Domain Adaptive Computational Models for Computer Vision

by

Hemanth Kumar Demakethepalli Venkateswara

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2017 by the
Graduate Supervisory Committee:

Sethuraman Panchanathan, Chair
Baoxin Li
Hasan Davulcu
Jieping Ye
Shayok Chakraborty

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

The widespread adoption of computer vision models is often constrained by the issue of domain mismatch. Models that are trained with data belonging to one distribution, perform poorly when tested with data from a different distribution. Variations in vision based data can be attributed to the following reasons, viz., differences in image quality (resolution, brightness, occlusion and color), changes in camera perspective, dissimilar backgrounds and an inherent diversity of the samples themselves. Machine learning techniques like transfer learning are employed to adapt computational models across distributions. Domain adaptation is a special case of transfer learning, where knowledge from a source domain is transferred to a target domain in the form of learned models and efficient feature representations.

The dissertation outlines novel domain adaptation approaches across different feature spaces; (i) a linear Support Vector Machine model for domain alignment; (ii) a nonlinear kernel based approach that embeds domain-aligned data for enhanced classification; (iii) a hierarchical model implemented using deep learning, that estimates domain-aligned hash values for the source and target data, and (iv) a proposal for a feature selection technique to reduce cross-domain disparity. These adaptation procedures are tested and validated across a range of computer vision applications like object classification, facial expression recognition, digit recognition, and activity recognition. The dissertation also provides a unique perspective of domain adaptation literature from the point-of-view of linear, nonlinear and hierarchical feature spaces. The dissertation concludes with a discussion on the future directions for research that highlight the role of domain adaptation in an era of rapid advancements in artificial intelligence.

ACKNOWLEDGEMENTS

You gave me the opportunity to grow and the freedom to explore
and when success came my way there was none who cheered more;
'Dr. Panch is my advisor' is a badge I will proudly wear,
its my privilege and honor you're guru, guide and chair.

I am indebted to Dr. Ye, for guidance in machine learning,
He showed me the ropes when he took me under his wing;
To the marquee team of Drs. Chakraborty, Davulcu, Li and Ye,
it means laurels to me to have you on my committee;

SCIDSE and ASU, your support has been relentless;
For seeing in me a TA - Navabi, Mutsumi and Calliss;
Christina from advising and Kathy et al. from Fulton;
Pam, Monica, Teresa and Brint - I thank you a ton.

Exploring uncharted waters upon the merry CUbiC boat,
a riot of swashbucklers kept my dreams and spirits afloat;
We conquered the horizon and raised the ASU flag high,
ably led by Cap'n Troy, we strived and aimed for the sky.

To CUbiC champions Morris, Terri, Rita and my mentor Vineeth;
To friends who cheered me on - Sai, Indu, Ganesh and Prasanth;
To buddies Mike, Corey, Scott, Brian, Arash, Ramesh, Ramin,
Meredith, Bijan, and partners Jose, Hiranmayi, Ragav and Binbin;
You've made this possible and I give many thanks from deep within.

The Sai Center in Mesa is an oasis in the desert;
Its people nourished my body and their music my soul;
On dreary days when I felt broken and wanted to quit,
I drank from its cool spring and rejuvenated my spirit.

Your love and blessings have guided me through and through;
Swami, Tata, Ajji, Amma and Naanna, I offer this to you.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

In a recent article in the Harvard Business Review, dated November 2016, leading computer science researcher Andrew Ng, compared artificial intelligence to electricity saying, "*A hundred years ago electricity transformed countless industries;* 20 *years ago the internet did, too. Artificial intelligence is about to do the same*," Ng (2016). On the other hand, the unprecedented success of artificial intelligence in recent years has also raised concerns from eminent scientist Stephen Hawking and prominent entrepreneur Elon Musk, regarding the implications of superhuman intelligence on humanity's future, Editorial (2016).

With exponential growth in technology, the utopian world of superhuman intelligent robots or, as some would like to call it - the 'technological singularity', [1] may not be far away. However, human intelligence is a competitive benchmark that machine intelligence is seeking to emulate and eventually outperform. One of the hallmarks of human intelligence, is the ability to adapt and transfer knowledge across multiple domains. For e.g., if a human is familiar with a language, they can easily understand almost anyone speaking it, even if they were to hear them for the first time, or, if a person has learned to drive a car, they can easily adapt to driving a truck, by adapting some of their previously learned knowledge to the new setting. In order to enhance machine intelligence to the level of human intelligence and beyond, machine learning models will have to model knowledge transfer. The ability to transfer knowledge will provide tools to process the vast amounts of unlabeled data available in the form of online video, audio, images and text. These advances in articial intelligence and ma-

[1] `https://en.wikipedia.org/wiki/Technological_singularity`

chine learning will greatly benefit a wide range of applications including, healthcare, communication, education and clean energy.

This dissertation discusses machine learning models for transferring knowledge. The concept of knowledge transfer and the need for adaptive machine learning models is illustrated in the following example. Consider the example of an autonomous driving car developed in the Silicon Valley, California. This system has been trained with data gathered from navigating roads in the valley. It can read and interpret road signage, avoid hitting pedestrians and safely navigate from point A to point B, in normal traffic conditions. However, the same level of performance cannot be expected when the car is put to test on the streets of London. In London, the data gathered by the car's sensors need to be interpreted differently and the rules for driving in London are quite different from those in the Silicon Valley - the signage is different, there are no turns on red, driving is to the left side of the road are merely some of them. The challenge lies with the fact that the car has been trained to interpret California street data and not London street data. A self-driving car will need to be trained with London street data (signage, images, pedestrians, etc.), before it can be put to test on the streets of London. However, it would be expensive and time consuming to acquire such labeled training data and retrain a new self-driving car. It is in these situations domain adaptation algorithms help to transfer the knowledge gained from learning to drive in California, and reduce the training effort when adapting the self-driving car to a new environment.

Domain adaptation algorithms are usually trained to adapt between two domains, the source and the target. The data from the source and the target, although similar, is from different distributions, for e.g. California street data vs. London street data. A machine learning model trained on the source dataset, is often adapted to the target dataset. The challenge for transfer of knowledge occurs when there is limited or no

labeled data in the target domain, which makes it hard to train models that need some form of supervision. This dissertation discusses developing models in domain adaptation for the computer vision application of classification.

1.1 Goals and Motivations

The goal of this dissertation is to propose domain adaptation models for image classification problems in computer vision. It seeks to highlight the role of domain adaptation in machine learning and summarize the literature in domain adaptation. It also intends to outline a set of directions for research in the future.

This dissertation has been inspired by some overarching challenges and goals in artificial intelligence, big-data analysis and ubiquitous computing. The motivations are highlighted below.

1. *Reverse-Engineer the Brain*: The National Academy of Engineering (NAE) has laid down 14 challenges for the 21st century [2] ranging from sustainability, clean resources and medicine. One of these challenges is to reverse-engineer the brain. It seeks to create machines capable of emulating human intelligence, the impact of which will go far beyond artificial intelligence, with applications in healthcare, manufacturing and communications. The ability to transfer previously gained knowledge and adapt to a wide range of data inputs, will be crucial to the success of this venture. A potential solution to this hard challenge will have to account for knowledge transfer and domain adaptation.

2. *Big Data Challenges*: The four V-s of big data are *Volume*, *Velocity*, *Variety* and *Veracity* [3]. With the advent of the Internet of Things (IoT), the scale of data (volume), the speed of data generation and processing (velocity), the

[2] http://www.engineeringchallenges.org/challenges.aspx

[3] https://en.wikipedia.org/wiki/Big_data

different types of data (variety) and the uncertainty of the data (veracity), are amplified. Most of this data is often not annotated (labeled). It is hard to create useful models using unlabeled data based on unsupervised learning. It is also impractical to train task specific models for every variation in data. Transfer learning and domain adaptation will play a crucial role in utilizing the inexhaustible supply of unlabeled data and in customizing pre-trained models and adapting them to the task at hand.

3. *Person-Centered Adaptation*: Human Centered Computing (HCC) is a branch of computer science that transcends traditional Human Computer Interaction (HCI) paradigms by placing the human at the center of research activity. In recent years, the concept of Person-Centered Computing (PCC) Panchanathan *et al.* (2012), has evolved to adapt computing to individual needs rather than adopting a 'one-size-fits-all' approach. The roots of PCC are based on the philosophy of *co-adaptation*, where a bidirectional interaction between the user and the system is used to co-adapt a system tailored to the needs and idiosyncrasies of a user. Computing has become nearly ubiquitous with computing devices embedded in the environment and in the devices that people use, such as phones, watches, wristbands, etc. To co-adapt this complex environment to the needs of an individual would require sophisticated models in transfer learning and domain adaptation. Such adaptation models will ensure individualized designs, while also maintaining applicability to a wide range of users.

1.2 Contributions

The contributions of the dissertation are as follows.

1. A new organization was proposed for the literature in domain adaptation. The various methods developed over the years are organized on the basis of feature

spaces viz., *linear*, *nonlinear* and *hierarchical* methods. This provides a unique perspective on domain adaptation models.

2. A linear model for domain adaptation was proposed with the Coupled-Support Vector Machine. When there are a few labeled data points available in the target domain, this method can be applied to learn SVM decision boundaries for the source and the target simultaneously, using standard SVM libraries.

3. A nonlinear embedding transform (NET) based on kernel-Principal Component Analysis (kernel-PCA) was implemented. In the NET, the joint distributions (data and labels) of two domains are aligned along with embedding the data to ensure that classification is enhanced. This work also introduced a validation procedure in the absence of labeled target data.

4. A hierarchical feature space (deep learning) procedure was developed. A deep learning network called Domain Adaptive Hashing (DAH), was created to estimate domain aligned hash values for the source and target images. A unique loss for unlabeled target data was introduced, which ensured discriminative hash values for the target.

5. An information gain based feature selection technique was proposed. The NP-hard problem of feature selection was solved using approximate solutions from related problems in graph theory. A nonlinear feature selection method was proposed for reducing cross domain disparity.

6. A new dataset (*Office-Home*) was introduced for object classification based domain adaptation. The dataset consists of images from 65 categories and 4 domains. The dataset consists of nearly $15,500$ images - a significant increase over existing datasets.

1.3 Dissertation Outline

The dissertation is structured in the following manner.

Chapter 2 provides an overview of domain adaptation. The first section is an introduction to domain adaptation. It introduces the different types of learning with labeled and unlabeled data. It is followed by an introduction to transfer learning and a discussion on the various types of transfer learning and the different kinds of domain adaptation with regards to availability of labeled data. The second section provides a theoretical analysis of domain adaptation along with a discussion on generalization bounds for performance. The third section describes the role of domain adaptation in computer vision research. It outlines the current state of research in domain adaptation, followed by a brief outline on the deep learning trends in domain adaptation. This section also introduces the *Office-Home* dataset for deep learning based domain adaptation, which is one of the contributions of this dissertation.

Chapter 3 is a literature survey on the research in domain adaptation for computer vision. The survey is organized to reflect the contributions of this dissertation. It looks upon domain adaptation in the light of feature spaces. It has a section on *linear* feature spaces, where linear feature models for domain adaptation are classified into categories. The following section gives an overview of *nonlinear* feature models that have been applied towards domain adaptation applications in computer vision. The subsequent section outlines the latest trend in computer vision - deep learning methods for domain adaptation. These are considered as *hierarchical* feature models because of the highly nonlinear and hierarchical nature of feature extraction. The chapter concludes with a discussion on a few miscellaneous models.

Chapter 4 describes a linear domain adaptation technique based on support vector machines. It is a semi-supervised domain adaptation procedure where there is labeled source data along with a few labeled target data samples. The chapter introduces a Coupled-Support Vector Machine (Coupled-SVM) model that trains two classifiers; one for the source and the other for the target. The source and target SVM decision boundaries are learned as a pair of coupled classifiers with the similarity between data from the source and target domains being modeled as the similarity between SVM decision boundaries. The coupled SVM formulation is reduced to a standard single SVM model that can be trained using existing SVM libraries.

Chapter 5 progresses from linear feature spaces to nonlinear feature spaces. This chapter outlines an unsupervised domain adaptation technique using kernel methods. It introduces the Nonlinear Embedding Transform (NET) for unsupervised domain adaptation. The NET reduces cross-domain disparity through nonlinear domain alignment. The NET model also embeds the domain-aligned data such that similar data points are clustered together. This results in enhanced classication. To determine the parameters in the NET model (and also for other unsupervised domain adaptation models), a validation procedure is introduced by sampling source data points that are similar in distribution to the target data.

Chapter 6 introduces a hierarchical feature based domain adaptation method. This chapter outlines a deep learning domain adaptation method based on hashing. This model exploits the feature learning capabilities of deep neural networks to learn representative hash codes to address the domain adaptation problem. There are two advantages to estimating a hash values: (i) hash values enable efcient storage and retrieval of data due to their fast query speed and low memory costs and (ii) during

prediction, the hash code of a test sample can be compared against the hash codes of the training samples to arrive at a more robust prediction. Extensive empirical studies on multiple transfer tasks corroborate the usefulness of the framework in learning efcient hash codes which outperform existing competitive baselines for unsupervised domain adaptation.

