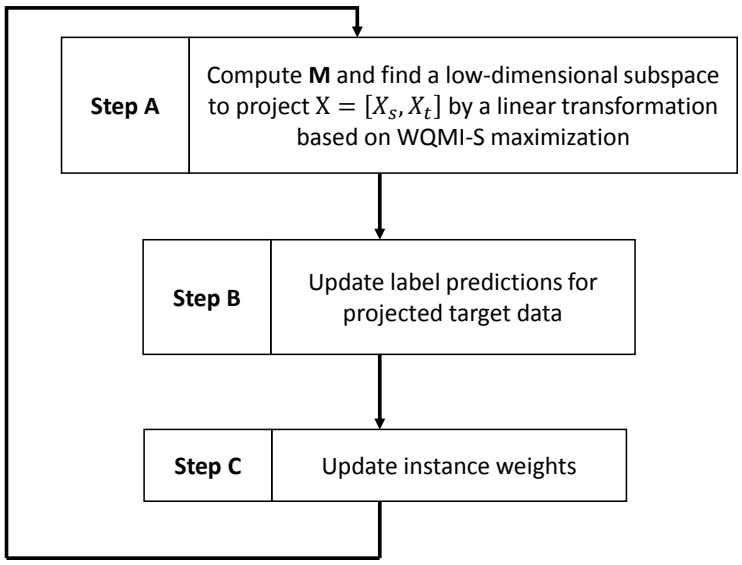

Figure 3.4: 3-step iterative approach of the proposed framework based on maximization of WQMI-S.

3.2 Proposed Weighting Scheme: weight transfer approach

We propose a scheme where instance weights will propagate from source domain towards target domain. Like label propagation in semi-supervised learning framework, source data weights along with label information will be propagated towards unlabeled target domain data. In DA setting, only source data are labeled and target data are unlabeled initially. Therefore, total weight mass distribution will be centered to source data and hence target data will be assigned negligible amount of weight ($\approxeq 0$) initially. The total weight of all samples is constant and re-distributed among relevant samples across domains in an iterative fashion. The goal is to conduct the weight re-adjustment at each iteration such that,

1. *Out-of-distribution* source samples are down-weighted so that only relevant source samples can contribute in the subspace learning process.

2. Each target sample can gain weight based on its label uncertainty i.e. higher confident (in terms of label prediction) target points will be assigned with higher weights than less confident ones.

With this approach, the distribution divergence between source and target domains is eventually minimized in the projected subspace as all target data and only relevant source data contribute effectively in the transformation. To do this, a fraction of weights from the total weight mass is subtracted and re-distributed among a *candidate* list of samples at each iteration. This causes the total weight mass gradually shifted towards target domain and the process of weight re-adjustment stops when change of instance weights in successive iterations is negligible. This can also be thought of an energy re-distribution scheme where at each iteration, a certain amount of energy is taken away from a system and then re-distributed among the *candidate* points of that system. Similarly in DA setting, at each iteration, $\alpha\%$ amount is deducted from total weight and then re-distributed among *candidate* points. After some iterations, a *convergence* state is reached where the target weights are stabilized. We define a *candidate* set, Ω consisting of source and target samples. All the target samples are included in the *candidate* list. Source samples that are among the K nearest neighbors of any target point, are included in Ω. Denoting $L_k(\boldsymbol{x})$ as the set of K nearest neighbors of a data point $\boldsymbol{x}$, we can define Ω as follows,

$$\Omega = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{X}_t \cup (\boldsymbol{x} \in \boldsymbol{X}_s \cap \boldsymbol{x} \in L_k(\boldsymbol{x}_t), \exists \boldsymbol{x}_t \in \boldsymbol{X}_t)\}.$$

According to above rule for *candidate* list creation, a source sample that is in the neighboring region of any target point in the learned subspace will be considered as a *candidate* point (source *candidate*). The weight re-adjustment will take place among the *candidate* list members. The weighting process consists of two main steps,

1. **Weight shrinking:** A fraction of weight is deducted from each instance weight (both source and target instance).

2. **Weight re-distribution:** The subtracted weight is re-distributed among the *candidate* list members.

At each iteration, the above two steps are employed to adjust the weight mass distribution among prospective samples. In other words, the goal is to align the weight mass such that source samples having less similarity with target ones are subject to be down-weighted and thus they have minimal influence on the learned subspace. Say at a certain iteration, α is the fraction of weight that is subtracted from the data set and needs to be re-assigned among the members of Ω. This weight adjustment is non-uniform. As for example, target *candidate* sample with higher label uncertainty should get less weight than the one with lower uncertainty (higher *confidence*) in label prediction. For simplicity, this *confidence* is measured with a simple probabilistic metric, $u(\boldsymbol{x})$ defined as,

$$u(\boldsymbol{x}) = \max(P(c|\boldsymbol{x}))$$

$u(\boldsymbol{x})$ is utilized in weight re-assignment process among target data points. Higher $u(\boldsymbol{x})$ indicates lower uncertainty in class prediction and vice versa. Source points that are members of the *candidate* list are treated uniformly i.e. equal amount of weight is assigned to all source *candidate* points.

As stated earlier, the goal is to learn a low-dimensional subspace that represents the commonality among source and target data points. Also our assumption is labeled source data size is larger than unlabeled target one, this is intuitive in practical sense. Considering this assumption, a balance in weight re-adjustment is maintained among source and target domain. In other words, α amount of weight is shrinked which is then re-distributed among *candidate* list members such that a balance is maintained among source *candidate* and target *candidate* points. Target *candidate* points are further categorized on their label prediction uncertainty. In summary, the *candidate* list Ω consists of three subsets which are,

Ω_s : It contains source domain data that are considered as candidate i.e. a source point residing in the neighboring region of any target point in the projected

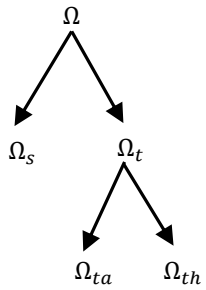

Figure 3.5: Formation of *candidate* list Ω.

space. $\Omega_s = \{(\boldsymbol{x}|\boldsymbol{x} \in \boldsymbol{X}_s) \cap (\boldsymbol{x} \in L_k(\boldsymbol{x}_i),\ \exists \boldsymbol{x}_i \in \boldsymbol{X}_t)\}$, where $L_k(\boldsymbol{x}_i)$ is the set of K nearest neighbors of $\boldsymbol{x}_i$.

Ω_{ta} : All target data points are included in this subset. $\Omega_{ta} = \{\boldsymbol{x}|\boldsymbol{x} \in \boldsymbol{X}_t\}$

Ω_{th} : Target data points having high *confidence* in class label predictions are included in this subset. $\Omega_{th} = \{\boldsymbol{x}|\boldsymbol{x} \in \boldsymbol{X}_t \cap u(\boldsymbol{x}) \geq \tau\}$, where τ is a *confidence* threshold.

Finally the formation of Ω will take the following form (see Figure 3.5),

$$\Omega = \{\Omega_s \cup \Omega_{ta} \cup \Omega_{th}\}$$

Now α amount of weight will be distributed among the members of Ω. Therefore, the equations to implement our weighting scheme will be,

$$\textbf{Shrinking weight: } w_i' = w_i - \alpha w_i, \text{for each } \boldsymbol{x}_i \in \boldsymbol{X}. \tag{3.5}$$

$$\textbf{Redistributing weight: } w_i^{new} = \begin{cases} w_i' + \frac{\eta\alpha}{|\Omega_s|}, & \text{if } \boldsymbol{x}_i \in \Omega_s. \\ w_i' + \frac{(1-\eta)\beta\alpha}{|\Omega_{th}|}, & \text{if } \boldsymbol{x}_i \in \Omega_{th}. \\ w_i' + \frac{(1-\eta)(1-\beta)\alpha}{|\Omega_{ta}|}, & \text{if } \boldsymbol{x}_i \in \Omega_{ta}. \end{cases} \tag{3.6}$$

Here η and β are balancing coefficients that control the distribution of weights among Ω_s and Ω_t and also among Ω_{ta} and Ω_{th} respectively. Each of these parameters are in the range from 0 to 1 i.e. $0 < \eta \leq 1$ and $0 < \beta \leq 1$. We will describe each of these parameters and their effect in the weight distribution. The above process of weight update takes place at each iteration.

Shrinking weight

According to Equation 3.5, a fraction α amount of weight is subtracted from each instance weight at each iteration. A weight invariant is maintained as $\sum_i w_i = 1$. As αw_i is extracted from each x_i, the weight allotted for re-distribution will be $\alpha \sum_i w_i = \alpha$.

Re-distributing weight

Now α amount of weight is adjusted among the members of Ω. First, α is allotted among Ω_s and Ω_t such that $\eta\alpha + (1-\eta)\alpha = \alpha$. The parameter η controls the partition of α. Concretely, η controls the fraction of weight to be transferred from source towards target domain.

The parameter β controls the weight re-assignment among the members of Ω_t. Target data points are weighted based on their *confidence* in class label prediction. The idea is to assign higher weights to highly *confident* (less uncertain) target points and assign less weights to others. $(1-\eta)\alpha$ amount of weights is allotted for distribution among members of Ω_t. This amount is further divided into two fractions such that $\beta(1-\eta)\alpha + (1-\beta)(1-\eta)\alpha = (1-\eta)\alpha$. The third line of Equation (3.6) indicates that a fraction of $(1-\eta)\alpha$ is distributed to all the target data points uniformly. The second line indicates that members of Ω_{th} gain a 'bonus'weight because of their high *confidence* in label predictions. Eventually these *confident* points can make effective contribution towards subspace learning and update of target data predictions for the following iteration.

The above approach is a generic weighting scheme for weighted QMI-S based subspace learning in domain adaptation setting. Any appropriate weighting scheme can be incorporated in our proposed framework. In this case, the parameters α,

η and β control the behavior of the weighting scheme. We propose an unsupervised parameter setup approach (referred to as Unsupervised Parameter Adaptation, UPA) in following discussions.

Parameter setting in weighting scheme for domain adaptation

The first parameter needed to be set is α which is defined as the fraction of weight shrinked from each instance weight. Therefore, the range of α is $0 < \alpha \leq 1.0$. Extracting αw_i from each sample weight w_i shrinks a total of α weight from all the data points. This α is later re-assigned among the *candidate* points to hold the weight invariant ($\sum_{i=1}^{n} w_i = 1$) true. As stated earlier, target points are initially zero weighted and are assigned with weights through out the iterations. Also a set of *confident* target points $\Omega_t h$ gain some 'bonus'weights. We argue that α is linearly proportional to the size of Ω_{th}, that is,

$$\alpha = \max(\upsilon \frac{|\Omega_{th}|}{|\Omega_{ta}|}, \zeta)$$

Here ζ is a constant of very small value (usually in the range of $\approxeq 10^{-3}$). This constant is used to avoid the scenario of $|\Omega_{th}| = 0$ causing infinite loop of the algorithm. In our implementation, $\upsilon = 0.5$ is used. The intuition behind the above formulation is that if Ω_{th} is large in any iteration, then a large number of target points have become highly *confident* and hence the necessity of weight allotment to target domain becomes high. In this situation, α should be large enough to assign target points with sufficient weights according to Equation (3.6). Concretely, If α is large (as Ω_{th} is large), then large amount of weight is shrinked in order to provide *confident* target points sufficient weights. This weight adjustment in target domain data is further described in details later.

Next concern is η and β. From Equation (3.6), it is evident that the shrinked weight α is re-adjusted among *candidate* points in two main steps utilizing these two

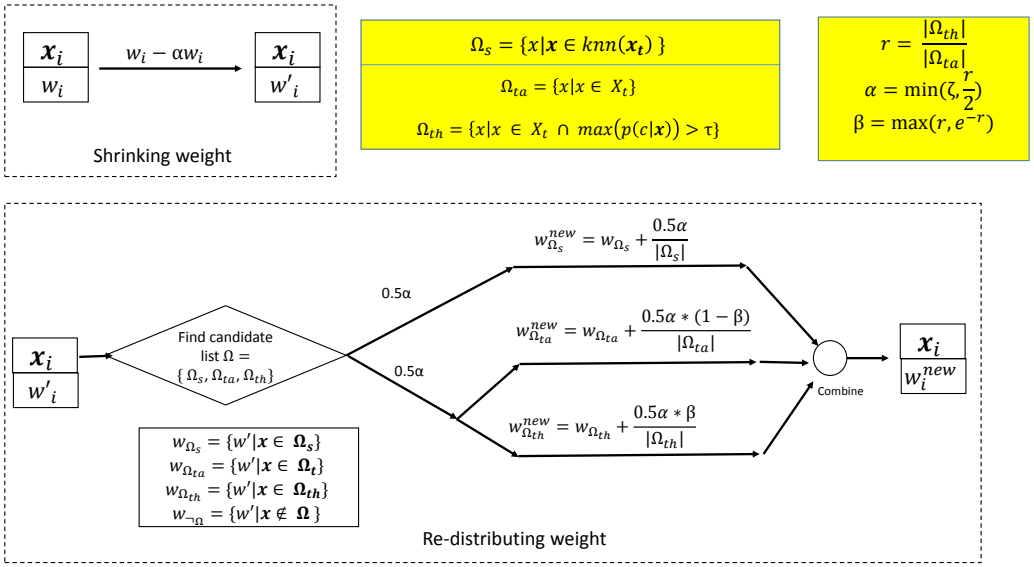

Figure 3.6: Proposed weighting scheme to be applied in iterative WQMI-S algorithm for domain adaptation.

parameters. They are described as follows. Also see Figure 3.6 for an overview of the proposed weighting scheme.

i. Weight re-adjustment among Ω_s and Ω_t: As the final goal is to learn a common feature subspace that minimizes the divergence between source and target domain, the α is distributed among Ω_s and Ω_t in an equal share, that is,$\eta = 0.5$. Such partition of α will protect transferring of excessive weights towards target domain as the idea is to construct the subspace with equal contribution from source and target domain. At each iteration of the iterative framework, weight is adjusted among source and target *candidate* points and this way, target points are weighted gradually through out the iterations. In summary, each source *candidate* point will be weighted according to following equation,

$$w^{new}_{\Omega_s} = w'_{\Omega_s} + \frac{0.5\alpha}{|\Omega_s|} \tag{3.7}$$

ii. Weight re-adjustment among target *candidate* points: As 0.5α is devoted for target data points, all of them are assigned weights from the allotted 0.5α based on their *confidence* in class label prediction which is measured by $u(\boldsymbol{x})$.