Chapter 7 proposes a feature selection method for domain adaptation. The first part of the chapter lays out a feature selection technique based on conditional mutual information (CMI). The technique adapts algorithms from related problems in graph theory to do feature selection. These algorithms provide very good approximations to the NP-hard problem of feature selection. The second part of the chapter outlines a model for domain adaptation using feature selection. This is based on the hypothesis that selecting the right features would help align the domains of the source and target. Preliminary results with digit based datasets validate the approach with promising results.

Chapter 8 outlines directions for future research in domain adaptation. This chapter is based on the insights gained from working on this dissertation. The proposed directions are a segue to the next generation of domain adaptation algorithms. It deals with arguments for a better understanding and modeling of domain shift and calls for the introduction of new and large datasets based on an improved interpretation of domain shift. Some of these outlooks for the feature should help guide new researchers in domain adaptation in formulating their research agenda.

Chapter 9 concludes the dissertation by summarizing the contributions of the dissertation.

1.4 Previously Published Work

The contents of Chapter (4) are based on previously published work, "*Coupled Support Vectors Machines for Supervised Domain Adaptation*" in Venkateswara *et al.* (2015b). Chapter (5) is adapted from published works, "*Multiresolution Match Kernels for Gesture Recognition*" in Venkateswara *et al.* (2013), and "*Nonlinear Embedding Transform for Unsupervised Domain Adaptation*" in Venkateswara *et al.* (2016) and "*Model Selection with Nonlinear Embedding for Unsupervised Domain Adaptation*" in Venkateswara *et al.* (2017a). Chapter (6) on deep learning based domain adaptation, is adapted from work accepted at the CVPR 2017 conference, "*Deep Hashing Network for Unsupervised Domain Adaptation*" Venkateswara *et al.* (2017b). Chapter (7) on feature selection is based partly on published work, "*Efficient Approximate Solutions to Mutual Information Based Global Feature Selection*" in Venkateswara *et al.* (2015a).

Chapter 2

DOMAIN ADAPTATION - BACKGROUND

While chapter (1) highlighted the role of domain adaptation in machine learning, this chapter provides a more detailed and formal introduction to domain adaptation in computer vision. It is organized as follows. Section (2.1) begins with a discussion on the types of learning leading to a classification of the different kinds of transfer learning and domain adaptation. Section (2.2) provides some theoretical guarantees when using a domain adaptation model. It also describes proxy measures for estimating the amount of discrepancy between domains. Section (2.3) gives an introduction to how domain adaptation models are developed and evaluated in computer vision. It discusses the datasets that are used and some drawbacks to the way research in domain adaptation is currently approached. The section concludes with a discussion on deep learning based domain adaptation that is the cutting edge of research in this area.

2.1 Introduction to Domain Adaptation

In order to understand the nature of domain adaptation, it will be useful to outline the different learning paradigms in machine learning and discuss how domain adaptation is related to them.

2.1.1 Unsupervised, Supervised and Semi-Supervised Learning

The traditional learning paradigms of machine learning are *unsupervised*, *supervised* and *semi-supervised* learning (Chapelle *et al.* (2006)). In an unsupervised learning setup, data is available as a set of n examples, $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, where $\boldsymbol{x}_i \in \mathcal{X}$

for all $i \in [1, \ldots, n]$. $\mathcal{X}$ is the feature space of input data. For e.g., $\mathcal{X}$ could be a subset of the Euclidean space of d-dimensions, $\mathbb{R}^d$, or a space of normalized gray scale images of dimensions $M \times N$, $\mathbb{R}^{M \times N}$. The data $\mathbf{X}$ is drawn from an arbitrary distribution $P(X)$, which is unknown. Here X is a random variable and $p(X = \boldsymbol{x}_i)$ is the probability for a data instance $\boldsymbol{x}_i$. The task of unsupervised learning is to model the data by finding structure in it. This effectively means, learning or estimating the distribution $P(X)$ given $\mathbf{X}$. Other approaches to understanding the data are clustering, quantile estimation, dimensionality reduction and outlier detection.

In a supervised learning paradigm, data is available in the form of labeled pairs, $\mathbf{X}_l = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$. Like in the case of unsupervised learning, $\boldsymbol{x}_i \in \mathcal{X}$ for all $i \in [1, \ldots, n]$ and $y_i \in \mathcal{Y}$ are called the labels. The label space $\mathcal{Y}$ could be discrete binary like $\mathcal{Y} = \{0, 1\}$ or discrete {1,. . . ,C} for applications in classification. $\mathcal{Y}$ could be a real number ($\mathbb{R}$) for applications in regression or it could be a space similar to $\mathcal{X}$ for applications is structured prediction Nowozin and Lampert (2011). The data $\mathbf{X}_l$ is drawn from a joint distribution $P(X, Y)$, which is unknown. Here, X and Y are random variables and $p(X = \boldsymbol{x}_i, Y = y_i)$ is the probability for the joint occurrence of $(\boldsymbol{x}_i, y_i)$. The task of supervised learning is to learn a mapping function $f : \mathcal{X} \to \mathcal{Y}$ based on the training example pairs in $\mathbf{X}_l$. This mapping function is used to predict the label y for a new data point $\boldsymbol{x}$. There are two standard approaches to estimating this mapping function. *Generative* models learn the joint distribution $P(X, Y)$ by estimating the marginal distributions $P(X)$, the prior distributions $P(Y)$ and the conditional distributions $P(X|Y)$. So, when a new data point $\boldsymbol{x}$ is provided, the model predicts its label y by applying the Bayes theorem,

$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{\int_{\mathcal{Y}} p(\boldsymbol{x}|y)p(y)dy} \tag{2.1}$$

Some of the examples of generative models are, Gaussian Mixture models, Hidden

Markov models, Naïve Bayes. *Discriminative* models on the other hand learn the posterior distribution $P(Y|X)$ directly as a function in order to predict the label y. Some examples of discriminative models are Support Vector Machines (SVM), Random Forests and Logistic Regression.

In the semi-supervised learning paradigm, there are two datasets available, $\mathbf{X}_l = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n_l}$ and $\mathbf{X}_u = \{\boldsymbol{x}_i\}_{i=1}^{n_u}$, where $\mathbf{X}_l$ is the labeled dataset and $\mathbf{X}_u$ is the unlabeled dataset. The data $\mathbf{X}_l$ is drawn from a joint distribution $P(X, Y)$, which is unknown. Similarly, $\mathbf{X}_u$ is drawn from a distribution $P(X)$, which is unknown. In addition, the marginal distribution $P(X)$ for $\boldsymbol{x} : (\boldsymbol{x}, y) \in \mathbf{X}_l$ is the same as $P(X)$ for $\boldsymbol{x} \in \mathbf{X}_u$. Similar to supervised learning, the goal is to learn a mapping function $f : \mathcal{X} \to \mathcal{Y}$ based on the training example pairs in $\mathbf{X}_l$ and $\mathbf{X}_u$. The challenge is, the number of labeled data points are few with $n_l \ll n_u$ and it may not be possible to estimate the joint probability $P(X, Y)$ or learn a discriminator $P(Y|X)$ with the limited labeled data. There are two approaches to solving semi-supervised learning problems, (i) *inductive* and *transductive*. In inductive learning, a prediction model is learned which can predict the labels y for the entire space, i.e., $\forall \boldsymbol{x} \in \mathcal{X}$, whereas in transductive learning, a prediction model is learned to predict labels y only for the unlabeled data $\mathbf{X}_u$. An example of a transductive model is the Transductive SVM in Joachims (1999).

Semi-supervised learning is closely related to domain adaptation. In a domain adaptation setting, the labeled dataset $\mathbf{X}_l$ can be viewed as the source dataset $\mathcal{D}_s$ and the unlabeled dataset $\mathbf{X}_u$ can be viewed as the target dataset $\mathcal{D}_t$. There is however one important distinction. The marginal distributions of the source $P_S(X)$ and the target $P_T(X)$, are different. If this distribution difference were to be bridged, semi-supervised learning algorithms can be easily adopted to domain adaptation. In the following subsection, domain adaptation is discussed in greater detail by contrasting

it with other transfer learning methods.

2.1.2 Transfer Learning

The traditional machine learning paradigms seen in the previous section, train statistical models in order to make predictions on unseen data in the future. However, the models learned using these paradigms can be viewed as *static* - meaning, they are not capable of adapting to changes in the data. In other words, these models do not guarantee optimal performance if the test data is vastly different from the data the models were trained with. For example, a facial expression recognition system trained using only female subjects, may perform poorly when tested with data from male subjects, because men tend to have more rugged facial features along with facial hair even. *Transfer learning* is the branch of machine learning that trains models to learn from multiple sources of data and adapt to test data from a different setting. Transfer learning can be said to be inspired from the way humans learn. For example, a human who has learned to ride a bicycle can adapt to riding a motorcycle with limited training and effort. Transfer learning can also be viewed as a problem of 'learning to learn', which is the ability to perform life-long learning and adaptation (Thrun and Pratt (1998)).

In the remainder of this subsection, different types of transfer learning are outlined and compared with each other. To this end, some notation and definitions are outlined below in line with Pan and Yang (2010). For the purpose of this discussion, the definitions of a "Domain" and "Task" are outlined. A domain $\mathcal{D}$ is said to consist of two components, a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ that governs the feature space, where $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$ is the set of samples from the feature space. For example, if the learning task is audio transcription, the data from different subjects can be treated as different domains. The voice of the subject can be

considered to be the feature space $\mathcal{X}$ and $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is the set of audio signals (words) uttered by the subject where $P(X)$ is the marginal probability that governs $\mathbf{X} \subset \mathcal{X}$. Two domains are considered different if their feature spaces are different (example, different users) or their probability distributions are different (example, casual conversation vs reading a report). If $\mathcal{D} = \{\mathcal{X}, P(X)\}$ is a domain, then a task $\mathcal{T}$ consists of two components, $\mathcal{T} = \{\mathcal{Y}, f(.)\}$, where $\mathcal{Y}$ is the label space and $f(.)$ is the function $f : \mathcal{X} \rightarrow \mathcal{Y}$. The function $f(.)$ is unknown and in a supervised setting, it is learned from training data pairs $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The function $f(x)$ can then be used to predict the label of a test instance $\boldsymbol{x}$. From a probabilistic perspective $f(\boldsymbol{x})$ can be viewed as the posterior probability $p(y|\boldsymbol{x})$. Sometimes a domain $\mathcal{D}$ can also be viewed as consisting of a joint space of features and labels and a joint probability distribution $\{(\mathcal{X} \times \mathcal{Y}), P(X, Y)\}$. In most of the examples of transfer learning, two domains are usually considered, the *source* and the *target*. With a slight abuse of notation, a source dataset is represented as a collection of data points $\mathcal{D}_s = \{(\boldsymbol{x}_1^s, y_1^s), \ldots, (\boldsymbol{x}_{n_s}^s, y_{n_s}^s)\}$, where $\boldsymbol{x}_i^s \in \mathcal{X}_S$ and $y_i \in \mathcal{Y}_S$. Similarly, a target dataset is represented as $\mathcal{D}_t = \{(\boldsymbol{x}_1^t, y_1^t), \ldots, (\boldsymbol{x}_{n_t}^t, y_{n_t}^t)\}$, where $\boldsymbol{x}_i^t \in \mathcal{X}_T$ and $y_i \in \mathcal{Y}_T$. The following definition provides a good starting point for the discussion on different kinds of transfer learning.

Definition 2.1.1. ***Transfer Learning****: (Pan and Yang (2010)) Given a source domain $\mathcal{D}_S$ and a source learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a target learning task $\mathcal{T}_T$, transfer learning aims to improve the target predictive function $f_T(.)$ using $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.*

When two domains are different ($\mathcal{D}_S \neq \mathcal{D}_T$), then either their feature spaces are not the same ($\mathcal{X}_S \neq \mathcal{X}_T$, like audio samples from two different subjects) or their probability distributions are different ($P_S(X) \neq P_T(X)$, like audio from casual talk vs

audio from reading a book) or both. Similarly, when two tasks are different ($\mathcal{T}_S \neq \mathcal{T}_T$, where $\mathcal{T}_S = \{\mathcal{Y}_S, P_S(Y|X)\}$ and $\mathcal{T}_T = \{\mathcal{Y}_T, P_T(Y|X)\}$), then either their label spaces are different ($\mathcal{Y}_S \neq \mathcal{Y}_T$, like transcription vs sentiment analysis) or the posterior distributions are different ($P_S(Y|X) \neq P_T(Y|X)$, the case where source and target are unbalanced in the number of user defined classes). Each of these combinations of domain difference and task difference gives rise to learning scenarios that can be addressed using transfer learning. The following paragraphs outline the most prominent learning paradigms that can be viewed as special cases of transfer learning.

A transfer learning model retains knowledge from one or more tasks, domains or distributions and applies that knowledge to develop an effective hypothesis for a new task, domain or distribution (Bruzzone and Marconcini (2010)). The different types of learning paradigms in machine learning that can be classified as transfer learning are, *multitask learning*, *self-taught learning*, *sample selection bias*, *lifelong machine learning*, *zero-shot learning* and *domain adaptation*.

1. *Multitask Learning (MTL)*: In this setting, labeled training data is available for a set of K tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\}$ where each task is associated with a different domain, $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$. Given the k^{th} task, it is not possible to estimate the empirical joint distribution $\hat{P}_k(X, Y)$ reliably with data from the k^{th} domain, $\mathcal{D}_k = \{\boldsymbol{x}_k^i, y_k^i\}_{i=1}^{n_k}$, $\boldsymbol{x}_k^i \in \mathcal{X}_k$ and $y_k^i \in \mathcal{Y}_k$. A good approximation for $\hat{P}_k(X, Y)$ is learned by exploiting the training data from all the domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$ and learning all the tasks simultaneously Bruzzone and Marconcini (2010). The tasks are different irrespective of the equality of the domains. In terms of availability of labels, all the domains usually have labels. Even by this definition, $\hat{P}_k(X, Y)$ is inferred by combining the data from all the tasks and learning all the tasks simultaneously. An introduction and a survey of multitask learning procedures is provided in Caruana (1997); Thrun and Pratt

(2012).

As an example, consider a problem with K tasks, where each task is represented by $\mathbf{X}_k \in \mathbb{R}^{n_k \times d}$, where n_k is the number of samples and d is the dimension. The labels are represented by $\mathbf{Y}_k \in \mathbb{R}^{n_k \times 1}$. The goal is to estimate a simple linear model $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_K]$ such that $\mathbf{Y}_k = \mathbf{X}_k \times \mathbf{W}_k$. One of the standard procedures to model task relatedness is to assume $\mathbf{W}_k$ are close to one another. Regularized Multitask Learning by Evgeniou and Pontil (Evgeniou and Pontil (2004)) is a landmark work in modeling task relatedness. The authors incorporated task relatedness by assuming the $\mathbf{W}_k$ are close to each other. The optimization problem sought to minimize,

$$\min_{\mathbf{W}} \quad \frac{1}{K}\Big(\sum_{i}^{K} Loss(\mathbf{X}_k, \mathbf{W}_k, \mathbf{Y}_k) + \lambda \sum_{k}^{K} \big|\big|\mathbf{W}_k - \frac{1}{K}\sum_{k'}^{K} \mathbf{W}_{k'}\big|\big|_2^2\Big) \tag{2.2}$$

The first term is a standard loss term. The second term captures the inter-task relationship, where tasks are closely related to each other and their distance from the mean task is minimized. Although the model is very elegant, real world tasks need not be so closely related to one another. Improvements upon this very basic model, along with other procedures to perform multitask learning are outlined in the following works: Bakker and Heskes (2003); Evgeniou and Pontil (2007); Collobert and Weston (2008); Weinberger *et al.* (2009); Kang *et al.* (2011); Kumar and Daumé III (2012) and Gong *et al.* (2012b).

2. *Self-taught Learning*: This learning paradigm was introduced in Raina *et al.* (2007). The concept of learning is based on how humans learn in an unsupervised manner from unlabeled data. In this paradigm, the transfer of knowledge is from unrelated domains in the form of learned representations. Given unlabeled data, $\{\boldsymbol{x}_u^1, \ldots, \boldsymbol{x}_u^k\}$ where $\boldsymbol{x}_u^i \in \mathbb{R}^d$, the self-taught learning framework estimates a set of K basis vectors that are later used as a basis to represent the

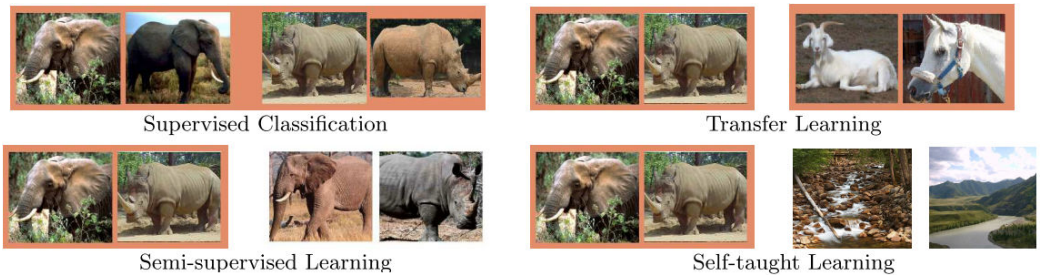


Figure 2.1: Different learning paradigms with labeled data in orange border. Supervised learning uses labeled examples, semi-supervised uses additional unlabeled examples, transfer learning uses additional labeled examples from different domain and self-taught learning uses unlabeled data to learn. (Image based on Raina *et al.* (2007)).

target data. Specifically,

$$\min \sum_i ||\boldsymbol{x}_u^i - \sum_j^K a_i^j \boldsymbol{b}_j||^2 + \beta||\boldsymbol{a}_i||_1$$
$$\text{s.t.}||\boldsymbol{b}_j|| \leq 1, \forall j \in 1, \ldots K \tag{2.3}$$

where, $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K\}$ are a set of basis vectors that are learned from unlabeled data and $\boldsymbol{b}_i \in \mathbb{R}^d$. For input data $\boldsymbol{x}_u^i$, the corresponding sparse representation is $\boldsymbol{a}_i = \{a_i^1, \ldots, a_i^K\}$ with a_i^j corresponding to the basis vector $\boldsymbol{b}_j$. The transfer of learning occurs when the same set of basis vectors $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K\}$ are used as a basis to represent labeled target data. Figure (2.1), provides an overview of the different learning paradigms compared with self-taught learning. Although the Figure (2.1) distinguishes transfer learning from self-taught learning, this discussion treats it as a special case of transfer learning. The unlabeled dataset (outdoor scene images in Figure (2.1)) can be considered as the source data set and the labeled dataset (elephants and rhinos in Figure (2.1)) can be treated as the target dataset. Some of the prominent machine learning and computer vision techniques that incorporate self-taught learning are Yang *et al.* (2009); Bengio (2009); Lee *et al.* (2009); Mairal *et al.* (2010).

3. *Sample Selection Bias*: The concept of sample selection bias was introduced in

Economics as a Nobel prize winning work by James Heckman in 1979 (Heckman (1979)). When the distribution of sampled data does not reflect the true distribution of the dataset it is sampled from, it is a case of sample selection bias. For example, a financial bank intends to model the profile of a loan defaulter in order to deny such defaulters a loan from the bank. It therefore builds a model based on the loan defaulters it has in its records. However, this is a small subset and therefore does not truthfully model the general public the bank wants to profile but does not have access to. Therefore, the defaulter profile generated by the bank can be considered to be offset by what is termed as the sample selection bias.

In this learning scenario, a dataset $\mathbf{X} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$ is made available. This dataset is used to estimate the joint distribution $\hat{P}(X, Y)$ which is an approximation for the true joint distribution $P(X, Y)$. However, $\hat{P}(X, Y) \neq P(X, Y)$ where $\hat{P}(X, Y)$ is the estimated distribution and $P(X, Y)$ is the true distribution (Bruzzone and Marconcini (2010)). This could be because of very few data samples, which could lead to a poor estimation of the prior distribution, $\hat{P}(X) \neq P(X)$. In other cases, when the training data does not represent the target (test) data, and introduces a bias in the class prior ($\hat{P}(Y) \neq P(Y)$), this eventually leads to incorrect estimation of the conditional ($\hat{P}(Y|X) \neq P(Y|X)$). When both the marginal ($\hat{P}(X) \neq P(X)$) and the conditionals are different ($\hat{P}(Y|X) \neq P(Y|X)$), the problem is referred to as sample selection bias Zadrozny (2004); Dudík *et al.* (2005); Huang *et al.* (2006). When only the marginals vary ($\hat{P}(X) \neq P(X)$) and the conditionals are approximately equal ($\hat{P}(Y|X) \approx P(Y|X)$), the problem is termed as covariate shift Shimodaira (2000); Quionero-Candela *et al.* (2009); Bickel *et al.* (2009); Gretton *et al.* (2009).

4. *Lifelong Machine Learning (LML)*: The concept of life long learning was discussed by Thrun in the seminal work Thrun (1996). The concept of transfer in life long learning can be formulated as follows. A machine learning model trained for K tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\}$ is updated by learning task $\mathcal{T}_{K+1}$ with data $\mathcal{D}_{K+1}$. The work discussed if learning the $K + 1^{th}$ task was easier than learning the first task. The key characteristics of life long learning are: (i) a continuous learning process, (ii) knowledge accumulation, and (iii) use of past knowledge to assist in future learning [1] Fei *et al.* (2016).

   Lifelong machine learning differs from multitask learning because it retains knowledge about previous tasks and applies that knowledge to learn new tasks. It also differs from standard domain adaptation which transfers knowledge to learn only one task (target). This germane concept of lifelong learning is closely related to the paradigm of *incremental learning* where a model is updated with new data to learn a new task. Lifelong machine learning can also be viewed as lifelong incremental learning. However, some incremental learners depend on data from previous tasks when learning a new task Mensink *et al.* (2013). Other approaches learn succinct data representations (also termed as exemplars) to model data from previous tasks and recall them when updating the classifier for a new task Rebuffi *et al.* (2017). In an uncompromising form of incremental learning, no data is used from previous tasks and the learner is updated using only the data from the new task Li and Hoiem (2016).

5. *One-shot Learning and Zero-shot Learning*: These can be viewed as extreme cases of transfer learning Goodfellow *et al.* (2016). Both these forms of transfer seek to learn data categories from minimal data. The key motivation is the

[1] `https://www.cs.uic.edu/~liub/Lifelong-Machine-Learning-Tutorial-KDD-2016.pdf`

ability to transfer knowledge from previously learned categories to recognize new categories. In one-shot learning, the model is trained to recognize a new category of data based on just one labeled example Fei-Fei *et al.* (2006). It relies on the ability of the model to learn representations that cleanly separate the underlying categories. The single labeled example from a new category is used to identify the cluster center around which other unseen examples of the same category are going to cluster. One-shot learning relies on the ability to discover representations that matter and those that do not in order to recognize categories.

Zero-shot learning is the ability to recognize new categories without having seen any example of the new category. Zero-data learning (Larochelle *et al.* (2008)) and zero-shot learning (Palatucci *et al.* (2009); Socher *et al.* (2013)) are examples where the model has learned to transfer knowledge from training data not completely related to the categories of interest. For example, a model that has been trained to recognize breeds of dogs can be provided with a description of the categories {fox, wolf, hyena, wild-dog}. Without having ever seen an image of any of these categories, the zero-data learning model can be trained to associate the textual description of the category to learn a model to recognize the new category.

6. *Domain Adaptation*: Domain adaptation is a special case of transfer learning where transfer of knowledge generally occurs between two domains, *source* and *target*. There are examples of multi-source domain adaptation as in Sun *et al.* (2011); Mansour *et al.* (2009); Chattopadhyay *et al.* (2012), although, two domain transfer is the popular setting. In domain adaptation, the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$ are not the same, and the goal is to solve a com-

mon task $\mathcal{T} = \{\mathcal{Y}, f(.)\}$. For example, in a image recognition task, the source domain could contain labeled images of objects against a white background and the target domain could consist of unlabeled images of objects against a noisy and cluttered background. Both the domains inherently have the same set of image categories. The difference between the domains is modeled as the variation in their joint probability distributions $P_S(X, Y) \neq P_T(X, Y)$. Standard domain adaptation assumes that there is plenty of labeled data in the source domain and there is no labeled data (or few samples, if any) in the target domain. Since there are no labeled samples (or very few, if any) of target data, it is difficult to get a good estimate of $\hat{P}_T(X, Y)$. The key task of domain adaptation lies in approximating $\hat{P}_T(X, Y)$ using the source data distribution estimation $\hat{P}_S(X, Y)$. This is possible because the two domains are 'correlated'. This correlation is often modeled as covariate shift where, $P_S(X) \neq P_T(X)$ and $P_S(Y|X) \approx P_T(Y|X)$. The goal of the domain adaptation problem is to minimize the expected prediction error on the target. If $h(\boldsymbol{x})$ is the hypothesis of interest, the expected error on the target distribution is given by [2],

$$\begin{aligned}
\epsilon_T(h) &= \mathbb{E}_{p_T(X=\boldsymbol{x})}\mathbb{E}_{p_{(Y=y|X=\boldsymbol{x})}}\big(h(\boldsymbol{x}) \neq y\big) \\
&= \sum_{\boldsymbol{x}} p_T(\boldsymbol{x})\mathbb{E}_{p_{(y|\boldsymbol{x})}}\big(h(\boldsymbol{x}) \neq y\big) \\
&\quad \text{Multiplying and dividing by } p_S(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}} \frac{p_S(\boldsymbol{x})}{p_S(\boldsymbol{x})} p_T(\boldsymbol{x})\mathbb{E}_{p_{(y|\boldsymbol{x})}}\big(h(\boldsymbol{x}) \neq y\big) \\
&\quad \text{The target error is equivalent to a weighted source error} \\
&\quad \text{with weights given by } \boldsymbol{w}, \\
\epsilon_T(h) &= \epsilon_S(h, \boldsymbol{w}) = \mathbb{E}_{p_S(\boldsymbol{x})}\frac{p_T(\boldsymbol{x})}{p_S(\boldsymbol{x})}\mathbb{E}_{p_{(y|\boldsymbol{x})}}\big(h(\boldsymbol{x}) \neq y\big) \qquad (2.4)
\end{aligned}$$

[2] `http://adaptationtutorial.blitzer.com/`

The above derivation uses the covariate shift assumption ($P_S(Y|X) \approx P_T(Y|X)$ and $P_S(X) \neq P_T(X)$) and the concept of shared support, i.e. ($P_S(X) = 0$ iff $P_T(X) = 0$). The weight for each source data point is $\frac{p_T(\boldsymbol{x})}{p_S(\boldsymbol{x})}$. Domain adaptation approaches that estimate the weights for source data points are called *instance weighting* techniques. Most domain adaptation algorithms estimate this weight using Kernel Mean Matching (Gretton *et al.* (2009)) or Kullback-Leibler divergence (Sugiyama *et al.* (2008)). Domain adaptation can also be viewed as a case of covariate shift or sample selection bias Quionero-Candela *et al.* (2009). Another approach to domain adaptation is *feature matching*, where a shared feature representation between the source and target is estimated. Examples of this technique are discussed in Pan *et al.* (2008, 2011); Long *et al.* (2013).