Two sub-steps are involved in this allotment process. In first sub-step, a fixed portion of 0.5α i.e. $0.5\alpha(1-\beta)$ amount of weight is distributed among all target points. The remaining $0.5\alpha\beta$ is alloted to points that are *confident* in their class label predictions. In other words, *confident* points will gain some 'bonus'weights. This is intuitive, as in practical scenario, source and target domain data do not share underlying common characteristics in a fairly equal proportion. From a narrow point of view, a single source point $\boldsymbol{x}_s$ may not possess exactly same structural property as that of any target point $\boldsymbol{x}_t$. Therefore, each data point will have its own unique contribution towards the subspace learning process. In summary, the weight adjustment of each target point will be as following,

$$w_{\Omega_{th}}^{new} = w'_{\Omega_{th}} + \frac{0.5\alpha\beta}{|\Omega_{th}|} \tag{3.8}$$

$$w_{\Omega_{ta}}^{new} = w'_{\Omega_{ta}} + \frac{0.5\alpha(1-\beta)}{|\Omega_{ta}|} \tag{3.9}$$

The above equations require the setup of Ω_{th} and β. Ω_{th} is constructed by choosing the threshold τ. A reasonable choice might be $\tau \geq 0.7$. In our research, we propose a unsupervised parameter setup approach (referred to as Unsupervised Parameter Adaptation) to fix this cutoff threshold. Concretely, target domain data are considered as *confident* in their class label predictions, when $u(\boldsymbol{x}) \geq \tau$ for $\boldsymbol{x} \in \boldsymbol{X}_t$.

Finally, we propose a decision rule to setup β which decides what fraction of 0.5α is allotted for 'bonus'. The rest of 0.5α i.e. $0.5(1-\beta)\alpha$ is re-distributed among all target points. Equal partition of 0.5α might be a simple choice i.e. $\beta = 0.5$. In this work, we propose the following rule for β,

$$r = \frac{\Omega_{th}}{|\Omega_{ta}|}$$

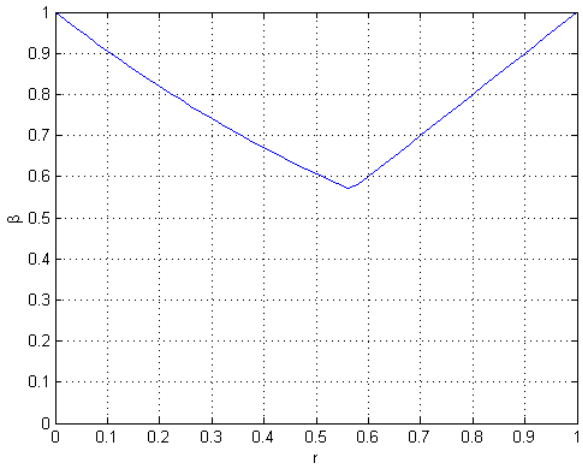

Figure 3.7: Plot of r vs. β.

$$\beta = max(r, exp(-r)).$$

It is noted that $min(\beta) = 0.57$ and $max(\beta) = 1.0$. The intuition is to set a minimum bound for 'bonus' fraction. Two possible extreme cases are also handled here: i) if r is small, most points are uncertain and hence they are assigned with very small weights and ii) if r is high, number of uncertain points will be low and in this case, these points should receive smaller weights. The behavior of r and β will follow the behavior plotted in Figure 3.7.

One potential question might be *why not the bonus weights always high?*. Based on our analysis we found that if we set the bonus weight fraction always high (say $\beta \geq 0.8$ or so), then false-positive points (i.e. points that are highly confident to incorrect class labels) will always be assigned with higher weights and this scenario might propagate through out the iterations resulting in erroneous weight adjustment which will affect $\boldsymbol{M}$ matrix formulation for the following iteration.

3.2.1 Unsupervised Parameter Adaptation (UPA)

In this section, an adaptation method to choose the appropriate value of the *confidence* threshold τ that is used to construct the set Ω_{th}. As our domain adaptation

framework is an iterative approach, appropriate choice of this parameter will eventually effect the following iterations. Instead of manually choosing τ, we propose an unsupervised parameter adaptation method (referred to as UPA). Each iteration conducts this adaptation process which involves learning multiple weighted QMI-S subspaces. In this approach, $\boldsymbol{w}$ is constructed according to Equation (3.5) and (3.6) for each possible value of τ to learn multiple WQMI-S subspaces. In each case, the remaining parameters are used according to above description. The idea is to select the *best* subspace learned using $\boldsymbol{w}$ with corresponding τ. We propose an intuitive idea to select among these subspaces. Assuming projected target data will be tightly clustered around their class means in an ideal subspace, K-means clustering is applied to them in each WQMI-S subspace, where K is set to the number of unique categories. From this clustering, a scatter metric $\mathcal{G} = \frac{J_b}{J_w}$ is computed with J_w and J_b representing within-cluster scatter and between-cluster scatter respectively [77]. The WQMI-S subspace with maximum $\mathcal{G}$ is chosen which will be eventually utilized in following iteration. The overall DA framework is summarized in Figure 3.8. $\mathcal{G}$ is defined as,

$$
\begin{aligned}
J_b &= \sum_k n_k |\boldsymbol{\mu_k} - \boldsymbol{\mu}| \\
J_w &= \sum_k \sum_{\boldsymbol{x} \in P_k} |\boldsymbol{\mu_k} - \boldsymbol{x}| \\
\mathcal{G} &= \frac{J_b}{J_w}
\end{aligned}
\tag{3.10}
$$

where n_k is the number of points belonging to cluster k, μ_k is the k-th cluster centroid and P_k is the partition of data points belonging to cluster k. The goal is to choose the subspace where the total scatter $\mathcal{G}$ is maximized for using corresponding τ.

As shown in Figure 3.8, at **Step-C**, for each possible value of τ, a unique weight vector is generated which is eventually used in constructing $\boldsymbol{M}$ matrix. Therefore, for each $\boldsymbol{M}$ matrix, a separate WQMI-S subspace can be learned and the scatter

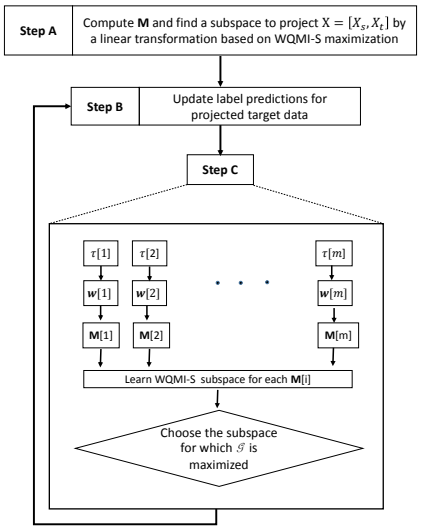

Figure 3.8: 3-step iterative approach of the proposed framework with intermediate UPA approach at each iteration. In Step C, for each possible value of τ (say $\tau[i]$), corresponding weight vector $\boldsymbol{w}[i]$ and $\boldsymbol{M}[i]$ are constructed to learn a subspace.

measure $\mathcal{G}$ is computed. The subspace is chosen based on the maximum $\mathcal{G}$ value. The detailed process of UPA has been summarized in Algorithm 3.

Algorithm 3 Unsupervised Parameter Adaptation (UPA)

1: Choose a set of possible values for τ e.g. $\tau = \{0.55, 0.6, 0.65, 0.7, 0.75\}$

2: **for** each τ **do**

3: Construct Ω_s as $\Omega_s = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{X}_s \cap (\boldsymbol{x} \in L_k(\boldsymbol{x}_t), \exists \boldsymbol{x}_t \in \boldsymbol{X}_t)\}$

4: Construct Ω_{th} as $\Omega_{th} = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{X}_t \cap u(\boldsymbol{x}) \geq \tau\}$

5: Construct Ω_{ta} as $\Omega_{ta} = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{X}_t\}$

6: Apply Equation (3.5) and (3.6) to construct (or update) the weight vector $\boldsymbol{w}$.

7: Compute $\boldsymbol{M}$ according to Equation (3.3) and make it symmetric as $\boldsymbol{M'} = \frac{\boldsymbol{M}+\boldsymbol{M}^{\mathrm{T}}}{2}$.

8: Solve standard eigen problem, $\boldsymbol{M'V} = \boldsymbol{V\Lambda}$.

9: Compute projected data $\boldsymbol{X}_p = \boldsymbol{K}^{\frac{1}{2}}\boldsymbol{V}$, where $\boldsymbol{K}$ is the centralized Gaussian kernel matrix.

10: Apply K-means clustering to the projected target domain data with K set to the number of unique class labels.

11: Measure total data scatter $\mathcal{G}$ using within-cluster scatter and between-cluster scatter (Equation (3.10)).

12: **end for**

13: Choose the WQMI-S subspace (alternatively, eigen matrix $\boldsymbol{V}$) for which $\mathcal{G}$ is maximized.

Therefore, our proposed domain adaptation framework showed in Figure 3.4 is revised with the inclusion of UPA module, revised framework is showed in Figure 3.8. The final overall algorithm is summarized in Algorithm 4.

3.2.2 *Convergence* criterion

Algorithm 4 will iteratively learn a subspace by maximization of quadratic mutual information with weighted instances. The *convergence* state is defined as the steady state when i) weight re-adjustment for target data in successive iterations is negligi-

ble or ii) $\sum_{i=1}^{n_s} w_i = \sum_{j=1}^{n_t} w_j$ with sum of weights for *non-candidate* source points is negligible. At this state, a stable low-dimensional subspace is found and the weight shifting from source domain data towards target domain reaches equilibrium. In the experimental evaluation, difference between target data weights in successive iterations is compared with a threshold. Representing current and previous weight for target points as two vectors $\boldsymbol{w}_i^{curr}$ and $\boldsymbol{w}_i^{old}$, the proposed algorithm will be converged or stopped when $|\boldsymbol{w}_i^{new} - \boldsymbol{w}_i^{new}| \leq \epsilon_w$, where ϵ_w is a small value to be chosen for the stopping criterion.

3.3 Weighting scheme: source-target imbalance

Another weighting scheme is provided here with a different perspective than the one described above. Usually source domain data size will be higher than the target domain data. Our goal was to learn a common subspace that represents the underlying common structure between source and target domain data. If the source domain data is much larger than target domain, then it can overwhelm and dominate over the target domain signal. In that case, the idea of common feature subspace is compromised. To overcome this issue, we propose to distribute total weight mass among source and target domain with a fare share. Initially all the samples will be weighted and through out the iterations this weight assignment will remain unchanged. Therefore, this weighting scheme will follow $\sum_{i=1}^{n_s} w_i = 0.5$ and $\sum_{i=1}^{n_t} w_i = 0.5$.

Each data point will be weighted according to the following equations,

$$w_i = \begin{cases} \frac{0.5}{n_s}, & \text{if } \boldsymbol{x}_i \in \boldsymbol{X}_s \,. \\ \frac{0.5}{n_t}, & \text{if } \boldsymbol{x}_i \in \boldsymbol{X}_t \,. \end{cases} \tag{3.11}$$

Algorithm 4 Iterative domaina adaptation algorithm based on subspace learning with maximization of WQMI-S.

1: **Input**: Data matrix $\boldsymbol{X}$=[$\boldsymbol{X}_s$,$\boldsymbol{X}_t$] where source domain data $\boldsymbol{X}_s \in \mathbb{R}^{n_s \times d}$ and target domain data $\boldsymbol{X}_t \in \mathbb{R}^{n_t \times d}$, label vector for source data $\boldsymbol{Y}_s \in \mathbb{R}^{n_s \times 1}$

2: **Output**: $\boldsymbol{X}_p \in \mathbb{R}^{n \times k}$, projected data with a linear transformation in k-dimensional subspace. Here $n = n_s + n_t$

Initialization:

3: E = 1; Weight vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]^T$

4: $w_i = \frac{E}{n_s}$, for each $\boldsymbol{x}_i \in \boldsymbol{X}_s$

5: $w_j = \frac{1-E}{n_t}$, for each $\boldsymbol{x}_j \in \boldsymbol{X}_t$

Initialize label distribution $P(c|\boldsymbol{x})$ as follows,

6: For each $\boldsymbol{x}_i \in \boldsymbol{X}_s$, $P(c_j|\boldsymbol{x}_i) = \frac{1}{N_c}$, where $c \in \{1, 2, \ldots N_c\}$

7: For each $\boldsymbol{x}_i \in \boldsymbol{X}_t$,

$$P(c_j|\boldsymbol{x}_i) = \begin{cases} 1 & \text{if } c_j = y_i \\ 0 & \text{otherwise} \end{cases}$$

8: Compute centralized kernel matrix, $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ computed as follows, $\boldsymbol{K} = \tilde{\boldsymbol{K}} - \boldsymbol{E}_n \tilde{\boldsymbol{K}} - \tilde{\boldsymbol{K}} \boldsymbol{E}_n + \boldsymbol{E}_n \tilde{\boldsymbol{K}} \boldsymbol{E}_n$, where $\tilde{\boldsymbol{K}}_{i,j} = \mathcal{N}(\boldsymbol{x}_i - \boldsymbol{x}_j ; \boldsymbol{0}, 2\sigma^2 \boldsymbol{I})$ and $\boldsymbol{E}_n$ is an $n \times n$ matrix with all elements equal to $1/n$.

Step-A:

9: Compute $\boldsymbol{M}$ according to Equation (3.3) and make it symmetric as $\boldsymbol{M'} = \frac{\boldsymbol{M}+\boldsymbol{M}^{\mathrm{T}}}{2}$.

10: Solve standard eigen problem, $\boldsymbol{M'V} = \boldsymbol{V\Lambda}$.

Step-B:

11: **repeat**

12: Compute projected data $\boldsymbol{X}_p = \boldsymbol{K}^{\frac{1}{2}} \boldsymbol{V}$.

for all $\boldsymbol{x}_i \in \boldsymbol{X}_t$ **do**

Find a set L_k consisting of K nearest neighbors of $\boldsymbol{x}_i$.

Update $P(c|\boldsymbol{x}_i)$ as $\sum_{\boldsymbol{x}_j \in L_k} w_j P(c|\boldsymbol{x}_j)$.

end for

Step-C:

Apply Algorithm 3 to choose an optimal WQMI-S subspace. Use the $\boldsymbol{V}$ matrix returned from Algorithm 3 for next iteration.

until *convergence*

3.4 Classification in target domain

Once the projected source and target data are obtained after terminating Algorithm 4, we can train a classifier with projected source data and corresponding class labels. This classifier is tested against the target domain data for classification. In the experiment section, K-nn classifier is used with $K = 1$.

3.5 Experiments

The same **Office** and **Caltech-256** dataset are used to test the weighted QMI-S framework. The four domains or datasets involved here are Amazon (A), DSLR (D), Webcam (W) and Caltech-256 (C). The experiments are conducted under the setting $n_s > n_t$ or at least there exists a balance in sizes between source and target domain. As before, all the labeled source domain data and unlabeled target domain data are used as input for the algorithm and the projected source data will be used as a training set to learn a classifier upon *convergence* of the algorithm. In Table 3.1, the classification accuracy for target domain data is reported using our proposed methods: QMI-S (DA method using soft-labeling induced maximization of mutual information), QMI-H (DA method with maximization of mutual information)

and WQMI-S (DA method with maximization of mutual information using instance weighted soft-labeling). In that table, a comparative result is also provided with other state-of-the-art approaches. For fair comparison, K-NN classifier is used with K=1 to classify target domain data for all the DA methods including ours.

From Table 3.1, it is noted that our WQMI-S algorithm surpasses other methods by a significant margin (on average 3.59% higher than the QMI-S approach and 7.35% higher than TJM approach which is the best reported result among other state-of-the-art approaches). The performance improves in 5 out of the 7 sub-problems from QMI-S approach by a large margin and very close in other two cases. This performance is a reflection of the contribution of both soft-class labeling and instance weighting. Iterative update of target prediction using soft-labeling generates a discriminative subspace. The soft-labeling smoothly updates the target data labeling towards a *confident* prediction upon *convergence* of the algorithm. On other hand, instance weighted soft-labeling scales the contribution of each sample towards the subspace learning process. We show that although source domain data is larger than target one, not all the source data have similar contribution in the subspace. Weighting data instances using the weighting scheme described above adjusts instance weights at each iteration in an unsupervised fashion. This also helps protect the scenario of *negative transfer* which is known as a critical problem in transfer learning scenario.

3.5.1 Classification using source *candidate* points

The proposed weighting scheme adjusts weights among source and target domain data using separate criteria. For source domain, higher weights are assigned to those which are closely located with respect to target data on the projected subspace. Concretely, a set of neighboring points is selected for each target sample on the projected space and source points that belongs to any of these sets are considered as source *candi-*

Table 3.1: Comparative results in terms of classification accuracy(%) of target domain data for 7 different sub-problems using nearest neighbor classifier. Each sub-problem is in the form of *source* → *target*, where C(Caltech-256), A(Amazon), W(Webcam) and D(DSLR) indicate four different domains.

Methods	C → A 1123 → 958	C → W 1123 → 295	C → D 1123 → 157	A → C 958 → 1123	A → W 958 → 295	A → D 958 → 157	W → D 295 → 157	Avg
Origfeat	23.70	25.76	25.48	26.00	29.83	25.48	59.24	30.78
PCA	36.95	32.54	38.22	34.73	35.59	27.39	77.07	40.36
GFK	41.02	40.68	38.85	40.25	38.98	36.31	80.89	45.28
SA	42.07	32.2	45.86	39.8	37.63	36.94	88.54	46.15
TCA	45.82	30.51	35.67	40.07	35.25	34.39	85.99	43.96
TFL	44.78	41.69	45.22	39.36	37.97	39.49	89.17	48.24
TJM	46.76	38.98	44.59	39.45	42.03	45.22	89.17	49.46
QMI-H	55.95	49.49	45.86	42.12	42.71	37.58	80.89	50.66
QMI-S	**57.72**	55.93	48.41	**41.76**	46.44	38.85	**83.44**	53.22
WQMI-S	56.99	**62.71**	**52.87**	41.59	**53.56**	**46.5**	**83.44**	**56.81**

Table 3.2: Classification accuracy (%) for target domain data using K-NN classifier trained with (I) all projected source points, (II) only projected source *candidate* points, (III) projected *non-candidate* source points.

	C→A	C→W	C→D	A→C	A→W	A→D	W→D
WQMI-S (I)	**56.16**	**60.00**	**50.32**	**41.23**	**49.15**	**47.77**	**83.44**
WQMI-S (II)	**56.06**	**60.00**	**50.32**	**41.23**	**49.15**	**47.77**	**83.44**
WQMI-S (III)	45.62	45.76	44.59	39.36	50.51	44.59	82.8

date points. This attempt will gradually identify source samples that share similar underlying structures and properties with target samples. Therefore, instead of using all the source points, we can train a classifier with only source *candidate* points and apply it to classify target domain data. The experimental result is provided in Table 3.2 where each row represents classification accuracy for 7 different sub-problems using a classifier trained with (I) all the projected source samples, (II) projected source *candidate* samples and (III) source samples that are not in the *candidate* list.

From Table 3.2, it is noted that source *candidate* points are sufficient for classifying

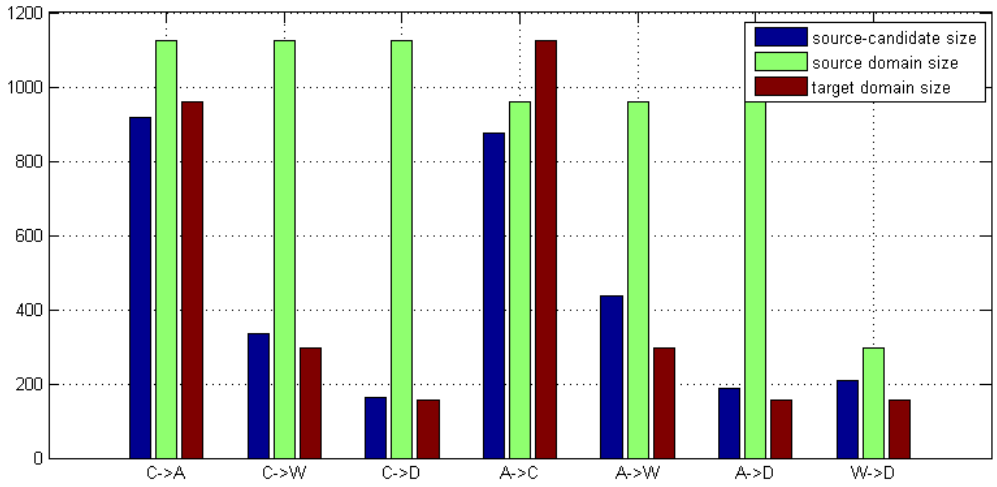

Figure 3.9: Comparison of source *candidate* size, source domain size and target domain size for each sub-problem.

target domain data. The classification accuracies with a classifier trained with all source data and with only *candidate* source data are almost similar. Therefore, our approach can significantly reduce the need for labeled training data for classifier training. Figure 3.9 shows the distribution of source *candidate* size compared to source and target domain data size, computed after termination of the iterative algorithm. Here we can see that the size of source *candidate* points is almost linearly proportional to the size of target domain data which is quite expected. Source *candidate* list will grow with the size of the target points. In an ideal discriminative subspace, data with the same class labels will be grouped together or tightly clustered. Here each cluster consists data from both source and target domain. The weighting scheme assigns higher weights to neighboring source points resulting in non-neighboring source points located distantly in the subspace. Therefore, in the projected space, target data are clustered along with their neighboring source domain data (see Figure 3.13).

Figure 3.10 shows the distribution of source *candidate* points in each class. It shows a balanced distribution of the source *candidate* points among classes and therefore, training a classifier only with the source *candidate* points yield similar performance as with training a classifier using all source points. It is worth noting that the

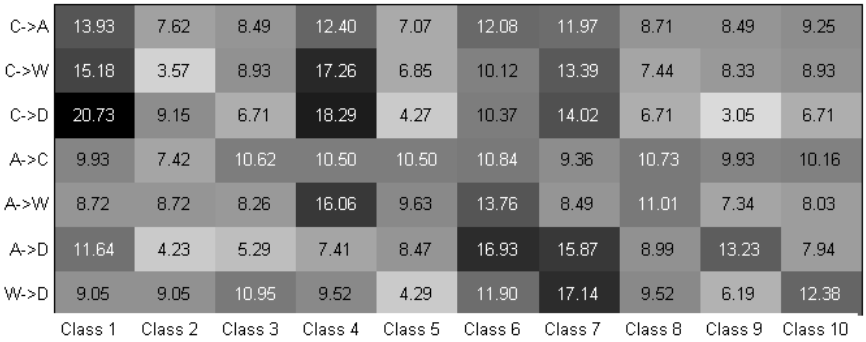

Figure 3.10: Distribution of source *candidate* points (%) among ten different classes for each sub-problem. Each row represents one sub-problem and i-th column denotes percentile of source *candidate* from i-th class over all source *candidate* points.

source *candidate* points contain most of the weights alloted for source domain data as at each iteration, the shrinked weight is re-distributed among source *candidate* points and target points. This also ensures that the learned subspace is dominated by the data samples that share underlying structural similarity and hence closely located on the projected space.

3.5.2 Weight distribution in source domain

In this section, we will closely analyze the weight assignment among source domain data. According to the proposed weighting scheme, source points residing in the neighborhood of target domain data are assigned with higher weights compared to other source points. This will cause source points that are not structurally similar with target ones to be placed at distant locations on the projected space. Hence the distance between each of these *non-candidate* points and any target point also increases through out the iterations as they share least similar structure with target domain data. We can analyze this phenomena by measuring the minimum of the distances between each source point and all the target points, that is,

$$d_m(\boldsymbol{x}_s) = \min dist(\boldsymbol{x}_s, \boldsymbol{x}_i), \quad \text{for } \boldsymbol{x}_i \in \boldsymbol{X}_t$$

Here $dist(.)$ measures the distance between two points and d_m is the minimum distance among pair-wise distances between a source point and all target points. According to the proposed weighting scheme, $d_m(\boldsymbol{x}_s)$ and the weight of $\boldsymbol{x}_s$ are inversely proportional that is, the increase of $d_m(\boldsymbol{x}_s)$ will result in smaller instance weight and vice versa. This behavior is illustrated in Figure 3.11 for four different sub-problems of Office+Caltech dataset. For each source sample, $(d_m(\boldsymbol{x}_s), w)$ pair is rendered using scatter plot, where w is a sample weight. As expected according to the proposed weighting scheme of iterative WQMI-S method, source *candidate* points are mapped in close proximity of target points as they share underlying similarity with target ones. Hence they are assigned with higher weights than *non-candidate* source points, also the mean (red indicator in Figure 3.11) of associated d_m distribution is comparatively smaller than *non-candidate* ones. The opposite behavior hold for *non-candidate* source samples i.e. their associated weights are very small and the mean of d_m distribution is comparatively higher than *candidate* ones. This proves the desired phenomena offered by our weighting scheme.

Another illustration is provided in Figure 3.12 that also expresses the behavior of d_m of a source sample vs. corresponding weight. 50 source samples are randomly selected and constructed a distance vector with corresponding d_m values. Distance and weight vectors are placed side by side where a single stripe represents one sample in either vector. Higher value is mapped with dark color and lower value with light color in the colormap. From this figure, it is observed that when a source point is mapped closely with target points in WQMI-S subspace (i.e. smaller d_m and light stripe), corresponding instance weight is higher (dark stripe) and vice versa.