Other procedures learn feature subspaces that are common to the source and target datasets. They project the source and target dataset into that space and train a classifier on the source and expect it to work on the projected dataset Fernando *et al.* (2013); Gong *et al.* (2012a); Long *et al.* (2014); Hoffman *et al.* (2012). Recently, there has been work modeling the difference in class prior and conditional shift Zhang *et al.* (2013). All of the above procedures can be viewed as *fixed representation* approaches. In a fixed representation approach, the features are predetermined and fixed and domain adaptation is performed using these pre-determined features. In recent years deep learning approaches have outperformed fixed representation techniques in domain adaptation. Deep learning based domain adaptation learns to extract transferable feature representations using deep neural networks Tzeng *et al.* (2014); Long *et al.* (2015); Ganin *et al.* (2016). The following chapter provides a classification of domain adaptation procedures including recent approaches involving deep learning.

In addition to these types of transfer learning there are a couple of other learning approaches that have elements of knowledge transfer. The problem of *concept drift* occurs when the data changes its distribution gradually over time. Models in this case must adapt to the changing distribution while also transferring knowledge from previously seen data. *Multimodal learning* can also be viewed as an example of transfer learning where a relationship is captured between representations in multiple modalities in order to enhance learning Srivastava and Salakhutdinov (2012).

2.1.3 Types of Domain Adaptation

This subsection outlines the different approaches to posing problems in domain adaptation. A detailed discussion on different approaches to solving domain adaptation problems is described in the following chapter. This subsection merely lists the different ways in which a domain adaptation problem is posed. There are two standard domain adaptation problem statements.

1. *Supervised or Semi-supervised Domain Adaptation*: In this setting the source domain consists of labeled data $\mathcal{D}_s = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{n_s}$ and the target domain also consists of labeled data $\mathcal{D}_t = \{\boldsymbol{x}_i^t, y_i^t\}_{i=1}^{n_t} \cup \{\boldsymbol{x}_i^t\}_{i=n_t+1}^{n_t+n_u}$. There are n_t labeled target data points and n_u unlabeled target data points with $n_t \ll n_u$. However, it is not possible to estimate the joint distribution $P_T(X, Y)$ over the target because of limited number of target samples n_t, without the risk of overfitting. The source dataset has more labeled samples than the target with $n_t \ll n_s$ and it can be used to estimate the joint distribution $P_S(X, Y)$. Therefore, the source dataset $\mathcal{D}_s$ is used along with $\mathcal{D}_t$ to train a classifier for the target data as in Daumé III *et al.* (2010); Saenko *et al.* (2010); Hoffman *et al.* (2013) and Venkateswara *et al.* (2015b). When n_s and n_t are of similar size, the problem can also be viewed as a multi-task learning setup.

2. *Unsupervised Domain Adaptation*: This is by far the most standard and also the most challenging approach to domain adaptation. In this setting the source domain consists of labeled data $\mathcal{D}_s = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{n_s}$ and the target data consists of only unlabeled data $\mathcal{D}_t = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$. The task is to learn a classifier for the target data using the source dataset $\mathcal{D}_s$ and the target dataset $\mathcal{D}_t$. There is no restriction on the number of source and target samples. The source data can be used to estimate the joint distribution $P_S(X, Y)$. But the source data is adapted to the target using the unlabeled target data to approximate $P_T(X, Y)$ as in Gopalan *et al.* (2011); Gong *et al.* (2012a); Long *et al.* (2014) and Venkateswara *et al.* (2016).

Apart from these standard setups for domain adaptation, there is also multi-source domain adaptation where, as the name indicates, there are multiple source domains and one target domain as in Mansour *et al.* (2009); Chattopadhyay *et al.* (2012) and the multiple source domains are adapted to the target to estimate a classifier for the target.

2.2 Performance Bounds for Domain Adaptation

Generalization bounds give the probability that a function chosen from a hypothesis set achieves a certain error in a statistical learning model. Generalized learning bounds have been applied to evaluate the consistency of Empirical Risk Minimization (ERM) based learning methods Vapnik (2013). However, these learning bounds are based on the assumption that the test set is drawn from the same distribution as the training set. There have been attempts to adapt these generalization bounds for domain adaptation, where the test (target) set is from a different distribution than the training (source) set Ben-David *et al.* (2010); Mansour *et al.* (2009) and Zhang *et al.* (2012).

A binary classification task is considered with input space $\mathcal{X} \subseteq \mathbb{R}^d$ and label space $\mathcal{Y} = \{0, 1\}$. The source domain is denoted by $\mathcal{D}_S = \{(\mathcal{X} \times \mathcal{Y}), P_S(X, Y)\}$, where $P_S(X, Y)$ is the source joint distribution. Similarly, the target domain is denoted by $\mathcal{D}_T = \{(\mathcal{X} \times \mathcal{Y}), P_T(X, Y)\}$, where $P_T(X, Y)$ is the target joint distribution. The source dataset consists of labeled data $\mathcal{D}_s = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{n_s}$ and the target dataset is $\mathcal{D}_t = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$. The goal of domain adaptation is to learn a classifier $h : \mathcal{X} \to \mathcal{Y}$ for the target data with minimum risk of prediction,

$$\epsilon_T(h) = \Pr_{(\boldsymbol{x}^t, y^t) \sim \mathcal{D}_T} \big(h(\boldsymbol{x}^t) \neq y^t\big) \tag{2.5}$$

where, Pr(.) is probability.

2.2.1 Divergence Between Domains

The standard procedure for bounding the target error in terms of the source error and a factor measuring the discrepancy between the source and the target. This reasoning is based on the notion that the source error is a good substitute for the target error when the distributions are similar. The distance between the marginal distributions $P_S^{\mathcal{X}}$ and $P_T^{\mathcal{X}}$ is defined for a hypothesis class $\mathcal{H}$ and is termed as the $\mathcal{H}$-divergence Kifer *et al.* (2004).

$$d_{\mathcal{H}}(P_S^{\mathcal{X}}, P_T^{\mathcal{X}}) = 2 \sup_{h \in \mathcal{H}} \Big| \Pr_{\boldsymbol{x}^s \sim P_S^{\mathcal{X}}} \big(h(\boldsymbol{x}^s) = 1\big) - \Pr_{\boldsymbol{x}^t \sim P_T^{\mathcal{X}}} \big(h(\boldsymbol{x}^t) = 1\big) \Big|. \tag{2.6}$$

The $\mathcal{H}$-divergence is based on the ability of the hypothesis class $\mathcal{H}$ to distinguish between samples generated from $P_S^{\mathcal{X}}$ and $P_T^{\mathcal{X}}$. An empirical divergence can also be estimated based on samples $\mathcal{D}_s$ and $\mathcal{D}_t$ from the two domains Ben-David *et al.* (2010).

$$\hat{d}_{\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) = 2 \left(1 - \min_{h \in \mathcal{H}} \left[\frac{1}{n_s} \sum_{i=1}^{n_s} I[h(\boldsymbol{x}_i^s) = 1] + \frac{1}{n_t} \sum_{i=1}^{n_t} I[h(\boldsymbol{x}_i^t) = 0] \right] \right) \tag{2.7}$$

where, $I[c]$ is an indicator function which is 1 when the condition c is true, otherwise it is 0. The goal is to determine the best classifier that can differentiate between

the two domains. The $\mathcal{H}$-divergence is then represented in terms of the error of that classification.

2.2.2 Proxy Divergence Measure

To determine the best classifier may be difficult in practice. Even in the space of linear classifiers, Equation (2.7) may be intractable. Ben-David et al. outline a procedure to determine a proxy distance as a substitute for the $\mathcal{H}$-divergence Ben-David *et al.* (2010). The proxy-distance is based on the same principle of learning a classifier to distinguish between the two domains. The *Proxy $\mathcal{A}$-distance* is given by,

$$\hat{d}_{\mathcal{A}} = 2(1 - 2\epsilon) \tag{2.8}$$

where, ϵ is the average error using a linear classifier to distinguish between data points from the two domains. The proxy distance measure has been used to estimate the distances between pairs of datasets in domain adaptation experiments Glorot *et al.* (2011), Long *et al.* (2015) and Venkateswara *et al.* (2017b).

2.2.3 Generalization Bound on Target Risk

To estimate a generalization bound for the target error, a few definitions are first outlined. For a hypothesis function $h : \mathcal{X} \rightarrow \{0, 1\}$ and a marginal distribution $P_S^{\mathcal{X}}$, the probability that the hypothesis disagrees with the labeling function $f(.)$ is given by,

$$\epsilon_S(h, f) = \mathbb{E}_{\boldsymbol{x} \sim P_S^{\mathcal{X}}}[|h(\boldsymbol{x}) - f(\boldsymbol{x})|] \tag{2.9}$$

If $\mathcal{H}$ is a hypothesis class and $\mathcal{A}_{\mathcal{H}}$ is a set of subsets over $\mathcal{X}$, that are a support over the hypothesis set $\mathcal{H}$, i.e. $\forall h \in \mathcal{H}$, $\{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{X}, h(\boldsymbol{x}) = 1\} \in \mathcal{A}_{\mathcal{H}}$, then the distance between two distributions $P_S^{\mathcal{X}}$ and $P_T^{\mathcal{X}}$ can be defined according to Blitzer *et al.* (2008)

as:

$$d_{\mathcal{H}}(P_S^{\mathcal{X}}, P_T^{\mathcal{X}}) = 2 \sup_{A \in \mathcal{A}_{\mathcal{H}}} |\Pr_{P_S^{\mathcal{X}}}[A] - \Pr_{P_S^{\mathcal{X}}}[A]|. \tag{2.10}$$

Following the definition of distance in Equation (2.10), a symmetric difference hypothesis space is defined as,

$$\mathcal{H}\Delta\mathcal{H} = \{h(\boldsymbol{x}) \oplus h'(\boldsymbol{x}) : h, h' \in \mathcal{H}\}, \tag{2.11}$$

where the XOR operator $\oplus$ indicates that $\forall g \in \mathcal{H}\Delta\mathcal{H}$ which labels $\boldsymbol{x}$ as positive, $\exists h, h' \in \mathcal{H} : h(\boldsymbol{x}) \neq h'(\boldsymbol{x})$. Similarly, $\mathcal{A}_{\mathcal{H}\Delta\mathcal{H}}$ is defined as a set of all subsets A such that $A = \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{X}, h(\boldsymbol{x}) \neq h'(\boldsymbol{x})\}$ for some $h, h' \in \mathcal{H}$. Along with the definition of error probability in Equation (2.9), the definition of distance in Equation (2.10) and the outline of the symmetric difference hypothesis space in Equation (2.11), Ben-David *et al.* (2010) derive the distance $d_{\mathcal{H}\Delta\mathcal{H}}$ between two distributions as,

$$|\epsilon_S(h, h') - \epsilon_T(h, h')| \leq \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(P_S^{\mathcal{X}}, P_T^{\mathcal{X}}). \tag{2.12}$$

Blitzer et al. Blitzer *et al.* (2008) derive a target error bound for a pair of source and target datasets $\mathcal{D}_s$ and $\mathcal{D}_t$ with n data samples based on a hypothesis space $\mathcal{H}$ that has a VC-dimension d. With a probability of at least $1 - \delta$ (over the choice of samples), for every $h \in \mathcal{H}$,

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) + 4\sqrt{\frac{2d\log(2n) + \log\frac{4}{\delta}}{n}} + \lambda. \tag{2.13}$$

Here $\lambda = \min_{h \in \mathcal{H}} \epsilon_T(h) + \epsilon_S(h)$, the sum of source and target errors for the least error hypothesis. The bound guarantees that if the distance $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ between the domains is small (i.e., the domains are similar) and n is large, then the target error can be approximated by the source error. If the combined error of the ideal hypothesis is large, then there is no classifier that can be trained using the source data which will be a good target hypothesis. If λ is small (as it is usually for domain adaptation), then the key factor is the $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$ for computing target error.

2.3 Domain Adaptation in Computer Vision

This section outlines how domain adaptation research is currently being pursued. It brings to attention the drawbacks with current approaches and proposes changes to the field.

2.3.1 Research in Domain Adaptation

While the problem of variations in data coming from different distributions is outstanding, the solutions provided by domain adaptation models have not been easily applied to real world applications in computer vision. This can be attributed to the manner in which domain adaptation models are currently being developed and evaluated in the research community. Most of the proposed solutions in domain adaptation are based on models developed in the following environment: (i) Two different datasets to represent the source and the target domains. (ii) Source dataset with labeled data and target dataset with unlabeled data. (iii) The label space of the source and target being exactly the same. These restrictions limit the adoption of the proposed domain adaptation models to real world problems and confine them to the research community.

While these self-imposed restrictions help to formulate a well-defined domain adaptation problem, they do not reflect a real world setting. An environment for domain adaptation that is closer to a real world setting ought to be: (i) Two well defined domains (rather than datasets) to represent the source and the target. This would entail that the domain shift between the two domains is modeled. (ii) Both the source and target domains have labeled and unlabeled data. The target labeled data is essential to evaluate the performance of the model. Currently, domain adaptation models are evaluated using test data. (iii) There is no restriction on the label space

of the domains being exactly the same. One weak restriction could be an intersection of the label spaces of the source and target.

The definition of a domain is rather ambiguous when dealing with images. Unlike in audio signal processing and text data processing (NLP), domains in computer vision are defined by dataset. Images from different datasets are likely to belong to two different domains. This is due to the bias introduced by the data capture methods and the representation procedures for a dataset and not necessarily the data itself Torralba and Efros (2011). There has been some work in estimating domains by segregating data from multiple datasets into clusters Gong *et al.* (2013b). However, this has not been applied or extended by subsequent research in the area. A primary focus area for research in domain adaptation would be to develop comprehensive models for domain shift.

2.3.2 Computer Vision Datasets for Domain Adaptation

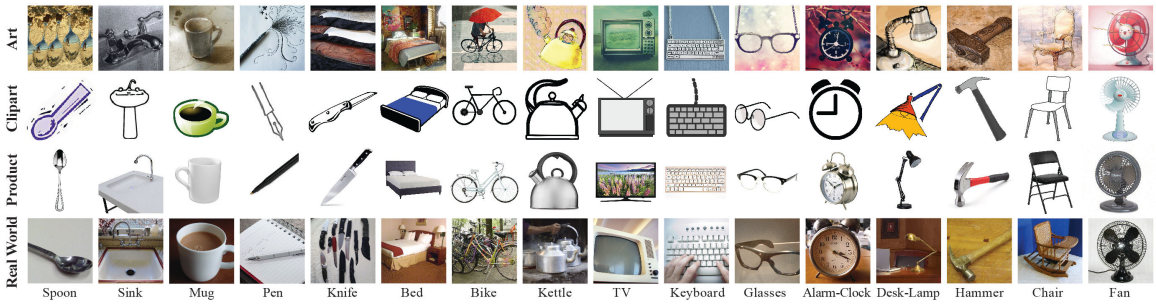


Figure 2.2: Sample images from the ***Office-Home*** dataset. The dataset consists of images of everyday objects organized into 4 domains; `Art`: paintings, sketches and/or artistic depictions, `Clipart`: clipart images, `Product`: images without background and `Real-World`: regular images captured with a camera. The figure displays examples from 16 of the 65 categories. Image based on Venkateswara *et al.* (2017b).

Evolution in datasets and the evolution of models for domain shift should complement each other. Current datasets for domain adaptation are not based on any models of domain shift. They are merely data samples coming from different sources

of data all with the same categories. The domain difference between these datasets is attributed to the 'bias' between the datasets, without a specific model characterizing the domain shift Torralba and Efros (2011). The domain adaptation procedures that are developed using these datasets can therefore be considered to be very generic. There is no guarantee on the performance of these procedures when applied to new problems. For e.g., if a domain adaptation approach were to be developed using the digit datasets (*USPS* and *MNIST* Jarrett *et al.* (2009)), there is no guarantee that this procedure would work well for a domain adaptation problem with medical images. On the other hand, if a dataset were to be created based on a domain shift model, then algorithms that are developed using this dataset can be applied to any domain adaptation problem where the same domain shift is observed. This is one primary reason for introducing new datasets for domain adaptation based on modeling domain shift.

Domain adaptation for vision based applications has generated great interest in the computer vision community in recent years Patel *et al.* (2015). Given the richness of their feature representations, deep learning based domain adaptation approaches like Tzeng *et al.* (2015a); Long *et al.* (2015); Ganin *et al.* (2016) have outperformed traditional shallow learning techniques Saenko *et al.* (2010); Pan *et al.* (2011); Gong *et al.* (2012a); Shekhar *et al.* (2013); Long *et al.* (2013); Fernando *et al.* (2013); Sun *et al.* (2015a). However, supervised deep learning models require a large volume of labeled training data. Unfortunately, existing datasets for vision-based domain adaptation are limited in their size and are not suitable for validating deep learning algorithms. In the absence of large datasets, domain adaptation algorithms are evaluated on datasets with few images.

The standard datasets for vision based domain adaptation are, facial expression datasets *CKPlus* (Lucey *et al.* (2010)) and *MMI* (Pantic *et al.* (2005)), digit datasets

SVHN (Netzer *et al.* (2011)), *USPS* and *MNIST* (Jarrett *et al.* (2009)), head pose recognition datasets *PIE* (Long *et al.* (2013)), object recognition datasets *COIL* (Long *et al.* (2013)), *Office* (Saenko *et al.* (2010)) and *Office-Caltech* (Gong *et al.* (2012a)). These datasets were created before deep-learning became popular and are insufficient for training and evaluating deep learning based domain adaptation approaches. For instance, the object-recognition dataset *Office* has 4110 images across 31 categories and *Office-Caltech* has 2533 images across 10 categories. A notable exception is the recently released *Cross-Modal Places* dataset with nearly 1 million images for indoor scene-recognition based on 5 different domains, viz., natural images, clip-art, sketches, text and spatial-text Castrejon *et al.* (2016).

One of the most popular datasets for computer vision is the *Office* dataset. Recent works in domain adaptation have also outlined some flaws with the dataset like label noise and lack of variation in object pose. They have used other datasets to evaluate their algorithms Bousmalis *et al.* (2017, 2016). Although the dataset lists 3 domains viz., `Amazon`, `DSLR`, and `Webcam`, it effectively has only 2 domains because the `DLSR` and `Webcam` domains are very similar with no domain discrepancy.

This also raises the question of what constitutes a domain in computer vision, for which there is no clear answer. Variations in vision based data can be attributed to the following reasons, viz., differences in image quality (resolution, brightness, occlusion, color), changes in camera perspective, dissimilar backgrounds and an inherent diversity of the samples themselves. Differences in any or all of these factors are what makes data from two domains dissimilar. A thorough evaluation of domain adaptation models can be done when testing with data that exhibits most of these variations. The existing datasets in domain adaptation for computer vision are very limited in the amount of variations in between the domains.

To address these limitations, a new dataset has been released as one of the con-

Table 2.1: Statistics for the *Office-Home* dataset. **Min: #** is the minimum number of images amongst all the categories, **Min: Size** and **Max: Size** are the minimum and maximum image sizes across all categories and **Acc.** is the classification accuracy.

Domain.	Min: #	Min: Size	Max: Size	Acc
Art	15	117×85 pix.	4384×2686 pix.	44.99±1.85
Clipart	39	18×18 pix.	2400×2400 pix.	53.95±1.45
Product	38	75×63 pix.	2560×2560 pix.	66.41±1.18
Real-World	23	88×80 pix.	6500×4900 pix.	59.70±1.04

tributions of this dissertation. The ***Office-Home*** is an object recognition dataset, that can be used to evaluate deep learning algorithms for domain adaptation. The *Office-Home* dataset consists of 4 domains, with each domain containing images from 65 categories of everyday objects and a total of around $15,500$ images. The domains include, `Art`: artistic depictions of objects in the form of sketches, paintings, ornamentation, etc.; `Clipart`: collection of clipart images; `Product`: images of objects without a background, akin to the Amazon category in *Office* dataset; `Real-World`: images of objects captured with a regular camera.

Public domain images were downloaded from websites like www.deviantart.com and www.flickr.com to create the `Art` and `Real-World` domains. `Clipart` images were gathered from multiple clipart websites. The `Product` domain images were exclusively collected from www.amazon.com using web-crawlers. The collected images were manually filtered on the basis of quality, size and content. The dataset has an average of around 70 images per category and a maximum of 99 images in a category. The primary challenge in creating this dataset was acquiring sufficient number of public domain images across all the 4 domains. Figure (2.2) depicts a sampling of 16 categories from the *Office-Home* dataset and Table (2.1) outlines some meta data for the dataset. The **Acc.** column in the Table (2.1) refers to classification accuracies using the LIBLINEAR SVM (Fan *et al.* (2008)) classifier (5-fold cross validation) with

deep features extracted using the VGG-F network Simonyan and Zisserman (2014). The dataset is made publicly available for research [3].

The domains in the *Office-Home* dataset include many of the variations that contribute to domain discrepancy. However, the *Office-Home* dataset is merely a stop-gap in the evolution of domain adaptation datasets. A more comprehensive understanding of domain discrepancy is necessary in order to create datasets for domain adaptation. An evolution in the nature of datasets will lead to more robust models for domain adaptation that can be applied to real world problems.

2.3.3 Deep Learning for Domain Adaptation

Most of the research in computer vision over the last few decades, has been driven by research in object recognition Oquab *et al.* (2014). From algorithms that recognize a few instance categories, to systems that can recognize thousands of categories of objects, computer vision has made rapid advances. The progress has been exponential in the last few years due to the following reasons:

- Deep neural network based architectures that extract features relevant to the problem using a deep hierarchical network. Training with deep massive networks having millions of parameters has been made possible by adapting graphical processing units (GPUs) for parallel processing of operations on data.

- Large labeled datasets with hundreds of object categories like Pascal VOC (Everingham *et al.* (2015)), Caltech256 (Griffin *et al.* (2007)) and more recently ImageNet (Deng *et al.* (2009)). Training large models with millions of parameters like deep neural networks requires large datasets in order to prevent over fitting.

[3] `https://hemanthdv.github.io/officehome-dataset/`

Both of these advancements complement each other. Training a deep neural network requires lots of data and to train a reliable recognition model for a large number of categories, requires a deep neural network that can extract task specific feature representations.

Prior to deep neural networks, object recognition was carried out in two stages, (i) feature extraction and (ii) recognition model. Feature extraction was implemented by using hand-crafted descriptors such as HOG, SIFT, SURF etc., to capture distinct patterns in an image, in order to estimate a vector using a bag-of-features representation. The work by Chatfield et al. provides an evaluation of the different techniques for encoding these feature descriptors Chatfield *et al.* (2011b). These vectors were then used to train object recognition models. Deep neural networks combine feature extraction and model creation into a single unit. The task-specific and highly discriminative features from deep models have been key to their successful adoption for multiple problems in computer vision.

In the last couple of years, domain adaptation models based on deep learning have outperformed methods based on hand-crafted features (also termed shallow methods). Deep architectures are a convenient tool to represent high level abstractions because of *composition*. Low-level representations are composed to represent a higher-level concept. Some of these low-level features may be common across different domains for a given task. Deep learning based architectures exploit the existing underlying common factors across multiple tasks Bengio *et al.* (2012).

Deep neural networks extract features at multiple layers and compose them in a hierarchical manner. The specificity and generality of these features with regards to the task varies between the layers. The work by Yosinski et al., captures the extant of generality and specificity of neurons in each layer Yosinski *et al.* (2014). Features from the lower layers of the network were demonstrated to be more transferable than

features from upper layers.

Deep learning systems like deep CNNs learn representations of data that capture underlying factors of variation between different tasks in a multi-task transfer learning setting. These representations also *disentangle* the factors of variation allowing for the transfer of knowledge between tasks Bengio *et al.* (2013). Therefore, deep learning methods are very successful for multitask learning and transfer learning based applications.

It is therefore reasonable to conclude that the future of domain adaptation lies in the successful adoption of deep learning techniques for transfer learning. All of these factors are crucial and need to be considered when applying deep neural networks for domain adaptation. The following chapter provides a summary of deep learning methods for domain adaptation, including the latest trend on adversarial networks.

Chapter 3

LITERATURE SURVEY

The dissertation approaches the problem of domain adaptation through the concept of feature spaces. The contributions of the dissertation are organized under linear, nonlinear and hierarchical feature based models. In consonance with the dissertation structure, the literature survey is also compiled under a similar construction. Apart from aligning with the dissertation organization, there are certain benefits to viewing domain adaptation in computer vision, through the lens of feature spaces.

- The wide range of research in the area is organized uniquely by the different survey compilations like Pan and Yang (2010); Beijbom (2012), Patel *et al.* (2015) and Csurka (2017). Each organization provides a salient perspective to domain adaptation from the view of domain alignment. This survey organization provides a unique and novel perspective to domain adaptation from the point of view of features, which promises new insights into the problem of domain adaptation.

- The advancements in computer vision can be traced to the progress of descriptors and feature representations. Beginning with hand-crafted features like HOG and SIFT Chatfield *et al.* (2011a), using linear classifiers like LIBLINEAR (Fan *et al.* (2008)), the state-of-the-art in computer vision currently is hierarchical feature spaces with highly nonlinear classifiers enabled by deep neural networks. Feature extraction and representation lies at the heart of computer vision problems like object recognition. It would therefore serve researchers well to view computer vision based domain adaptation from the point of view

of feature spaces.