Finally, Figure 3.13 shows the final output subspace generated by the proposed WQMI-S framework and the current best work in the literature (TJM). Our goal

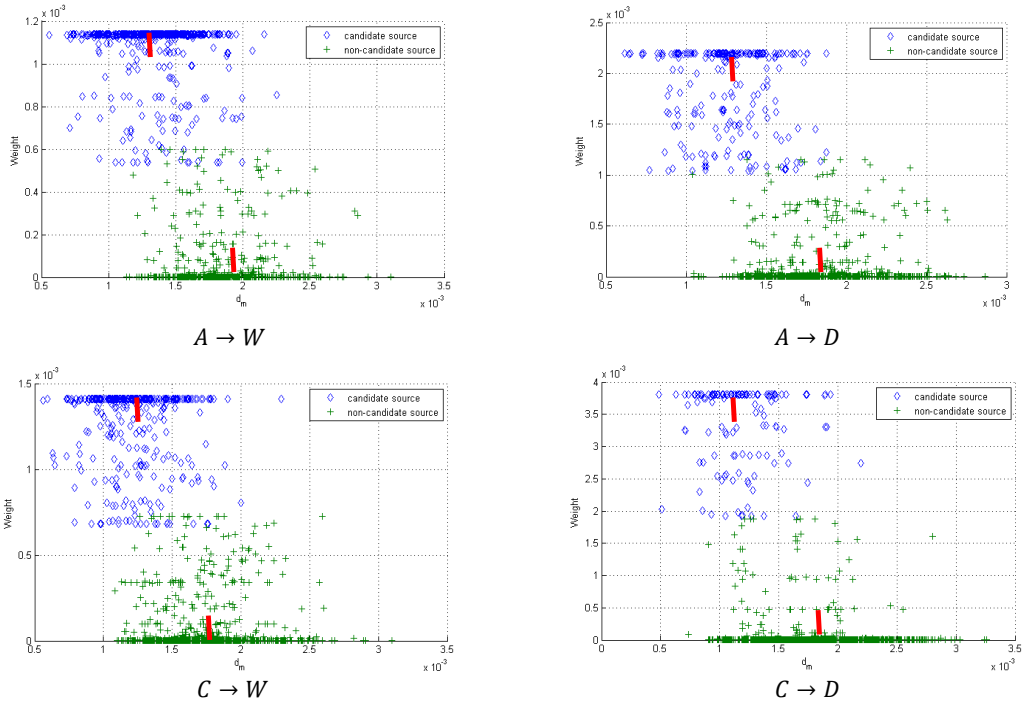

Figure 3.11: Scatter plots of d_m vs. weight of source domain data for 4 different sub-problems. Source *candidate* points are higher weighted than *non-candidate* ones. The mean of d_m distribution is idicated with red line which is lower (higher) for source *candidate* (*non-candidate*) points.

was to learn a domain adaptive discriminative subspace where source and target domain data with same class label will be tightly clustered. The figure is generated using a popular high dimensional data visualization software, known as t-SNE [3]. The projected data generated from the learned subspace are transformed into two dimensional data using t-SNE and shown as a scatter plot in Figure 3.13. The two columns of the figure represent two different sub-problems A→D and A→W. In part (a) and (d) of Figure 3.13, source and target domain data distributions are visualized for two different sub-problems. We can see that target points are clustered with a well-separated margin along with source points that share similar underlying structure. In part (b) and (e) of the figure, we can see the same data distributions annotated with corresponding class labels, where source points are known a priori and target data are annotated with predicted labels. In part (c) and (f) of the figure, we see the same scenario generated using TJM approach. It is worth noting that projected data

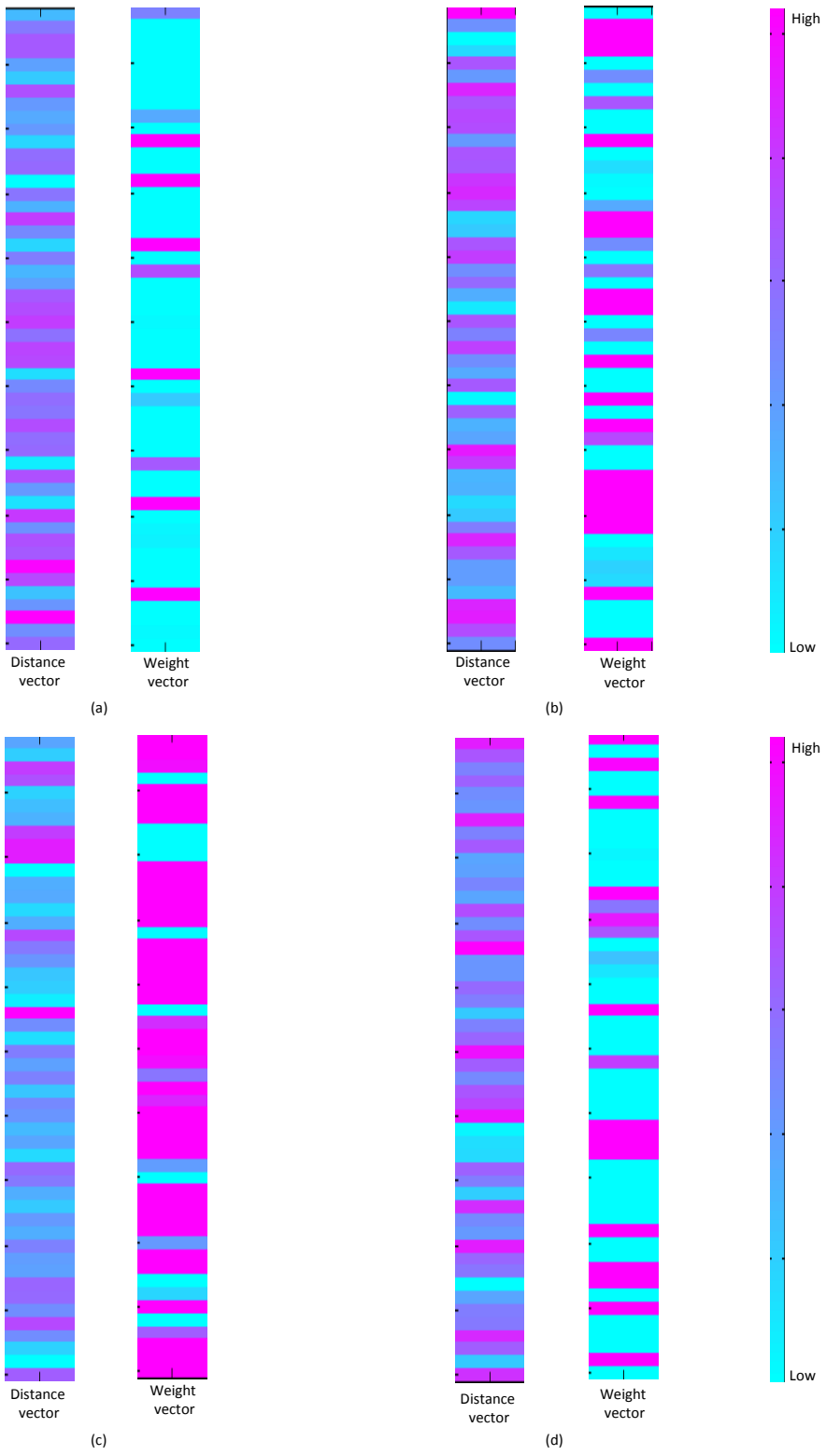

Figure 3.12: Illustration of distance vector constructed with d_m values and corresponding weight vector of 50 randomly selected source samples using colormap. Each sub-figure is representing one sub-problem. In each vector, a single stripe represents d_m or weight of one sample. Higher value of d_m is associated with lower value of corresponding weight and vice versa.

Table 3.3: Comparative results in terms of classification accuracy(%) of target data for 12 different sub-problems using the weighting scheme of source/target balancing. Each sub-problem is in the form of *source* → *target*, where C(Caltech-256), A(Amazon), W(Webcam) and D(DSLR) indicate four different domains.

Methods	C → A	C → W	C → D	A → C	A → W	A → D	W → C	W → A	W → D	D → C	D → A	D → W	Avg
Origfeat	23.70	25.76	25.48	26.00	29.83	25.48	19.86	22.96	59.24	26.27	28.5	63.39	31.37
PCA	36.95	32.54	38.22	34.73	35.59	27.39	26.36	29.35	77.07	29.65	32.05	75.63	39.65
QMI-H	55.95	49.49	45.86	42.12	42.71	37.58	30.37	35.8	80.89	35.71	38.31	61.02	46.32
QMI-S	57.72	**55.93**	**48.41**	**41.76**	46.44	38.85	30.72	**36.74**	**83.44**	**38.38**	**42.48**	77.63	49.88
WQMI-S(S≡T)	**57.93**	55.25	**48.41**	41.59	**47.12**	**39.49**	**32.77**	**36.74**	**83.44**	37.49	40.92	**77.97**	**49.93**

are more tightly clustered around class center in the WQMI-S subspace than TJM subspace resulting in better classification accuracy. The visualization of the scatter plot also provides a practical proof of the efficacy of our proposed iterative subspace learning approach based on maximization of WQMI-S.

3.5.3 Weighting scheme to control source-target imbalance

In this section, experiments are conducted by implementing the second weighting scheme provided in Section 3.3. This process was intended to protect the imbalance in learning a common subspace between source and target domain such that source signal cannot overwhelm the target one and vice versa. See Table 3.3 for a comparative performance of the methods. The last row of this table represents the proposed WQMI-S subspace learning approach with weighting scheme to balance source and target domain size. In some cases where source domain size is very small compared to target domain (e.g. W(n_s=295)→C(n_t=1123)), this type of weight balancing is effective in transfer learning scenario as the source information is too small compared to target domain size. Therefore, boosting the source information to a balanced level such that source and target domain contribute uniformly to the subspace learning, is an effective process in terms of dealing with data size imbalance in source and target domains.

In this chapter, a subspace learning framework is proposed which is incorporated

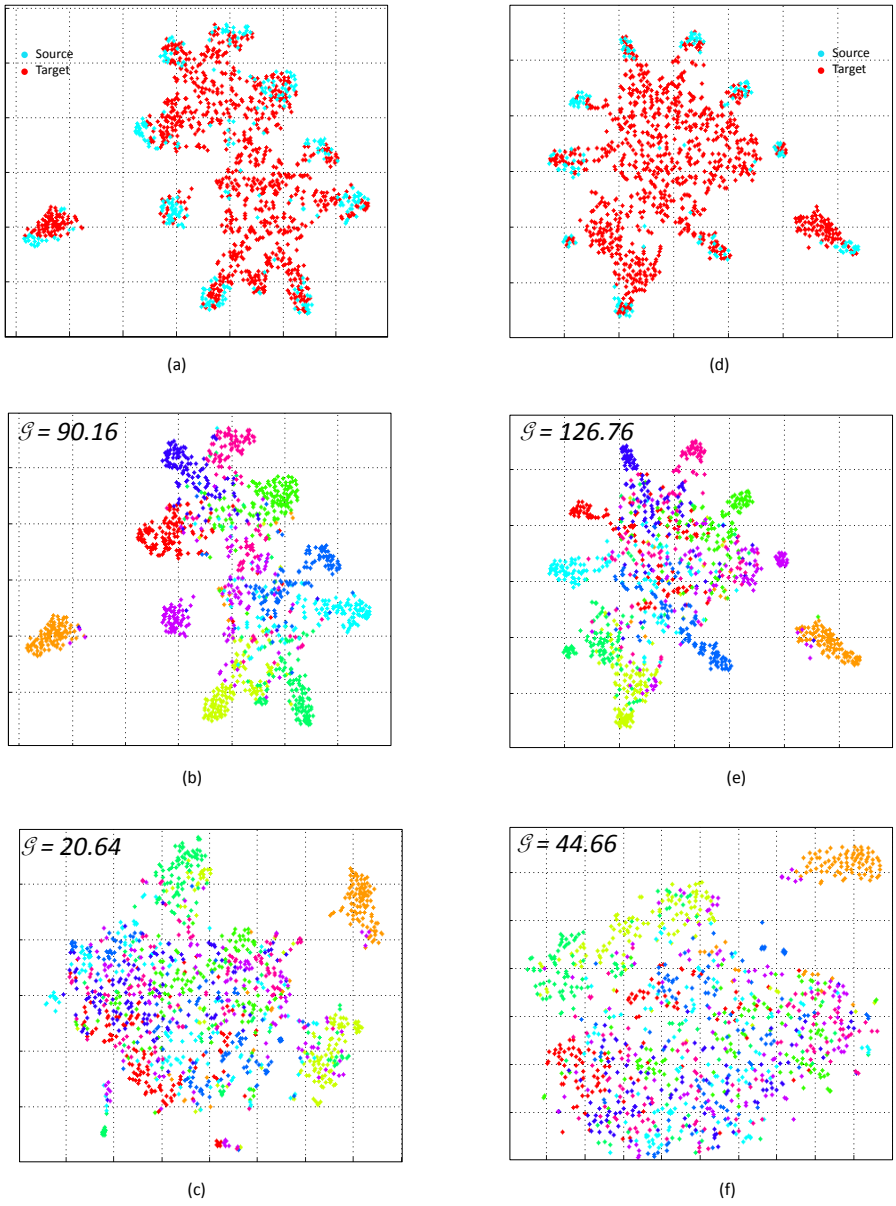

Figure 3.13: Illustration of projected data on WQMI-S subspace in 2d using t-sne method [3]. The left column of sub-figures is for the sub-problem A→D and the right one for A→W. Each sub-figure is a two-dimensional visualization of the learned subspace. (a and d) source and target point distributions in WQMI-S subspace , (b and e) same distributions with data annotated with class labels (represented as color) in WQMI-S subspace, (c and f) class data distributions in TJM subspace. To assess the class discriminative nature of each subspace quantitatively, total scatter metric $\mathcal{G}$ is also reported.

with both instance weighting and soft-class assignment. The experimental evaluation proves the usefulness of weighted instances along with soft-labeling based approach described in previous chapter. Instance weighting is effective in minimizing the impact of unrelated samples in the learning process. In this work, object categories in both domains are same. In future, this work can be extended to recognize novel object category in the target domain with the help of source domain data. A possible outline of such work is provided in the Conclusion chapter of this report.