The chapter is organized as follows. Section (3.1) outlines the mathematical notation for describing domain adaptation. The linear feature models are described in Section (3.2). The nonlinear feature models are summarized in Section (3.3). The latest research in domain adaptation is captured under hierarchical feature space models in Section (3.4). Sundry models that do not adhere to the above classification are outlined in Section (3.5).

3.1 Notation

The notation used in the chapter is defined as follows. The source dataset is represented by $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s} \subset \mathscr{S}$ and the target dataset is represented by $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t} \subset \mathscr{T}$. In matrix notation, the source data is given by $\mathbf{X}_S = [\boldsymbol{x}_1^s, \ldots, \boldsymbol{x}_{n_s}^s] \in \mathbb{R}^{d \times n_s}$ and the target data by $\mathbf{X}_T = [\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_{n_t}^t] \in \mathbb{R}^{d \times n_t}$. Similarly, the corresponding labels are represented by $Y_S = [y_1^s, \ldots, y_{n_s}^s]$ and $Y_T = [y_1^t, \ldots, y_{n_t}^t]$ for the source and target data respectively. The source and target data have the same dimensionality where, $\boldsymbol{x}_i^s$ and $\boldsymbol{x}_i^t \in \mathbb{R}^d$ and the label space for the two domains is identical, i.e., y_i^s and $y_i^t \in \{1, \ldots, C\}$. Additional terms are introduced for later use like, $\mathbf{X} \coloneqq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] = [\mathbf{X}_S, \mathbf{X}_T]$, with $n = n_s + n_t$. In the unsupervised domain adaptation setting, the target labels Y_T are unknown and the joint distributions of the two domains are different with, $P_S(X, Y) \neq P_T(X, Y)$.

3.2 Linear Feature Spaces for Domain Adaptation

This section outlines existing models for domain adaptation that are based on linear feature spaces. These include linear transformations of data to reduce domain disparity, adapting linear classifiers from one domain to another, linearly projecting data to a subspace to increase domain overlap, aligning moments of the data from

the domains using linear transformations and an assortment of random methods.

3.2.1 Linear Transformation Models

In this set of linear approaches, a linear transformation matrix is estimated to transform the source (or target) data so that it becomes similar in distribution to the target (or source) data. Saenko et al. in Saenko *et al.* (2010), developed one of the earliest transformation based approaches in domain adaptation for image classification. Their algorithm determines a transformation matrix $W \in \mathbb{R}^{d \times d}$. W is used to estimate the similarity between pairs of source and target data points by computing,

$$\text{sim}_W = \boldsymbol{x}^{s\top} W \boldsymbol{x}^t \tag{3.1}$$

The similarity can be viewed as a dot product between transformed source data $W^\top \boldsymbol{x}^s$ and the target data $\boldsymbol{x}^t$. W is estimated through an optimization problem which consists of constraints on the similarity between pairs of data points from the source and target along with a regularization factor on W. The minimization problem is given by,

$$\min_W \quad \text{tr}(W) - \log\,\det(W) + \lambda \sum_i c_i(\mathbf{X}_S^\top W \mathbf{X}_T) \tag{3.2}$$

The first two terms control the smoothness of W with regularization and the last term minimizes the error sum over all constraint pairs $c_i(.)$. The solution to W is obtained with information theoretic metric learning (ITML), Davis *et al.* (2007). A nonlinear version of this algorithm is outlined in Kulis *et al.* (2011).

Jhuo et al. in Jhuo *et al.* (2012), learn a transformation matrix $W \in \mathbb{R}^{d \times d}$, to transform the source $\mathbf{X}_S$ so that it can be expressed as a linear combination of target

data points. The minimization problem to arrive at W is given by,

$$\begin{aligned} \min_{W,Z,E} \quad & \text{rank}(Z) + \alpha||E||_{2,1} \\ \text{s.t.} \quad & W\mathbf{X}_S = \mathbf{X}_T Z + E, \\ & WW^\top = \mathbf{I}, \end{aligned} \tag{3.3}$$

where the first term penalizes the rank of the matrix $Z \in \mathbb{R}^{n_t \times n_s}$. This leads to a low rank structure for Z which essentially captures the reconstruction of related source data points. The $\ell_{2,1}$ norm encourages columns of error matrix E to be zero, which accounts for the outlier source data points. α is a weighting factor. The last constraint on W ensures that W is an orthonormal basis. The transformed source data is augmented to the target data. The authors also extend the algorithm to a multi-source domain setting.

The authors in Fernando *et al.* (2013), learn a transformation matrix $M \in \mathbb{R}^{k \times d}$ to align the source subspace with the target subspace. If $\mathbf{U}_S \in \mathbb{R}^{d \times k}$ is the k-dimensional subspace for the source data (i.e. $\mathbf{U}_S = PCA(\mathbf{X}_S, k)$ where PCA is Principal Component Analysis) and $\mathbf{U}_T \in \mathbb{R}^{d \times k}$ is the k-dimensional subspace for $\mathbf{X}_T$, the transformation matrix M is obtained by

$$\begin{aligned} \min_{M} \quad & ||\mathbf{U}_S M - \mathbf{U}_T||_F^2 \\ = \quad & ||\mathbf{U}_S^\top \mathbf{U}_S M - \mathbf{U}_S^\top \mathbf{U}_T||_F^2 \\ = \quad & ||M - \mathbf{U}_S^\top \mathbf{U}_T||_F^2, \end{aligned} \tag{3.4}$$

where, the last equation uses $\mathbf{U}_S^\top \mathbf{U}_S = \mathbf{I}$. The optimal solution is given by $M = \mathbf{U}_S^\top \mathbf{U}_T$. The transformed source data is obtained with $\hat{X}_S^\top \leftarrow X_S^\top \mathbf{U}_S \mathbf{U}_S^\top \mathbf{U}_T$ and the target data is transformed as in $\hat{X}_T^\top \leftarrow X_T^\top \mathbf{U}_T$. A 1-NN (Nearest Neighbour) classifier is trained with the transformed source data and tested on the transformed target data.

The last set of transformation based methods deal with graph laplacian based embedding of data to reduce domain disparity. In Yao *et al.* (2015), the authors consider a binary classification setting with source data $\mathbf{X}_S$, unlabeled target data $\mathbf{X}_{Tu}$ and labeled target data $\mathbf{X}_{Tl}$. The labels Y_S and Y_T belong to $\{-1, +1\}$. The algorithm learns a linear source classifier $f_s(\boldsymbol{x}) = \boldsymbol{w}_s^\top \mathbf{M}_S \boldsymbol{x} + b_s$ and a linear target classifier $f_t(\boldsymbol{x}) = \boldsymbol{w}_t^\top \mathbf{M}_T \boldsymbol{x} + b_t$. Here, $\{\boldsymbol{w}_s, \boldsymbol{w}_t\} \in \mathbb{R}^k$, $\{b_s, b_t\}$ are biases, and $\{\mathbf{M}_S, \mathbf{M}_T\} \in \mathbb{R}^{k \times d}$ are orthonormal transformation matrices. The optimization problem minimizes the classification error along with an embedding loss. The embedding loss ensures the labeled data are clustered according to category, i.e., the positive samples in the projected data are close together and the negative samples are close together. In addition, the unlabeled data is also projected to maintain its existing neighborhood relations. The algorithm incorporates *structural risk*, *structure preservation* and *manifold regularization* in one model to develop a semi-supervised domain-adaptive subspace-learning algorithm (SDASL).

3.2.2 Linear Max-Margin Models

This set of linear methods is based on adapting a linear Support Vector Machine (SVM) classifier from the source to the target. The domain adaptive SVM (DASVM) by Bruzzone and Marconcini (2010), is a unique linear SVM based solution for unsupervised domain adaptation. The DASVM initially trains a SVM classifier using the labeled source data. In successive iterations the decision boundary for the SVM is re-estimated as batches of unlabeled target data are added to the pool of labeled data and labeled source data is removed in batches. The SVM classifier is used to predict the labels (termed as 'semi-labels') for the unlabeled target data. At every iteration, a batch of unlabeled data that is closest to the SVM boundary within the SVM margins is chosen to be added to the pool of labeled data. Any semi-labeled

data that changes its labels across iterations is removed from the labeled set. Also, a batch of source data points that is far from the decision boundary and therefore does not affect the decision boundary, is removed from the labeled set. The algorithm converges when the number of unlabeled data points within the margin goes below a threshold. The authors also outline a circular cross validation strategy to validate model parameters for the SVM using unlabeled target data.

The authors in Yang *et al.* (2007a), adapt a SVM classifier trained on the source (auxillary) data to the target (primary) data. This is a supervised domain adaptation problem that has labeled data in the target domain. Let $\boldsymbol{w}_s$ be the source data SVM classifier, then the target data SVM classifier is modeled so that it is not very different from the source SVM with the following optimization problem,

$$\begin{aligned} \min_{\boldsymbol{w}_t} &= \frac{1}{2}||\boldsymbol{w}_t - \boldsymbol{w}_s||^2 + C\sum_{i=1}^{n_t}\xi \\ \text{subject to} \quad & \xi \geq 0, \quad y_i^t \boldsymbol{w}_y^\top \boldsymbol{x}_i^t \geq 1 - \xi \ \forall i \end{aligned} \tag{3.5}$$

The first term in the optimization framework is the regularization over the decision boundaries. The constraint ensures that the target classifier $\boldsymbol{w}_t$ is similar to the source classifier $\boldsymbol{w}_s$. In a closely related work, Yang *et al.* (2007b), the authors train a SVM classifier for the target data which is based on the source data SVM.

$$\begin{aligned} f_t(\boldsymbol{x}) &= f_s(\boldsymbol{x}) + \Delta f(\boldsymbol{x}) \\ &= \boldsymbol{w}_s^\top \boldsymbol{x} + \boldsymbol{w}_t^\top \boldsymbol{x} \end{aligned} \tag{3.6}$$

Both of the above works are also extended to the nonlinear setting by introducing kernels. In Aytar and Zisserman (2011), the authors extend the Adapt-SVM to a Projective Model Transfer SVM (PMT-SVM), by introducing a constant factor γ,

which controls the amount of transfer.

$$\min_{\boldsymbol{w}_t} = \frac{1}{2}||\boldsymbol{w}_t||^2 + \gamma||P\boldsymbol{w}_t||^2 + C\sum_{i=1}^{n_t} l(\boldsymbol{x}_i^t, y_i^t; \boldsymbol{w}_t)$$
$$\text{subject to} \quad \boldsymbol{w}_t^\top \boldsymbol{w}_s \geq 0. \tag{3.7}$$

Here $P = \mathbf{I} - \frac{\boldsymbol{w}_s \boldsymbol{w}_s^\top}{\boldsymbol{w}_s^\top \boldsymbol{w}_s}$ is the projection matrix and $l(.)$ is the hinge loss. $||P\boldsymbol{w}_t||^2 = ||\boldsymbol{w}_t||^2 \sin^2\theta$, where θ is the angle between $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$. The constraint confines $\boldsymbol{w}_t$ to the positive half-space of $\boldsymbol{w}_s$.

A technique that combines both linear transformations and classifier adaptation is presented in Hoffman *et al.* (2013). The Max Margin Domain Transform (MMDT) algorithm learns a transformation matrix W, that projects the target data to a subspace that makes it similar to the source data. Simultaneously, the model learns a SVM classifier $\{\boldsymbol{w}, b\}$, to classify both the source and target data. The minimization problem is given by,

$$\begin{aligned}
\min_{W,\boldsymbol{w},b} &= \frac{1}{2}||W||_F^2 + \frac{1}{2}||\boldsymbol{w}||^2 \\
\text{s.t.} \quad & y_i^s\left(\begin{bmatrix}\boldsymbol{x}_i^s \\ 1\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{w} \\ b\end{bmatrix}\right) \geq 1, \quad i \in \{1, \ldots, n_s\} \\
& y_i^t\left(\begin{bmatrix}\boldsymbol{x}_i^t \\ 1\end{bmatrix}^\top W^\top \begin{bmatrix}\boldsymbol{w} \\ b\end{bmatrix}\right) \geq 1, \quad i \in \{1, \ldots, n_t\}
\end{aligned} \tag{3.8}$$

The first and the second terms are the regularization on the transformation matrix W and the decision boundary $\boldsymbol{w}$, respectively. The constraints are the standard max-margin constraints for SVM, with the target data being transformed by W. The parameters W and $\{\boldsymbol{w}, b\}$ are estimated using alternating minimization strategies.