CHAPTER 4

Linear Transformation By Optimizing Individual Projection Direction

In this chapter, a procedure for linear transformation is proposed where each projection vector of the projection matrix is optimized independently. In linear transformation based dimensionality reduction techniques, projection matrix $\boldsymbol{W}$ is learned that maps the original data into a low-dimensional manifold subspace. Say, $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ represents a data matrix, where each row represents an m dimensional data point. Then a projection matrix $\boldsymbol{W} \in \mathbb{R}^{m \times k}$ is derived that will project the data into a low-dimensional manifold optimizing an objective function. Therefore, the embedded data $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$ in the k dimensional space ($k << m$) will be,

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}$$

And the generic optimization problem for this linear transformation will be,

$$\boldsymbol{W} = \arg \max_{condition} f(\boldsymbol{x})$$

where $f(\boldsymbol{x})$ is the objective function. Therefore, in a k dimensional space, there will be k number of projection directions (projection vector). In previous chapters, such optimization problem is proposed that tries to find a projection matrix with the optimization criterion: maximization of mutual information between data and corresponding class labels in the projected subspace. According to Equation 2.13, the problem formulation is as follows,

$$\boldsymbol{A}^* = \arg \max_{\boldsymbol{A}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{A} = \boldsymbol{I}} \mathrm{tr}\big\{ \boldsymbol{A}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{M} \boldsymbol{K} \boldsymbol{A} \big\}$$

The above equation is a supervised dimensionality reduction technique and the number of projection directions will be $N_c - 1$, where N_c is the total number of classes.

$\boldsymbol{A}^*$ was solved by formulating Equation 4 as a trace ratio optimization problem.

In this chapter, an alternative approach is formulated to find the matrix $\boldsymbol{A}^*$ (in kernel case) or $\boldsymbol{W}$ (in linear case). As discussed in previous chapters, the projected data in QMI-S space will be $\boldsymbol{X}_p = \boldsymbol{K}\boldsymbol{A}^*$ and $\boldsymbol{W} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{A}$. Therefore, we need to solve for $\boldsymbol{A}^*$ that will be used to yield projected data. $\boldsymbol{A}^*$ is matrix of size $n \times k$, where n is the data size and k is the dimension of learned projected space. Therefore, each column $\boldsymbol{A}_i^*$ of the matrix $\boldsymbol{A}^*$ is a projection vector. Traditionally, projection matrix is learned using an optimization of a single objective function and multiplying this projection matrix with data (in original feature space) or kernel (for data projected in kernel space), we can get the final low-dimensional projected data. Our idea in this research is to optimize each projection direction individually. Each projection direction will be optimized by applying 2-class optimization problem using the optimization objective mentioned in Equation (4). The only difference is instead of using N_c class labels associated with the data, we will generate a pseudo label set $+1, -1$. The original label set $C \in \{1, 2, \ldots, N_c\}$ will be randomly partitioned into two subsets and each subset of original labels will assume one pseudo label (+1 or -1). The partition will be generated randomly. An example of some possible partitions are provided in Table 4.1. Therefore, each datum is annotated with either +1 or -1.

As for example, assume we have a dataset corresponding to 5 classes and one random partition yields $\{1, 2\}/\{3, 4, 5\}$. Say, the subset of classes $1, 2$ is represented with a pseudo-label of +1 and the other one $\{3, 4, 5\}$ is represented with a pseudo-label of -1. Thus, data corresponding to class labels 1 and 2 will be annotated with +1 and data corresponding to class labels 3, 4, 5 will be annotated with -1. According to the QMI-S optimization problem, the final feature space will be $N_c - 1$ dimensional, where N_c is the total number of classes. In the two class problem, the final subspace

Table 4.1: Some possible partitions of a set 5 classes into two pseudo labels

Pseudo label	+1	-1
Partition of original label	1,2	3,4,5
	1	2,3,4,5
	3,4	1,2,3
	2,4,5	1,3
	1,2,3,4	5

dimension will be 1. Thus in effect, we are trying to optimize one projection vector as an individual optimization problem with binary class data.

The intuition behind this approach is two fold:

Firstly, The optimization of a single projection direction based on two class problems has leveraged the benefit of discriminative binary feature space [12, 78]. Each vector is optimized with a binary decision function: *which pseudolabel does a data sample belong to?*. Therefore, if n_b is the total number of possible binary partitions, then a feature space will be generated with n_b dimensions. This feature space is analogous to n_b dimensional binary features. Also this will effectively generate a class discriminative feature space and ease the task of classification in that space.

Secondly, The object categories are usually organized in a hierarchical formation. This hierarchical category structure is quite practical, as for example, three object classes 'sedan', 'SUV'and 'van'all can be grouped under a parent category 'motor vehicle'. Therefore, it is intuitive to build a model following this hierarchical structure of object categories i.e. learning a model to be capable of differentiating among high-level object classes and then the next lower level and so on. The proposed partition of class labels considers all possible formation of

object categories into two meta groups. Our conjecture is, the proposed binary partition of classes serves as a proxy to the hierarchical object category structure, as it optimizes a projection vector (or feature) based on a binary grouped categories. The creation of object grouping is totally unsupervised and each optimized feature in the learned feature space is responsible for differentiating between the corresponding binary category groups.

4.1 Problem formulation

We will apply the similar methodology of subspace learning described in previous chapters. The only difference is the partitioning of data. The dimension of final feature subspace is dependent on the number of classes available. Say, there is a set of N_c number of classes available and this set is partitioned randomly. The total number of possible unique partitions will be,

$$n_b = \binom{N_c}{1} + \binom{N_c}{2} + \cdots + q\binom{N_c}{\lfloor \frac{N_c}{2} \rfloor} = 2^{N_c-1} - 1$$

Here $\binom{\cdot}{\cdot}$ denotes binomial coefficient operator. The last term is multiplied by $q = \frac{1}{2-(N_c \mod 2)}$ to break the symmetry for even number of classes. n_b number of optimization problems need to be resolved. This seems a little bit costly, but as each optimization problem is independent of each other, we can take the advantage of parallel programming. For a specific binary partition, each data point is assigned with a pseudo-label of +1 or -1 based on its original class label. Therefore, the set of class labels $C \in \{1, 2, \ldots, N_c\}$ is cast as $C_b \in \{+1, -1\}$. We will apply the weighted QMI-S version of the DA framework with a different weighting scheme. Source data and target data are weighted differently. The soft-labeling is still applicable for this two-class optimization problem except that the label distributions will be of dimension 2. The $\boldsymbol{M}$ matrix is constructed accordingly. One main advantage of this proposed approach is that, the core computation of the optimization problems discussed in

previous chapters is not altered here, rather it will be used as a functional module for optimizing each projection direction. Therefore, the final optimization objective for each projection direction will be (based on the formulation of Chapter 2),

$$\boldsymbol{A_i^*} = \arg \max_{\boldsymbol{A_i^{\mathrm{T}} K A_i = I}} \frac{\mathrm{tr}\{\boldsymbol{A_i^{\mathrm{T}} K M' K A_i}\}}{\mathrm{tr}\{\boldsymbol{A_i^{\mathrm{T}} K K A_i}\}} \tag{4.1}$$

Here $\boldsymbol{K}$ is the Gaussian kernel matrix and $\boldsymbol{A}_i$ is a projection vector. n_b number of $\boldsymbol{A}_i$ need to be learned and all these $\boldsymbol{A}_i$ will form the desired $\boldsymbol{A}$ matrix. $\boldsymbol{A}$ will be constructed by stacking all the optimized projection vectors in a matrix in column order i.e. $\boldsymbol{A} = [\boldsymbol{A}_1 \,|\boldsymbol{A}_2 \,| \ldots |\boldsymbol{A}_{n_b}]$. The projected data will be obtained by $\boldsymbol{X}_p = \boldsymbol{KA}$. Concretely, the original data are encoded with dense binary features where each feature element is an indicator if a point belongs to a pseudo-class label (+1 or -1). If a projection vector is optimized based on a binary partition $\{C_+, C_-\}$, that vector encodes the discriminative information of a point with respect to $\{C_+, C_-\}$. Hence this feature encoding reveals the discriminative information corresponding to the available class labels. Also we argue that the feature space is decorrelated and statistically independent as each projection direction corresponding to a specific feature is optimized separately and independently. Each direction is representing a feature learned with binary partition C_+ or C_-.

Weighting scheme Each projection vector is learned using maximization of weighted QMI-S. Therefore, each data point is assigned weight to participate in the learning process for each projection vector independently i.e. for each optimization problem, instance weighting is applied separately. Assume the optimization of a projection vector for a binary partition of class labels $C_b^i = \{C_+, C_\}$ and the data are partitioned using this binary partition as $\boldsymbol{X}_b = [\boldsymbol{X}_+ \boldsymbol{X}_-]$ i.e. if y_i is the class label of $\boldsymbol{x}_i$ and $y_i \in C_+$, then $\boldsymbol{x}$ belongs to $\boldsymbol{X}_+$ and the same true for C_-. As the optimization is based on a binary class partition, we employ a balanced weight distribution among

$\boldsymbol{X}_+$ and $\boldsymbol{X}_-$. The following equation for instance weighting is applied,

$$w_i = \begin{cases} \frac{0.5}{|\boldsymbol{X}_+|}, & \text{if } \boldsymbol{x}_i \in \boldsymbol{X}_+ . \\ \frac{0.5}{|\boldsymbol{X}_-|}, & \text{if } \boldsymbol{x}_i \in \boldsymbol{X}_- . \end{cases} \tag{4.2}$$

where $|\boldsymbol{X}_+|$ and $|\boldsymbol{X}_-|$ represent the size of the corresponding data partition. It is worth noting that $\sum_{i=1}^{n} w_i = 1$ i.e. the total weight of the dataset is normalized to one. This weighting will prevent the imbalance in the size of data partition and learn a projection vector with respect to the corresponding binary partition and one set of the data partition will not overwhelm the other one during learning.

4.2 Optimization with labeled and unlabeled data

In this section, we will discuss how to optimize a projection direction independently with labeled source domain data and unlabeled target domain data. The same procedure is employed as in Chapter 2 to incorporate unlabeled target domain data into the optimization framework. The target domain data are initialized with uniform class label distribution. The optimization of each projection vector is an iterative process where the target label predictions are updated at each iteration using one of the pseudo-label $\{C_+, C_-\}$ with soft-labeling. The iterative optimization will converge or stop, when the subspace distance between successive iterations fall below a negligible threshold (as described in Chapter 2). Here the subspace is one-dimensional consisting of one projection vector. The weighting scheme will be different than the one described earlier. The equations for the weighting scheme will be,

$$w_i = \begin{cases} \frac{0.25}{|\boldsymbol{X}_+|}, & \text{if } \boldsymbol{x}_i \in \boldsymbol{X}_+ . \\ \frac{0.25}{|\boldsymbol{X}_-|}, & \text{if } \boldsymbol{x}_i \in \boldsymbol{X}_- . \\ \frac{0.50}{|\boldsymbol{X}_t|}, & \text{if } \boldsymbol{x}_i \in \boldsymbol{X}_t . \end{cases} \tag{4.3}$$

The above scheme controls the datasize imbalance among source and target domain and also the imbalance in the data size of binary partition. It is noted that in this

scheme, the target data weights are kept fixed as they are unknown in unsupervised domain adaptation case. Many other weighting schemes can be designed to incorporate labeled and unlabeled data in the optimization process. Therefore, it will be a 3-step optimization process for each projection vector, summarized as follows,

Step 1: Create a binary partition of class labels $\{C_+, C_-\}$ and partition the source domain data based on this binary labels.

Step 2: Learn a one dimensional subspace using weighted QMI-S maximization.

Step 3: Update target label prediction using binary class labels, update the data partition $|\boldsymbol{X}_+|$ and $|\boldsymbol{X}_-|$ and repeat Step-3 till convergence.

The above procedure is similar to the optimization process described in Chapter 2. It is worth noting that the unlabeled target domain data will not be involved in the first iteration of the optimization process. After the 1st iteration, they are projected into the learned subspace and assigned with class label predictions. From the subsequent iterations, the target domain data are involved in the learning process with their predicted class labels. The overall procedure is summarized using a block diagram in Figure 4.1.

4.3 Experiments

In this section, we will verify the proposed method of domain adaptive subspace learning using the previously described Office+Caltech dataset. After learning the projection matrix, all source and target domain data are transformed into the learned vector space. Afterwards, a K-NN classifier is trained with the projected source domain data to apply it for annotating unlabeled target domain data. The classification accuracy is also compared with the methods proposed in previous chapter. We refer the method proposed in this chapter as WQMI-S$^{\text{ib}}$ to differentiate it (individual optimization using binary partition of labels) from other approaches. See Figure 4.2 for

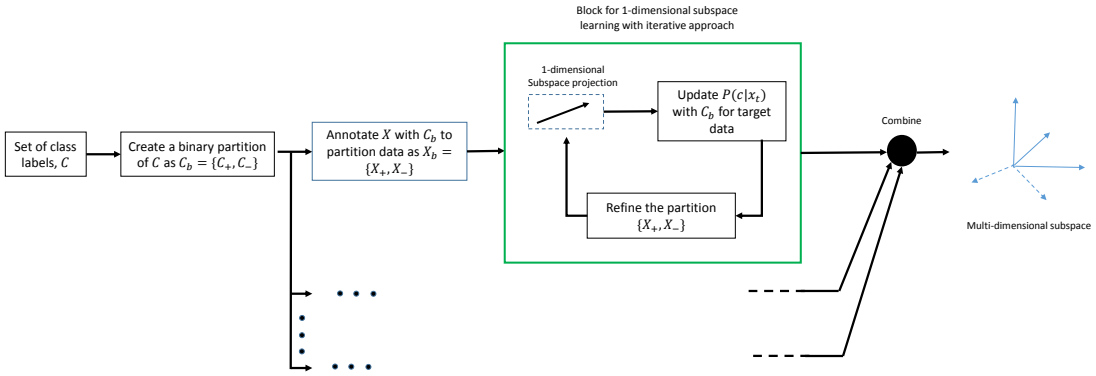

Figure 4.1: Block diagram of the proposed method. Binary partiton of class labels is created and for each partition, a projection vector is learned independently by an iterative approach of WQMI-S maximization. Finally, all the projection vectors are stacked column wise to form a projection matrix.