3.2.3 Linear Alignment of Moments

In these set of linear models, domain adaptation is achieved by alining the moments of the source and target data. One such algorithm is the Feature LeArning with

second Moment Matching (FLAMM) method in Jiang *et al.* (2015). In FLAMM, the authors derive a bound for the expected error difference between the source data and the target data for a common classifier $h(\boldsymbol{x})$. This error difference can be minimized by minimizing the difference between the second moments given by $M_S = \frac{1}{n_s}\mathbf{X}_S\mathbf{X}_S^\top$ and $M_T = \frac{1}{n_t}\mathbf{X}_T\mathbf{X}_T^\top$. If $M = M_S - M_T$, the goal is to learn a transformation matrix P such that the transformed data $P^\top\mathbf{X}$ has the domains aligned. This condition can be represented as a minimization over $\text{tr}(P^\top M^2 P)$. The minimization problem is given by,

$$\min_P ||\mathbf{X}^\top - \mathbf{X}^\top P||_F^2 + \gamma_1 \text{tr}(P^\top \Lambda P) + \gamma_2 \text{tr}(P^\top M^2 P) \tag{3.9}$$

Here, Λ is a diagonal matrix where $\Lambda_{ii} = X_i X_i^\top$, and X_i is the row i of $\mathbf{X}$. The first term is to ensure the projected data is similar to the target so as to avoid trivial solutions when the mismatch between source and target data is small. The solution to P is iteratively refined by updating $\mathbf{X} \leftarrow P^\top\mathbf{X}$ in every iteration and solving for Equation (3.9).

The CORrelation ALignment (CORAL) algorithm in Sun *et al.* (2015a), is another unsupervised domain adaptation method that seeks to match the second order statistics of the source and the target. In CORAL, the authors learn a transformation matrix to transform the source data so that the source and target covariance matrices are nearly identical. If the source data is transformed by matrix A, i.e., $\hat{\mathbf{X}}_S \leftarrow A\mathbf{X}_S$, the difference between the correlation matrices of the transformed source $\hat{C}_S$ and the target C_T, is given by,

$$\begin{aligned} &\min_A ||\hat{C}_S - C_T||_F^2 \\ =&\min_A ||A^\top C_S A - C_T||_F^2. \end{aligned} \tag{3.10}$$

Here C_S is the correlation of the original source data. The optimal solution to Equa-

tion (3.10) is given by,

$$A^* = (\mathbf{U}_S \Sigma_S^{+\frac{1}{2}} \mathbf{U}_S^\top)(\mathbf{U}_{T[1:r]} \Sigma_{T[1:r]}^{\frac{1}{2}} \mathbf{U}_{T[1:r]}^\top), \tag{3.11}$$

where, $C_S = \mathbf{U}_S \Sigma_S \mathbf{U}_S^\top$ is the singular value decomposition of C_S and $C_T = \mathbf{U}_T \Sigma_T \mathbf{U}_T^\top$ is the singular value decomposition of C_T. The value r is the minimum of the ranks of C_S and C_T. The authors interpret Equation (3.11) in the following manner. The first term within the parenthesis can be viewed as a source whitening matrix. The second term can be viewed as a target coloring matrix. It implies that the source is first whitened to remove source related covariance. It is then colored with the target related covariance so that the covariance becomes similar to the target. The authors also extend this work to a deep learning based CORAL model in Sun and Saenko (2016).

3.3 Nonlinear Feature Spaces for Domain Adaptation

This section describes the literature for domain adaptation in nonlinear feature spaces. Here, the domain disparity is reduced by projecting the data from the domains into high-dimensional nonlinear feature spaces using kernels. A standard procedure that is applied for nonlinear domain adaptation is Maximum Mean Discrepancy (MMD). A brief introduction to MMD is provided at the beginning of the section before proceeding to the different methods that apply MMD for domain adaptation.

Given data samples $\mathcal{D}_s$ and $\mathcal{D}_t$ from two distributions, the MMD measure estimates the distance between the distributions. Most existing measures require the assumption of certain parameters to estimate the distance between distributions, but not the MMD. The MMD is a nonparametric distance estimate designed by embedding the data into a Reproducing Kernel Hilbert Space (RKHS), Smola *et al.* (2007). The data is mapped to a high-dimensional (possibly infinite-dimensional) space de-

fined by $\Phi(\mathbf{X}) = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]$. $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ defines a mapping function and $\mathcal{H}$ is a RKHS. The dot product between the high-dimensional mapped vectors $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{y})$, is estimated by the kernel-trick. The dot product is given by the positive semi-definite (psd) kernel, $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$. The kernel $k(.)$ can be viewed as a similarity measure between $\boldsymbol{x}$ and $\boldsymbol{y}$. A standard kernel function is the Gaussian radial basis function (RBF), $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{||\boldsymbol{x}-\boldsymbol{y}||^2}{2\sigma^2})$. The similarity measure between all pairs of data points in $\mathbf{X}$, can represented using the kernel gram matrix which is given by, $\mathbf{K} = \Phi(\mathbf{X})^\top \Phi(\mathbf{X}) \in \mathbb{R}^{n \times n}$. Gretton et al. in Gretton *et al.* (2007), introduced the MMD to estimate the distance between $\mathcal{D}_s$ and $\mathcal{D}_t$, which is given by,

$$\text{MMD} = \left|\left| \frac{1}{n_s} \sum_{i=1}^{n_s} \phi(\boldsymbol{x}_i^s) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\boldsymbol{x}_j^t) \right|\right|_{\mathcal{H}}^2 \tag{3.12}$$

The distance between the two distributions is the distance between their means in the RKHS. When the RKHS is universal, the MMD measure approaches 0 only when the distributions are the same (Smola *et al.* (2007)). Many of the methods in the following subsections apply the MMD in different ways to achieve nonlinear domain adaptation.

3.3.1 Max Margin Kernel Methods

In an early adoption of the MMD for domain adaptation, Duan et al. in Duan *et al.* (2009), introduced a Domain Transfer SVM (DTSVM) model for video concept detection. The MMD distance between the source and target domains is represented as,

$$\begin{aligned} \text{MMD} &= \left|\left| \frac{1}{n_s} \sum_{i=1}^{n_s} \phi(\boldsymbol{x}_i^s) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\boldsymbol{x}_j^t) \right|\right|_{\mathcal{H}}^2 \\ &= ||\Phi(\mathbf{X})\boldsymbol{m}||^2 = \text{tr}(\Phi(\mathbf{X})^\top \Phi(\mathbf{X})\mathbf{M}) = \text{tr}(\mathbf{KM}), \end{aligned} \tag{3.13}$$

where, $\boldsymbol{m} \in \mathbb{R}^n$ is a vector with its first n_s values as $1/n_s$ and the last n_t values as $-1/n_t$. The matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ is given by $\mathbf{M} = \boldsymbol{m}\boldsymbol{m}^\top$. $\mathbf{K}$ and $\Phi(\mathbf{X})$ have been defined at the beginning of the section. Along with the SVM component, the DTSVM is given by,

$$\min_{\mathbf{K} \succeq 0} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \frac{1}{2}\text{tr}(\mathbf{K}\mathbf{M})^2 + \lambda\Big(\boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}(\boldsymbol{\alpha} \circ \boldsymbol{y})^\top \mathbf{K}^L (\boldsymbol{\alpha} \circ \boldsymbol{y})\Big) \tag{3.14}$$

The second term is the SVM dual formulation for the labeled data where $\mathbf{K}^L$ is the kernel over labeled data and $\boldsymbol{\alpha}$ is the SVM vector of dual variables for the labeled data. This is a semi-definite programming problem with $\mathbf{K} \succeq 0$. The SVM related constraint on $\boldsymbol{\alpha}$ is $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^{n_l} | C\mathbf{1} \geq \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\alpha}^\top \boldsymbol{y} = 0\}$, where n_l is the number of labeled data points from source and target and C is the SVM constant. The authors adopt a multiple kernel formulation for $\mathbf{K}$ where, $k(.) = \sum_{i=1}^{M} d_m k_m$, with $d_m \geq 0$ and $\sum_{m=1}^{M} d_m = 1$ and k_m for $m \in \{1, \ldots, M\}$, are predefined positive semi-definite (psd) kernels. The problem now reduces to estimating coefficients $\boldsymbol{d} = [d_1, \ldots, d_M]^\top$ and $\boldsymbol{\alpha}$. The min-max problem is solved to reach a saddle point through alternating optimization for $\boldsymbol{d}$ and $\boldsymbol{\alpha}$. The same idea is elaborated using different loss functions along with a study on complexity in Duan *et al.* (2012).

In the Selective Transfer Matching (STM) algorithm Chu *et al.* (2013), the authors outline a novel SVM technique for domain adaptation for facial action unit recognition. STM performs *instance selection*, where a set of source data points are selected that have a distribution similar to the target and these are used to train a SVM. The nonlinear SVM component in the primal form is given by,

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}_S \boldsymbol{\beta} + C\sum_{i=1}^{n_s} s_i.l(y_i, \mathbf{K}_{S_i}^\top \boldsymbol{\beta}) \tag{3.15}$$

In the above equation $\mathbf{K}_S$, is the kernel matrix for the source (training data). $\boldsymbol{\beta}$ are the coefficients for the SVM decision boundary and the regularization over the

decision boundary $||\boldsymbol{w}||^2$ is given by the first term. This is the result of the representer theorem Kimeldorf and Wahba (1970), where the SVM solution can be expressed as $f(\boldsymbol{x}) = \sum_{i=1}^{n_s} \beta_i k(\boldsymbol{x}_i^s, \boldsymbol{x})$ and $\beta_i \neq 0$ for support vectors only. $l(.)$ is the hinge loss with $l(y, .) = \max(0, 1 - y.)$ and $\mathbf{K}_{Si}$ is the i^{th} row of $\mathbf{K}_S$. Each of the source data points has an importance (weight) associated with it which is given by s_i. The loss value from each of the source data points is weighted by s_i. The weights s_i, $i = 1, \ldots, n_s$, are estimated using MMD in the following quadratic programming problem,

$$\left|\left| \frac{1}{n_s} \sum_{i=1}^{n_s} s_i \phi(\boldsymbol{x}_i^s) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\boldsymbol{x}_j^t) \right|\right|_{\mathcal{H}}^2. \tag{3.16}$$

In order to simplify the equation, $\kappa_i := \frac{n_s}{n_t} \sum_{j=1}^{n_t} k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^t)$, $i = 1, \ldots, n_s$ and $\mathbf{K}_{S_{ij}} = k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^s)$ are defined. The minimization can now be represented as a quadratic programming problem,

$$\begin{aligned} \min_{\boldsymbol{s}} &= \frac{1}{2} \boldsymbol{s}^\top \mathbf{K}_S \boldsymbol{s} - \kappa^\top \boldsymbol{s}, \\ &\text{s.t. } s_i \in [0, B], \ \left| \sum_{i=1}^{n_s} s_i - n_s \right| \leq n_s \epsilon. \end{aligned} \tag{3.17}$$

In the first constraint, the scope of discrepancy between source and target distributions is limited, with $B \to 1$, leading to an unweighted solution. The second constraint ensures that $w(x)P_S(x)$, remains a probability distribution (Gretton *et al.* (2009)). Equations (3.15) and (3.17) are biconvex (convex when either of $\boldsymbol{s}$ or $\boldsymbol{\beta}$ is fixed). The solution is arrived at using alternate optimization methods.

3.3.2 MMD - Instance Weighting and Selection Methods

Instance selection in domain adaptation deals with sampling source data points whose distribution is similar to the target. This approach is based on the reasoning that although the source and target data are from different distributions, there could be some overlap between the two distributions. MMD is applied to identify data points in the source whose distribution is similar to the target data.

The Joint Optimization based Transfer and Active Learning (JO-TAL), proposes a single framework to perform transfer and active learning by solving a convex optimization problem in Chattopadhyay *et al.* (2013). To implement transfer learning, the framework uses the MMD (Borgwardt *et al.* (2006)) measure to weight the source data points $\boldsymbol{x}^s \in \mathcal{D}_s$, such that their distribution is similar to the unlabeled target dataset $\mathbf{X}_{Tu}$ (unlabeled data points from the target). The authors included active learning into the MMD measure by sampling a block of b data points from $\mathbf{X}_{Tu}$. These are the batch of unlabeled data points that will be queried for labels (active learning). The model also accounts for labeled target data (if any) with dataset $\mathbf{X}_{Tl}$. The minimization problem is given by,

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \left|\left| \frac{1}{n_s + n_l + b} \left(\sum_{i \in \mathcal{D}_s} \beta_i \phi(\boldsymbol{x}_i^s) + \sum_{j \in \mathbf{X}_{Tl}} \phi(\boldsymbol{x}_j^t) + \sum_{k \in \mathbf{X}_{Tu}} \alpha_k \phi(\boldsymbol{x}_k^t) \right) \right.\right. \\ & \left.\left. - \frac{1}{n_u - b} \sum_{k \in \mathbf{X}_{Tu}} (1 - \alpha_k) \phi(\boldsymbol{x}_k^t) \right|\right|_{\mathcal{H}}^2 \\ \text{s.t.} \quad & \alpha_i \in \{0, 1\},\ \beta_i \in [0, 1],\ \boldsymbol{\alpha}^\top \mathbf{1} = b \end{aligned} \tag{3.18}$$

n_s is the number of source data points in $\mathcal{D}_s$, n_l is the number of labeled target data points in $\mathbf{X}_{Tl}$, b is the batch size, n_u is the number of unlabeled target data points in $\mathbf{X}_{Tu}$, $n_t = n_u + n_l$, and $\mathbf{1}$ is a vector of ones. β_i are the weights for the source data points and $\boldsymbol{\alpha}$ is a binary vector that samples b unlabeled target points for query. All selected points will have $\alpha_i = 1$. Equation (3.18) can be written as,

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}_S \boldsymbol{\beta} + \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{K}_U \boldsymbol{\alpha} + \boldsymbol{\beta}^\top \mathbf{K}_{SU} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{K}_U \mathbf{1} - \boldsymbol{\beta}^\top \mathbf{K}_{SU} \mathbf{1} \\ & + \boldsymbol{\alpha}^\top \mathbf{K}_{UL} \mathbf{1} + \boldsymbol{\beta}^\top \mathbf{K}_{SL} \mathbf{1} + \text{const} \\ \text{s.t.} \quad & \alpha_i \in \{0, 1\},\ \beta_i \in [0, 1],\ \boldsymbol{\alpha}^\top \mathbf{1} = b \end{aligned} \tag{3.19}$$

The $\mathbf{K}$ terms are the kernels over the source (S), the labeled target (L) and the unlabeled target (U). The last four terms excluding the constant are linear in $\boldsymbol{\alpha}$ and

$\boldsymbol{\beta}$. Equation (3.19) can be expressed as a Quadratic Programming problem in a single variable $\boldsymbol{x} = [\boldsymbol{\alpha}^\top, \boldsymbol{\beta}^\top]^\top$. The constraint on $\boldsymbol{\alpha}$ makes the problem NP-hard and it can be relaxed to $\alpha_i \in [0, 1]$ and the b highest values are chosen as the active learning batch. The formulation is very elegant and accounts for all the desirable properties of transfer and active learning, namely, representativeness, diversity, and minimum redundancy.