Table 4.2: Classification accuracy (%) for target domain data in 7 different sub-problems of Office+Caltech dataset. The last row represents the accuracies achieved with the method described in this chapter.

Methods	C→A	C→W	C→D	A→C	A→W	A→D	W→D
QMI-S	**57.72**	55.93	48.41	41.76	46.44	38.85	**83.44**
WQMI-S	56.16	**60.00**	**50.32**	41.23	**49.15**	**47.77**	**83.44**
WQMI-S^{ib}	51.98	50.17	48.41	43.10	44.07	**43.95**	78.34

a detailed comparative results in classification accuracy.

From Table 4.2, it can be noted that the proposed approach can perform similarly well as other approaches proposed in previous chapters. Therefore, this method effectively generates a class discriminative feature encoding which is as good as the original WQMI-S or QMI-S subspace in terms of classification accuracy. Figure 4.2 shows the visualization of the 1 dimensional data projection along a specific projection direction. It is noted that projected data are well separated based on the pseudo class labels $\{C_+, C_-\}$. Hence the data set $\boldsymbol{X}$ are projected by well separated margin that separates $\boldsymbol{X}_+$ from $\boldsymbol{X}_-$. Hence from the figure, it can be implied that each projection vector optimizes the corresponding binary partition problem and generates a class discriminative 1-dimensional space which can well-separate the two pseudo-

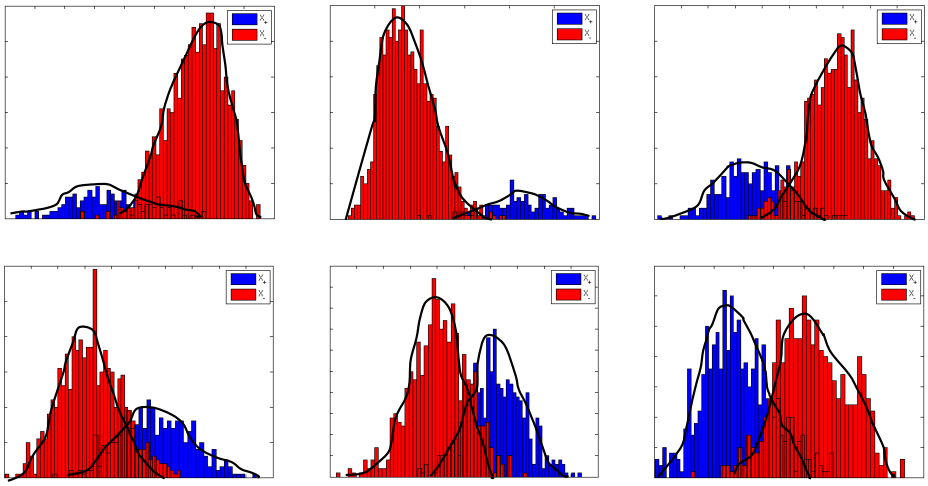

Figure 4.2: Histogram plot of six randomly selected projection vector optimized with binary class labels. Each sub-figure plots the 1-dimensional projection of data along a projection direction which shows a well-defined class separation of $\{C_+, C_-\}$.

classes of the binary partition. The figure shows six randomly selected projection vector.

4.3.1 Applying ICA

Considering each multi-dimensional projection vector as a multi-dimensional signal, we can apply Independent Component Analysis (ICA) [79, 80] to the generated signals in order to decorrelate and extract the original signals. In other words, by applying ICA, the underlying low-dimensional manifold can be extracted from the large number of binary signals or projection vectors. The motivation is intuitive as each individual projection vector is optimized with a binary partition of class labels where each member of a partition contains set of original class labels. Therefore, this projection vector can be considered as a binary signal and the same is true for all other projection vectors. Although each projection vector is an outcome of a binary optimization, the underlying originating signals will be much less than the number of multi dimensional features (as small as the number of original class labels). In Table 4.3, the classification accuracy is reported for different dimensional ICA space. We

Table 4.3: Dimensionality reduction by ICA and PCA to the WQMI-S^{ib} subspace and measuring classification accuracy on the reduced dimensional subspace. Here 'dim'represents the dimension of the final subspace after applying PCA or ICA.

Methods		C→A	C→W	C→D	A→C	A→W	A→D	W→D
WQMI-S^{ib}+ICA	dim=10	**52.30**	46.78	44.59	41.76	**46.10**	**44.59**	82.17
	dim=20	49.58	40.34	46.50	42.03	39.32	35.67	86.62
	dim=30	49.58	37.97	**47.77**	**43.19**	40.68	41.40	89.81
	dim=50	48.33	38.31	44.59	42.83	38.31	33.12	**91.72**
WQMI-S^{ib}+PCA	dim=10	49.06	46.78	46.50	42.12	45.08	42.04	80.25
	dim=20	49.16	47.12	46.50	42.12	44.75	42.04	80.89
	dim=30	49.06	47.12	45.86	42.21	45.08	42.68	80.89
	dim=50	49.16	47.12	46.50	42.21	44.75	42.04	80.89
WQMI-S^{ib}		49.16	**47.46**	46.50	42.03	45.08	42.68	80.89

also reported the classification accuracy for PCA subspace where multi-dimensional data are transformed into PCA space instead of ICA. From Table 4.3, it is noted that ICA can be helpful to leverage the underlying discriminative features and generate a 'better'subspace in terms of class discrimination. This is quite intuitive as the features generated by all the projection vectors is a dense representation of binary features [12, 81] that may encode abundant irrelative information from the originating image signals. ICA can successfully decorrelate the originating multi-dimensional signals (here referred to as projection vectors for the transform matrix) and therefore classification accuracy in this low-dimensional ICA space is higher than the original high dimensional binary feature space. To understand the efficacy of ICA, we also report the classification accuracy by projecting WQMI-S^{ib} data into PCA space. In some of the cases, PCA results are almost comparable to that of ICA.

In future work, investigation can be carried out to develop a process for determining the dimension of the final low-dimensional subspace that can i) minimize entropy

along each feature direction and ii) maximize entropy across all features or projection directions simultaneously.

CHAPTER 5

Applying Non-linear Method for QMI Maximization

Subspace learning based on non-parametric quadratic mutual information involves solving an objective function which is a trace ratio optimization problem (elaborately discussed in previous chapters). In machine learning literature, there are similar problems that deal with trace ratio optimization, one popular example is Fisher Linear Discriminant Analysis (LDA) [82]. Essentially, it is a dimensionality reduction technique which takes the following form,

$$\arg \max_{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{V}=I} \frac{\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}\}}{\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V}\}} \tag{5.1}$$

Here 'tr' refers to trace of a matrix, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a symmetric matrix and $\boldsymbol{B}$ and $\boldsymbol{C}$ is symmetric positive-definite matrices of size $n \times n$. The above expression reaches its maximum (or minimum) when $\boldsymbol{V}$ is an orthogonal basis of the eigen space associated with $\boldsymbol{A}$ subject to a normalization constraint for $\boldsymbol{V}$. This problem is intractable. A widely used resolution is to cast this as a ratio trace problem which is stated as follows,

$$\arg \max_{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{V}=I} \mathrm{tr}\{(\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V})^{-1}(\boldsymbol{V}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V})\} \tag{5.2}$$

The ratio trace problem can be solved by applying generalized eigen value decomposition method (GEVD). As $\boldsymbol{A}$ is symmetric and $\boldsymbol{B}$ is positive definite, there exists n real eigen values corresponding to the generalized eigen value problem, $\boldsymbol{A}\boldsymbol{\Lambda} = \boldsymbol{B}\boldsymbol{U}\boldsymbol{\Lambda}$. Λ is a diagonal matrix of eigen values and $\boldsymbol{U}$ is a matrix of corresponding eigen vectors. For simplicity, $\boldsymbol{C}$ is often used as identity matrix $\boldsymbol{I}$ and the condition defines the orhogonality of the projection matrix $\boldsymbol{V}$. $\boldsymbol{U}$ comprises of a set of eigen vectors

$[\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_p]$ with p largest (or smallest) eigen values. Concretely, the GEVD maximizes $\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}\}$ subject to a constraint $\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V}\} = \boldsymbol{I}$ and the solution can be expressed as follows,

$$\mathrm{tr}\{\boldsymbol{U}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{U}\} = \mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}\} = \lambda_1 + \lambda_2 + \lambda_3 + \cdots + \lambda_p$$

Here λ_i is e eigen value. Although this is a very effective solution, it has been proved in the literature that this simplified version of trace ratio problem often diverges from its true optimization solution [65, 83]. Researchers have proposed a good number of approaches to deal with this issue from different perspective. In our work, we adopt one such solution, known as Newton-Lanczos algorithm and plug in this module into our iterative domain adaptation framework with a hope to yield better classification performance. To recall, the proposed domain adaptation framework tries to learn a low-dimensional subspace by maximizing the following objective function,

$$\boldsymbol{A}^* = \arg\max_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{A}=\boldsymbol{I}} \frac{\mathrm{tr}\{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{M}'\boldsymbol{K}\boldsymbol{A}\}}{\mathrm{tr}\{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{K}\boldsymbol{A}\}} \tag{5.3}$$

Here $\boldsymbol{K}$ is the kernel matrix and $\boldsymbol{A}$ is the desired matrix to obtain the projection matrix $\boldsymbol{W} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{A}$. In previous chapters, this optimization problem is resolved based on generalized eigen value solution and we found that GEVD approach shows a competitive performance in term of classification accuracy for the unlabeled target domain data. In this chapter, we will investigate the issue of applying non-linear Newton-Lanczos method to solve the above trace ratio problem. Therefore, the only focus is to deal with the trace ratio problem in our framework, other modules of the proposed DA algorithm will remain same.

5.1 Newton-Lanczos algorithm

Following [83], we will provide a brief overview of the Newton-Lanczos algorithm. The algorithm deals with a ratio of two traces, therefore the final ratio output is a

scalar value. It is proved that there must be an optimum ratio value β^* for which the trace ratio will reaches its maximum or minimum. Therefore, the ratio of two traces will follow this upper bound,

$$\frac{\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}\}}{\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V}\}} \leq \beta* \tag{5.4}$$

This implies that $\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}\} - \beta*\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V}\} \leq 0$ i.e. for an optimal β^* we can have $\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}(\boldsymbol{A}-\beta^*\boldsymbol{B})\boldsymbol{V}\} \leq 0$ for any orthogonal $\boldsymbol{V}$. However, the maximum of a trace is the sum of the largest eigen values. Therefore, we can have a maximum β^* when trace of $\boldsymbol{A}-\beta^*\boldsymbol{B}$ reaches zero, in other words, when the sum of the largest p eigen values of $\boldsymbol{A}-\beta^*\boldsymbol{B}$ equals zero, we can have the optimal β^*. Here, p is the dimension of the projected space. Therefore, according to [83], the trace ratio optimization problem can be stated as,

$$f(\beta) = \max_{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}=I} \mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}(\boldsymbol{A}-\beta\boldsymbol{B})\boldsymbol{V}\} = \lambda_1 + \lambda_2 + \cdots + \lambda_p \tag{5.5}$$

It has been proved that function f is a non-increasing function of β and $f(\beta) = 0$ iff $\beta = \beta^*$. In their work, they proposed a non-linear method for finding the optimal β^*.

The search for the optimal β is a root finding problem for the function f. This involves solving eigen value problems for the matrix $\boldsymbol{A}-\beta\boldsymbol{B}$ several times. The authors employed newton's fixed point iteration method along with the Lanczos procedure to make the overall procedure simpler. They derived the derivative function of $f(p)$ in details. Readers are suggested to go through [83] for a detailed explanation and derivations of the method. The derivative of f has been derived as $-\mathrm{tr}\{\boldsymbol{V}(\boldsymbol{\beta})^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V}(\boldsymbol{\beta})\}$, where $\boldsymbol{V}(\boldsymbol{\beta})$ is the matrix of eigen vectors for a specific root β. Using this formulation, Newton's method involves a fixed point iteration formula for finding the optimal β which takes the following form,

$$\beta_{new} = \beta - \frac{\mathrm{tr}\{\boldsymbol{V}(\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{A}-\beta\boldsymbol{B})\boldsymbol{V}(\boldsymbol{\beta})\}}{-\mathrm{tr}\{\boldsymbol{V}(\boldsymbol{\beta})^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V}(\boldsymbol{\beta})\}}$$

$$= \frac{\mathrm{tr}\{\boldsymbol{V}(\boldsymbol{\beta})^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}(\boldsymbol{\beta})\}}{\mathrm{tr}\{\boldsymbol{V}(\boldsymbol{\beta})^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}(\boldsymbol{\beta})\}}$$

According to the algorithm provided in [83], an iterative loop takes place to update β till *convergence*. At each iteration, two main steps are implemented, they are i) Applying Lanczos algorithm to compute p largest eigen vectors and eigen values for the matrix $\boldsymbol{A} - \beta^{*}\boldsymbol{B}$, ii) Update β using the above mentioned rule of Newton's method. Therefore, their algorithm is an iterative approach involving Lanczos method to obtain eigen solution of $\boldsymbol{A} - \beta^{*}\boldsymbol{B}$ and Newton's method to update the root of $f(\beta)$ i.e. the ratio of two traces β. The *convergence* criterion is also provided in their work and it has be guaranteed that if $\boldsymbol{A}$ is a symmetric matrix and $\boldsymbol{B}$ is a symmetric positive-definite matrix, then their iterative approach to solve trace ratio problem reaches its optimal value.

Our contribution here to utilize this alternative solution of trace ratio problem into our DA framework. We can use this procedure for both unweighted and weighted version of QMI-S approach as both of them require solving trace ratio optimization problem. As an example, here we will plug-in Newton-Lanczos algorithm of trace ratio optimization to our proposed DA method (see Algorithm 2) and analyze how the performance in terms of classification accuracy for target domain data is effected. To do this, we need to transform Equation (5.3) as follows, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}^{\frac{1}{2}}$ can be substituted by another variable $\boldsymbol{V}^{\mathrm{T}}$ i.e. $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{K}^{\frac{1}{2}} = \boldsymbol{V}^{\mathrm{T}}$ to cast this equation into the standard from of trace ratio optimization (Equation (5.1)). Therefore, Equation (5.3) takes the following form,

$$\boldsymbol{V}^{*} = \arg\max_{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}=\boldsymbol{I}} \frac{\mathrm{tr}\Big\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{M'}\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{V}\Big\}}{\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{V}\}} \tag{5.6}$$

Now considering $\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{M'}\boldsymbol{K}^{\frac{1}{2}} = A$ and $\boldsymbol{K} = \boldsymbol{B}$, we can apply Newton-Lanczos method to solve the above optimization problem. Once we get $\boldsymbol{V}$, we can obtain our desired

matrix $\boldsymbol{A}$ as follows,

$$\boldsymbol{A}^{\mathrm{T}}\,\boldsymbol{K}^{\frac{1}{2}} = \boldsymbol{V}^{\mathrm{T}}$$

$$\boldsymbol{A} = \boldsymbol{K}^{-\frac{1}{2}}\,\boldsymbol{V}$$

After learning $\boldsymbol{A}$, the projected data is computed as, $\boldsymbol{X}_p = \boldsymbol{K}\boldsymbol{A} = \boldsymbol{K}^{\frac{1}{2}}\,\boldsymbol{V}$. It is noted that, computation of the inverse square root of the kernel matrix is not needed which is essentially an expensive operation. As the projected data is computed, the other modules of the proposed DA framework should be same. We hope that applying non-linear method for finding a low-dimensional discriminative subspace will produce better outcome compared to the traditional generalized eigen vector method. Using this approach, the updated procedure is described in Algorithm 5.

The *convergence* criterion in this algorithm is same as before defined in Chapter 2. That is, distance between two subspaces in a Grassmannian manifold is used as a stopping criterion for this algorithm. If this distance (measured as angle between two subspaces) falls below a minimum threshold, then the algorithm will stop. For details of the *convergence* criterion, readers are suggested to review Chapter 2.

5.2 Experiments

Office dataset is used to verify the efficacy of this non-linear optimization method. As before, there are four domains involved: Webcam (W), DSLR (D), Amazon (A) and Caltech (C). The experimental protocol will be same as was followed in previous chapters. All the projected source data will be used to learn a classifier which is applied to detect the class labels of the target data in projected space.

Table 5.1 shows the performance of two different approaches: QMI-S (soft-labeling induced maximization of quadratic mutual information described in Chapter 2) and QMI-S$^{\text{N-L}}$ (soft-labeling induced quadratic mutual information maximized with Newton-

Algorithm 5 Applying Newton-Lanzcos method for Subspace learning based on iterative QMI-S (QMI-S$^{\text{N-L}}$)

Input: Data matrix $\boldsymbol{X}$=[$\boldsymbol{X}_s$;$\boldsymbol{X}_t$]$\in \mathbb{R}^{n\times d}$ where source data $\boldsymbol{X}_s \in \mathbb{R}^{n_s\times d}$ and target data $\boldsymbol{X}_t \in \mathbb{R}^{n_t\times d}$, source data labels $[y_1, y_2, \ldots, y_s]^T$.

Output: $\boldsymbol{X}_p \in \mathbb{R}^{n\times k}$, k-dimensional projected data.

Initialization: For $\boldsymbol{x}_i^t \in \boldsymbol{X}_t$, $P(c|\boldsymbol{x}_i^t) = \frac{1}{N_c}$ for each $c \in \{1, 2, \ldots N_c\}$. For $\boldsymbol{x}_i^s \in \boldsymbol{X}_s$, $P(c|\boldsymbol{x}_i^s) = 1$ if $c = y_i$ and $P(c|\boldsymbol{x}_i^s) = 0$ otherwise, for each $c \in \{1, 2, \ldots N_c\}$.

Compute a centralized Gaussian kernel matrix, $\boldsymbol{K}\in \mathbb{R}^{n\times n}$.

repeat

Step-I:

  Compute $\boldsymbol{M}$ matrix using Eq.(2.11).

  Set $\boldsymbol{M'} = \frac{\boldsymbol{M}+\boldsymbol{M}^{\mathrm{T}}}{2}$

  repeat

    Assign $\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{M'}\boldsymbol{K}^{\frac{1}{2}} = A$ and $\boldsymbol{K} = \boldsymbol{B}$

    Solve standard eigen problem, $(\boldsymbol{A} - \beta\boldsymbol{B})\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}$.

    Update β as $\beta = \frac{\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}\}}{\mathrm{tr}\{\boldsymbol{V}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{V}\}}$

  until *convergence (β)*

  Compute projected data $\boldsymbol{X}_p = \boldsymbol{K}^{\frac{1}{2}}\boldsymbol{V}$.

Step-II:

  Train a classifier f using projected source data $\boldsymbol{X}_p^s$ and apply it to update $P(c|\boldsymbol{x}_i^t)$ with soft-labeling.

until *convergence*.

Table 5.1: Classification accuracy(%) of target data for 12 different sub-problems along with MI between data and corresponding labels using Newton-Lanzcos based nonlinear optimization . Each sub-problem is in the form of *source* $\rightarrow$ *target*, where C(Caltech-256), A(Amazon), W(Webcam) and D(DSLR) indicate four different domains.

		C→A	C→W	C→D	A→C	A→W	A→D	W→C	W→A	W→D	D→C	D→A	D→W
QMI-S	Acc	57.72	55.93	48.41	41.76	46.44	38.85	30.72	36.74	83.44	38.38	42.48	77.63
	MI($\times 10^{-5}$)	0.099	0.0883	0.093	0.1241	0.1481	0.162	0.1584	0.1904	0.2569	0.1515	0.1953	0.2491
QMI-S$^{\text{N-L}}$	Acc	49.37	46.1	43.95	39.63	42.71	36.94	30.10	35.49	82.17	38.56 ↑	41.65	82.37 ↑
	MI($\times 10^{-5}$)	0.1501	0.1405	0.1506	0.1664	0.196	0.2124	0.1886	0.2273	0.2968	0.1671	0.2095	0.2845

Lanczos method of nonlinear optimization). It is worth noting that for each of the subproblems, the QMI between data and corresponding labels (after final annotation of target domain data) is increased compared to the GEVD approach applied in QMI-S algorithm. The classification accuracies are comparable to that of the original QMI-S approach. The iterative QMI-S$^{\text{N-L}}$ helps improve the maximization of mutual information. The classification performance is not improved in accordance with increased QMI. Newton-Lanczos optimization causes the target data to be more expanded nearing source data on the learned subspace from the beginning of the iterative algorithm. This causes the label predictions of target data to become freeze very fast, may be in couple of initial iterations.

As future work, an interleaved approach involving prediction for target data and Newton-Lanczos optimization can be applied. Concretely, line 11 to line 14 of Algorithm 5 is dedicated for the Newton-Lanczos optimization which has been adopted from [83] and we have used the software package provided by the authors. From the algorithm, it is noted that $\boldsymbol{M'}$ matrix is being updated at every iteration and hence $\boldsymbol{A}$ matrix is learned. After the convergence of Newton-Lanczos algorithm, target data predictions are updated resulting in update of $\boldsymbol{M'}$ matrix eventually. In the interleaving approach, we can move this target prediction and $\boldsymbol{M'}$ computation inside the optimization loop of Newton-Lanczos process. Therefore, instead of delaying the update of $\boldsymbol{A}$ till maximization of trace ratio is reached, by conducting this advanced update, it is possible to enhance the iterative approach resulting in the overall convergence of the framework faster.

CHAPTER 6

Conclusion and Future Work

In computer vision and machine learning area, solving a classification, detection or regression task depends on an effective model which is trained to deal with that task. Training a model efficiently has been a challenging task as it needs to generalize over varied testing environment which might be significantly different from the training domain. With the advancement of image capturing devices like cellular phones, web camera, DSLR camera etc. enormous amount of visual data are being generated at every moment. Most of these data remain unannotated. Manual annotation of these huge number of images by human intervention is often infeasible and creates a bottleneck to train an effective supervised model for the task at hand. Transfer learning has evolved as a rescue to overcome this scenario. In our research, we focus on a specific case of transfer learning, widely known as 'Domain Adaptation'. We focused on the classification task in DA context, the proposed framework can also be extended to other vision tasks like detection, regression etc. which will be considered as a future research direction from here. The main challenge was to overcome the distribution divergence between source and target domain data such that a classifier trained using source information can be deployed effectively in the unknown target environment. If all the target domain data are unlabeled, then it becomes more challenging to establish a knowledge transfer from source to target domain. One intuitive hypothesis is that if the set of object classes is same across domains, then source and target data share some underlying common structural properties in a low-dimensional manifold. Utilizing this hypothesis, researchers have proposed a good number of al-

gorithms based on learning a shared subspace using source and target data, where the distribution divergence has been minimized. In our research, we investigate one step ahead and try to learn a class discriminative subspace resulting in minimization of differences in marginal and conditional distributions across domains.

We have been able to successfully integrate soft class labeling and instance weighting into the subspace learning framework based on non-parametric quadratic mutual information, referred to as WQMI-S. We also show that applying an iterative process of label prediction of target data, each target point along with neighboring source points can influence its neighboring unlabeled target samples for label prediction on the learned subspace. Thus label information from source domain gradually propagates towards target domain with a goal to predict the unknown target data and these newly annotated data eventually influence the construction of QMI maximized subspace for the next iteration. Instance weighting into the QMI formulation has also been proved effective as not all the source domain data share similar structural properties with target domain data. Hence instances are weighted based on their shared relevance with a very intuitive weighting scheme that automatically adjusts instance weights through out the iterations. In our research, we used trace ratio optimization to learn the subspace and apply eigen value decomposition method to generate the basis of the subspace as an approximate solution of trace ratio optimization.

Future work

We investigated some alternative learning processes of QMI subspace in Chapter 4 and Chapter 5 and provided intuitive analysis to shed light on future research direction. In Chapter 4, we proposed an alternative subspace learning methodology based on a binary task optimization along each projection direction. The intuition was to generate a class discriminative dense feature subspace such that the raw data can be

encoded using class discriminative information. This is a generic paradigm for dense feature generation, which has been also adapted in domain adaptation context. We showed that the performance of this process in DA framework is comparable to other proposed methods. As a future work, applying Independent Component Analysis (ICA) to the generated projection vectors in order to separate the originating source signals will be an intuitive approach to try. In the experiment section of Chapter 4, an experimental analysis to support application of ICA is provided. Each projection vector is optimized based on a binary partition of class labels. Therefore, each of them represents a multidimensional signal that is optimized to discriminate between two components of the binary partition. Each partition is actually constructed from the original label set. This is the motivation to apply ICA with a goal to unravel the originating source signals, where the number of originating source signals is same as the number of class labels used to generate binary partitions. Determining the dimensionality of the final ICA subspace is till an open issue.

In Chapter 5, non-linear Newton-Lanczos process is used in QMI-S maximization instead of generalized eigen vector decomposition. As future work, an interleaved approach involving prediction for target data and optimization of Newton-Lanczos algorithm can be applied. Non-liner Newton-Lanczos iteratively search for a subspace till trace ratio is maximized and multiple intermediate subspaces are generated in this process. In our domain adaptation framework, target data predictions need to be updated at each iteration. Therefore, this prediction process can be coupled with the Newton-Lanczos optimization by updating the $\boldsymbol{M}$ matrix after each intermediate learned subspace involved in Newton-Lanczos. Lastly, we propose another research direction to extend our work for unseen or novel classes in the target domain which are not present during training phase or in source domain. The question to ask here is *how to recognize an object that is not seen during training phase of a classification*

model. Researchers in this area are also dealing with this issue and try to overcome the difficulties of facing novel object category with the help of auxiliary data related to novel class, this line of research is often referred to as *zero shot learning*. One possible direction is to learn separate manifolds for each class in source domain and project each target data onto these manifolds to construct a class *relevance* vector. These vectors might be further clustered to form pseudo labels for target data that will serve as an auxiliary data for the novel class.

In summary, we deal with a very challenging but interesting topic of recent computer vision research that has a valuable and significant impact in building efficient models for an AI task. Algorithms and variants that are optimized to overcome bottlenecks usually faced during the learning process of a robust adaptive model, have been proposed with extensive analysis .

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] L. Duan, D. Xu, and I. Tsang, "Learning with augmented features for heterogeneous domain adaptation," *arXiv preprint arXiv:1206.4660*, 2012.

[3] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[4] M. Fritz, B. Leibe, B. Caputo, and B. Schiele, "Integrating representative and discriminant models for object category detection," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1363–1370, IEEE, 2005.

[5] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 11–18, IEEE, 2006.

[6] R. Aljundi, R. Emonet, D. Muselet, and M. Sebban, "Landmarks-based kernelized subspace alignment for unsupervised domain adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *IEEE International Conference on Computer Vision, 2013*, pp. 2960–2967, 2013.

[8] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer joint matching for unsupervised domain adaptation," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 1410–1417, IEEE, 2014.

[9] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[10] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *Signal Processing Magazine, IEEE*, vol. 32, no. 3, pp. 53–69, 2015.

[11] M. Pandey and S. Lazebnik, "Scene recognition and weakly supervised object localization with deformable part-based models," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1307–1314, IEEE, 2011.

[12] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 12, pp. 2037–2041, 2006.

[13] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1717–1724, 2014.

[14] C. Wang and S. Mahadevan, "Heterogeneous domain adaptation using manifold alignment," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, p. 1541, 2011.

[15] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, "Undoing the damage of dataset bias," in *Computer Vision–ECCV 2012*, pp. 158–171, Springer, 2012.

[16] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *arXiv preprint arXiv:0902.3430*, 2009.

[17] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in neural information processing systems*, pp. 1410–1418, 2009.

[18] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 4, pp. 594–611, 2006.

[19] M. Rohrbach, S. Ebert, and B. Schiele, "Transfer learning in a transductive setting," in *Advances in neural information processing systems*, pp. 46–54, 2013.

[20] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, "Multi-task learning for classification with dirichlet process priors," *The Journal of Machine Learning Research*, vol. 8, pp. 35–63, 2007.

[21] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*, pp. 759–766, ACM, 2007.

[22] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2066–2073, IEEE, 2012.

[23] M. Long, J. Wang, G. Ding, J. Sun, and P. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2200–2207, 2013.

[24] J. Donahue, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell, "Semi-supervised domain adaptation with instance constraints," in *IEEE Conference on Computer Vision and Pattern Recognition, 2013*, pp. 668–675, 2013.

[25] H. Daumé III, A. Kumar, and A. Saha, "Frustratingly easy semi-supervised domain adaptation," in *Proc. of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pp. 53–59, 2010.

[26] H. Daumé III and D. Marcu, "Domain adaptation for statistical classifiers.," *J. Artif. Intell. Res.(JAIR)*, vol. 26, pp. 101–126, 2006.

[27] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proceedings of the twenty-first international conference on Machine learning*, p. 114, ACM, 2004.

[28] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[29] M. Rohrbach, M. Stark, and B. Schiele, "Evaluating knowledge transfer and zero-shot learning in a large-scale setting," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1641–1648, IEEE, 2011.

[30] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, "Zero-shot learning through cross-modal transfer," in *Advances in neural information processing systems*, pp. 935–943, 2013.

[31] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Self-taught clustering," in *Proceedings of the 25th international conference on Machine learning*, pp. 200–207, ACM, 2008.

[32] Z. Wang, Y. Song, and C. Zhang, "Transferred dimensionality reduction," in *Machine learning and knowledge discovery in databases*, pp. 550–565, Springer, 2008.

[33] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, pp. 193–200, ACM, 2007.

[34] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Transferring naive bayes classifiers for text classification," in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, vol. 22, p. 540, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[35] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[36] G. Foster, C. Goutte, and R. Kuhn, "Discriminative instance weighting for domain adaptation in statistical machine translation," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 451–459, Association for Computational Linguistics, 2010.

[37] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data source," in *Proceedings of the 22nd international conference on Machine learning*, pp. 505–512, ACM, 2005.

[38] J. Blitzer, R. McDonald, and F. 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.

[39] H. Daumé III, "Frustratingly easy domain adaptation," *arXiv preprint arXiv:0907.1815*, 2009.

[40] Y. Aytar and A. Zisserman, "Tabula rasa: Model transfer for object category detection," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2252–2259, IEEE, 2011.

[41] J. J. Lim, *Transfer learning by borrowing examples for multiclass object detection*. PhD thesis, Massachusetts Institute of Technology, 2012.

[42] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision–ECCV 2010*, pp. 213–226, Springer, 2010.

[43] B. Kulis, K. Saenko, and T. 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.

[44] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint," *International Journal of Computer Vision*, vol. 109, no. 1-2, pp. 74–93, 2014.

[45] W. Li, L. Duan, D. Xu, and I. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," 2013.

[46] J. Hoffman, E. Rodner, J. Donahue, T. Darrell, and K. Saenko, "Efficient learning of domain-invariant image representations," *arXiv preprint arXiv:1301.3224*, 2013.

[47] L. Fe-Fei, R. Fergus, and P. Perona, "A bayesian approach to unsupervised one-shot learning of object categories," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1134–1141, IEEE, 2003.

[48] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 999–1006, IEEE, 2011.

[49] A. Zweig and D. Weinshall, "Exploiting object hierarchy: Combining models from different category levels," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, IEEE, 2007.

[50] M. Fink, "Object classification from a single example utilizing class relevance metrics," *Advances in Neural Information Processing Systems*, vol. 17, pp. 449–456, 2005.

[51] T. Tommasi and B. Caputo, "Frustratingly easy nbnn domain adaptation," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 897–904, IEEE, 2013.

[52] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[53] D. Bouzas, N. Arvanitopoulos, and A. Tefas, "Graph embedded nonparametric mutual information for supervised dimensionality reduction," 2015.

[54] K. Torkkola, "Feature extraction by non parametric mutual information maximization," *The Journal of Machine Learning Research*, vol. 3, pp. 1415–1438, 2003.

[55] M. Hellman and J. Raviv, "Probability of error, equivocation, and the chernoff bound," *IEEE Trans. Inf. Theor.*, vol. 16, pp. 368–372, Sept. 2006.

[56] R. M. Fano and D. Hawkins, "Transmission of information: A statistical theory of communications," *American Journal of Physics*, vol. 29, no. 11, pp. 793–794, 1961.

[57] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[58] J. C. Principe, *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010.

[59] J. N. Kapur, *Measures of information and their applications*. Wiley-Interscience, 1994.

[60] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, pp. 1065–1076, 1962.

[61] C. Archambeau, M. Valle, A. Assenza, and M. Verleysen, "Assessment of probability density estimation methods: Parzen window and finite gaussian mixtures," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pp. 4–pp, IEEE, 2006.

[62] B. Scholkopft and K.-R. Mullert, "Fisher discriminant analysis with kernels," *Neural networks for signal processing IX*, vol. 1, p. 1, 1999.

[63] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[64] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[65] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *Neural Networks, IEEE Transactions on*, vol. 20, no. 4, pp. 729–735, 2009.

[66] Y.-F. Guo, S.-J. Li, J.-Y. Yang, T.-T. Shu, and L.-D. Wu, "A generalized foley–sammon transform based on generalized fisher discriminant criterion and its ap-

plication to face recognition," *Pattern Recognition Letters*, vol. 24, no. 1, pp. 147–158, 2003.

[67] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, "Trace ratio vs. ratio trace for dimensionality reduction," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[68] J. J. Thiagarajan and K. N. Ramamurthy, "Subspace learning using consensus on the grassmannian manifold," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 2031–2035, IEEE, 2015.

[69] J. Hamm and D. D. Lee, "Grassmann discriminant analysis: a unifying view on subspace-based learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 376–383, ACM, 2008.

[70] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Tech. Rep. 7694, California Institute of Technology, 2007.

[71] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[72] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[73] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[74] L. Ingber, "Simulated annealing: Practice versus theory," *Mathematical and computer modelling*, vol. 18, no. 11, pp. 29–57, 1993.

[75] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *Circuits and Devices Magazine, IEEE*, vol. 5, no. 1, pp. 19–26, 1989.

[76] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," tech. rep., Citeseer, 2002.

[77] L. Rokach and O. Maimon, *Data Mining and Knowledge Discovery Handbook*, ch. Clustering Methods, pp. 321–352. Boston, MA: Springer US, 2005.

[78] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Computer Vision–ECCV 2012*, pp. 759–773, Springer, 2012.

[79] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, pp. 1129–1159, 1995.

[80] L. K. Hansen, J. Larsen, and T. Kolenda, "Blind detection of independent dynamic components," *In proc. IEEE ICASSP'2001*, vol. 5, pp. 3197–3200, 2001.

[81] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[82] M. Welling, "Fisher linear discriminant analysis," *Department of Computer Science, University of Toronto*, vol. 3, 2005.

[83] T. T. Ngo, M. Bellalij, and Y. Saad, "The trace ratio optimization problem," *SIAM review*, vol. 54, no. 3, pp. 545–569, 2012.

[84] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[85] J. Hoffman, E. Rodner, J. Donahue, B. Kulis, and K. Saenko, "Asymmetric and category invariant feature transformations for domain adaptation," *International Journal of Computer Vision*, pp. 1–14, 2014.

[86] J. Hoffman, T. Darrell, and K. Saenko, "Continuous manifold based adaptation for evolving visual domains," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 867–874, IEEE, 2014.

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

[88] T. Tommasi, F. Orabona, and B. Caputo, "Safety in numbers: Learning categories from few examples with multi model knowledge transfer," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3081–3088, IEEE, 2010.

APPENDIX A

List of Symbols

In this thesis, boldface lowercase letters are used to represent vectors, e.g. $\boldsymbol{x}$, and boldface uppercase letters for matrices, e.g., $\boldsymbol{M}$. Italicized letters are reserved for scalars, e.g., i, n.

$\mathcal{D}_s, \mathcal{D}_t$ source and target domain.

n total number of data points.

n_s, n_t number of data points from source and target domain respectively $n_s + n_t = n$.

k dimensionality of learned subspace.

d dimensionality of raw input feature space.

$\boldsymbol{X}_s$ **,**$\boldsymbol{X}_t$ data matrix of source and target domain data respectively, $\boldsymbol{X}_s \in \mathbb{R}^{n_s \times d}$, $\boldsymbol{X}_t \in \mathbb{R}^{n_t \times d}$.

$\boldsymbol{X}$ input data matrix constructed with $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$, $\boldsymbol{X} = [\boldsymbol{X}_s\,; \boldsymbol{X}_t\,] \in \mathbb{R}^{n \times d}$

$\boldsymbol{x}$ a general data point in $\mathbb{R}^d$.

$\mathbf{C}$ set of class labels.

N_c number of unique object categories.

$p(\boldsymbol{x})$ marginal data density distribution. $\int p(\boldsymbol{x}) d\boldsymbol{x} = 1$.

$P(c)$ prior probability of class c.

$p(\boldsymbol{x}, c)$ joint probability density function.

$p(c|\boldsymbol{x})$ the class conditional distribution.

$\boldsymbol{w}$ weight vector consisting of individual sample weight, $\boldsymbol{w} = [w_1, w_2, \cdots, w_n]^T \in \mathbb{R}^{n\times 1}$.

S_c a shorthand notation for $P(c)$.

$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}}\, e^{\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)}$$

where $|\boldsymbol{\Sigma}|$ is the determinate of the covariance matrix and e is the base of the natural logarithm.

$\boldsymbol{V}$ matrix of eigen vectors (in column order) associated with standard eigen decomposition of matrix $\boldsymbol{M}$.

$\boldsymbol{K}$ centralized Gaussian kernel matrix.

Φ matrix with data mapped in Kernel space, $\Phi \in n \times m$.

W projection matrix for linear transformation by maximizing QMI.

$\boldsymbol{A}$ matrix of coefficients for representing $\boldsymbol{W}$ in terms of $\boldsymbol{\Phi}$, $\boldsymbol{W} = \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{A}$,$\boldsymbol{A} \in \mathbb{R}^{n\times k}$.

α fraction of weight subtracted from each instance weight, w_i.

Ω set of *candidate* data (both source and target domain) that go through weight re-adjustment.

Ω_s, Ω_t sets of *candidate* source and target domain data respectively.

Ω_{th} set of target domain data that are *confident* in corresponding class label predictions.

β fraction of weight used as 'bonus'to be distributed among members of Ω_{th} (Chapter 3).

VITA

MOHAMMAD NAZMUL ALAM KHAN

Candidate for the Degree of

Doctor of Philosophy

Dissertation: **A FRAMEWORK FOR TRANSFER LEARNING: MAXIMIZATION OF QUADRATIC MUTUAL INFORMATION TO CREATE DISCRIMINATIVE SUBSPACES**

Major Field: Computer Science

Biographical:

Personal Data: Born in Dhaka, Bangladesh on May 2, 1984.

Education:
Received the B.S. degree from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, 2007, in Computer Science and Engineering
Completed the requirements for the degree of Doctor of Philosophy with a major in Electrical Engineering Oklahoma State University in July, 2016.

Experience:
Graduate Teaching Assistant, Department of Computer Science, Oklahoma State University, Stillwater, OK
Graduate Research Assistant, Department of ECE, Oklahoma State University, Stillwater, OK
Software Intern, Central Rural Electric Cooperative, Stillwater, OK
Summer Intern, Kimray Inc., Oklahoma City, OK