In another example of instance selection, Gong et al. in Gong *et al.* (2013a) identify landmarks from the source dataset using multiple kernels. The MMD formulation is given by,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} & \left\| \frac{1}{\sum_i^{n_s} \alpha_i} \sum_i^{n_s} \alpha_i \phi(\boldsymbol{x}_i^s) - \frac{1}{n_t} \sum_j^{n_t} \phi(\boldsymbol{x}_j^t) \right\|_{\mathcal{H}}^2 \\ \text{s.t.} & \frac{1}{\sum_i \alpha_i} \sum_i \alpha_i y_{ic}^s = \frac{1}{n_s} \sum_i y_{ic}^s \\ & \boldsymbol{\alpha} \in \{0, 1\}^{n_s}, \end{aligned} \tag{3.20}$$

where, y_{ic}^s indicates $y_i^s = c$. The nonzero values of $\boldsymbol{\alpha}$ are the indices of the sampled source data points. The first constraint ensures class balance in the instance selection. The second condition makes the problem a hard combinatorial problem. It is relaxed by introducing variable $\beta_i = \alpha_i (\sum_i \alpha_i)^{-1}$. Unlike $\boldsymbol{\alpha}$ which is a binary variable, $\boldsymbol{\beta} \in [0, 1]^{n_s}$ is continuous. The geodesic flow kernel is used estimate the kernel with,

$$\begin{aligned} k(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) \\ &= \exp\Big(- \frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathbf{G} (\boldsymbol{x}_i - \boldsymbol{x}_j)}{\sigma^2} \Big), \end{aligned} \tag{3.21}$$

where $\mathbf{G}$, is determined using subspaces $\mathbf{U}_S$ and $\mathbf{U}_T$ of the source and target that are estimated using PCA (Gong *et al.* (2012a)). The geodesic flow kernel will be discussed towards the end of the chapter. The authors select landmarks at multiple granularity by defining different kernels with $\{\sigma_q \in [\sigma_{min}, \sigma_{max}]\}_{q=1}^{Q}$. The domain

invariant mapping is represented as $\phi_q(\boldsymbol{x}) = \sqrt{\mathbf{G}_q}\boldsymbol{x}$. These domain invariant vectors from multiple kernels are concatenated into a super vector $\boldsymbol{f}$ and cast into a multiple kernel SVM model in order to learn a domain adaptive classifier.

3.3.3 MMD - Spectral Methods

In this class of nonlinear methods, domain adaptation is achieved by nonlinear projection of the data where the projection matrix is a solution to an eigen-value problem. All of the methods in this subsection model the domain adaptation problem using kernel-PCA. In kernel-PCA a coefficient matrix $\mathbf{A} \in \mathbb{R}^{n\times k}$ is determined and the nonlinear projection of $\mathbf{X}$ is obtained using $\mathbf{A}$. The projection matrix is obtained by solving,

$$\max_{\mathbf{A}^\top\mathbf{A}=\mathbf{I}} \operatorname{tr}(\mathbf{A}^\top\mathbf{K}\mathbf{H}\mathbf{K}^\top\mathbf{A}). \tag{3.22}$$

Here, $\mathbf{H}$ is the $n\times n$ centering matrix given by $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}$, $\mathbf{I}$ is an identity matrix and $\mathbf{1}$ is a $n \times n$ matrix of 1s. The matrix of coefficients is $\mathbf{A} \in \mathbb{R}^{n\times k}$ and the nonlinear projected data is given by $\mathbf{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n] = \mathbf{A}^\top\mathbf{K} \in \mathbb{R}^{k\times n}$. In order to account for domian alignment after the projection, MMD can be incorporated as follows:

$$\min_{\mathbf{A}} \left|\left| \frac{1}{n_s}\sum_{i=1}^{n_s} \mathbf{A}^\top\boldsymbol{k}_i - \frac{1}{n_t}\sum_{j=n_s+1}^{n} \mathbf{A}^\top\boldsymbol{k}_j \right|\right|_{\mathcal{H}}^2 = \operatorname{tr}(\mathbf{A}^\top\mathbf{K}\mathbf{M}_0\mathbf{K}^\top\mathbf{A}), \tag{3.23}$$

where, $\mathbf{M}_0$, is the MMD matrix which given by,

$$(\mathbf{M}_0)_{ij} = \begin{cases} \frac{1}{n_s n_s}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_s \\ \frac{1}{n_t n_t}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_t \\ \frac{-1}{n_s n_t}, & \text{otherwise,} \end{cases} \tag{3.24}$$

Maximizing Equation (3.22) and minimizing Equation (3.23) is achieved with the following optimization problem,

$$\min_{\mathbf{A}^\top\mathbf{K}\mathbf{H}\mathbf{K}^\top\mathbf{A}=\mathbf{I}} \operatorname{tr}(\mathbf{A}^\top\mathbf{K}\mathbf{M}_0\mathbf{K}^\top\mathbf{A}) + \gamma||\mathbf{A}||_F^2, \tag{3.25}$$

where, the final term is the regularization over $\mathbf{A}$ to ensure smoothness. The solution to Equation (3.25) is a generalized eigen-value problem given by,

$$\Big(\mathbf{K}\mathbf{M}_0\mathbf{K}^\top + \gamma\mathbf{I}\Big)\mathbf{A} = \mathbf{K}\mathbf{D}\mathbf{K}^\top\mathbf{A}\mathbf{\Lambda}. \tag{3.26}$$

where the eigen-values are the Lagrangian constants captured in the diagonal matrix $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_k)$. The coefficient matrix $\mathbf{A}$ is given by the k-smallest eigen-vectors of Equation (3.26). The domain-aligned data points are then given by $\mathbf{Z} = \mathbf{A}^\top\mathbf{K}$. This is the Transfer Component Analysis (TCA) model described in Pan *et al.* (2011) although using a different derivation and notation.

The authors in Long *et al.* (2014), extend the TCA to incorporate instance sampling with a sparse norm over the coefficient matrix $\mathbf{A}$. The optimization problem with these constraints is given by,

$$\min_{\mathbf{A}^\top\mathbf{K}\mathbf{H}\mathbf{K}^\top\mathbf{A}=\mathbf{I}} \mathrm{tr}(\mathbf{A}^\top\mathbf{K}\mathbf{M}_0\mathbf{K}^\top\mathbf{A}) + \gamma\Big(||\mathbf{A}_s||_{2,1} + ||\mathbf{A}_t||_F^2\Big), \tag{3.27}$$

where, $\mathbf{A}_s \coloneqq \mathbf{A}_{1:n_s,:}$ is the transformation matrix for the source instances and $\mathbf{A}_t \coloneqq \mathbf{A}_{n_s+1:n,:}$ is the transformation matrix for the target. The $\ell_{2,1}$-norm regularization ensures row-sparsity of $\mathbf{A}_s$ effectively doing instance selection of the source data points. Similar to the TCA, Equation (3.27) is transformed to a generalized eigen-value problem and $\mathbf{A}$ is estimated.

The TCA and the TJM can be viewed as aligning the marginal probabilities $P_S(X)$ and $P_T(X)$ of the source and the target data. To ensure the joint distributions ($P_S(X, Y)$ and $P_T(X, Y)$) of the source and target are aligned, the Joint Distribution Adaptation (JDA) (Long *et al.* (2013)), algorithm is adopted, which seeks to align both the marginal and conditional probability distributions of the projected data. The marginal distributions are aligned as in Equation (3.23). The conditional distribution difference can also be minimized by introducing matrices M_c, with $c = 1, \ldots, C$,

defined as,

$$(\mathbf{M}_c)_{ij} = \begin{cases} \frac{1}{n_s^{(c)} n_s^{(c)}}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_s^{(c)} \\ \frac{1}{n_t^{(c)} n_t^{(c)}}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_t^{(c)} \\ \frac{-1}{n_s^{(c)} n_t^{(c)}}, & \begin{cases} \boldsymbol{x}_i \in \mathcal{D}_s^{(c)}, \boldsymbol{x}_j \in \mathcal{D}_t^{(c)} \\ \boldsymbol{x}_j \in \mathcal{D}_s^{(c)}, \boldsymbol{x}_i \in \mathcal{D}_t^{(c)} \end{cases} \\ 0, & \text{otherwise.} \end{cases} \tag{3.28}$$

Here, $\mathcal{D}_s$ and $\mathcal{D}_t$ are the subsets of source and target data points respectively. $\mathcal{D}_s^{(c)}$ is the subset of source data points whose class label is c and $n_s^{(c)} = |\mathcal{D}_s^{(c)}|$. Similarly, $\mathcal{D}_t^{(c)}$ is the subset of target data points whose class label is c and $n_t^{(c)} = |\mathcal{D}_t^{(c)}|$. For the target data, since the labels are not known, the predicted target labels are used to determine $\mathcal{D}_t^{(c)}$. The target data labels are initialized using a classifier trained on the source data and refined over iterations. Incorporating both the conditional and marginal distribution alignments, the JDA optimization problem is written as,

$$\min_{\mathbf{A}^\top \mathbf{K} \mathbf{H} \mathbf{K}^\top \mathbf{A} = \mathbf{I}} \text{tr}(\mathbf{A}^\top \mathbf{K} \sum_{c=0}^{C} \mathbf{M}_c \mathbf{K}^\top \mathbf{A}) + \gamma ||\mathbf{A}||_F^2. \tag{3.29}$$

Similar to the TCA and TJM, Equation (3.29) is transformed to a generalized eigenvalue problem and $\mathbf{A}$ is estimated. The domain-aligned data points are then given by $\mathbf{Z} = \mathbf{A}^\top \mathbf{K}$.

3.4 Hierarchical Feature Spaces for Domain Adaptation

In recent years, deep neural networks have revolutionized the field of machine learning. Deep learning based domain adaptation has outperformed non-deep learning algorithms because of the highly discriminatory nature of the features extracted using deep neural networks. The features extracted using deep neural networks are termed 'hierarchical' due to the hierarchical nature of the model and the nonlinear multilayer

structure of the network. This section describes recent work in the last few years in the area of deep learning based domain adaptation.

3.4.1 Naïve Deep Methods

Deep convolutional neural networks (CNN)s have been shown to be very good feature extractors. Deep CNNs trained on millions of images are by themselves very good feature extractors, not just for the dataset they are trained on, but for any generic image. Razavian et al. in Razavian *et al.* (2014), have demonstrated how a deep CNN trained on the ILSVRC 2013 ImageNet dataset (ImageNet (2013)) can be used for extracting generic features for any image. Regular SVMs trained on these generic features have shown astounding results across multiple applications like, scene recognition, fine grained recognition, attribute recognition and image retrieval. A pre-trained CNN can be used to extract generic features for the source and the target. This can be termed as a naïve form of domain adaptation.

Pre-trained deep neural networks can also be fine tuned to the task at hand. It is well documented that the lower layers of a CNN extract generic features that are common across multiple tasks and the upper layers extract task specific features. Features transition from general to specific by the last layer of the CNN. The work by Yosinski et al. in Yosinski *et al.* (2014), captures the extant of *generality* and *specificity* of neurons in each layer. Transferability has been shown to be negatively affected by two issues: (i) the specificity of neurons (to the source task) in the higher layers, adversely affects transfer to the target task, (ii) the fragile nature of dependencies between layers that are task specific. Adding new layers to a pre-trained (trained on source data) network and retraining it with target data, is another intuitive method to transfer knowledge in a deep learning setting. When the entire newly adapted network is fine tuned with target data, it can lead to a very efficient

adaptation. This form of adaptation has been explored in Oquab *et al.* (2014). The authors demonstrate a procedure to reuse the layers trained on the ImageNet dataset to compute mid-level representations for images. Despite the differences in image statistics, these features lead to improved results for object and action classification for different datasets.

The authors in Donahue *et al.* (2014), study the features extracted from the final layers of a deep neural network for a fixed set of object classification and detection tasks. The generic features from the 5th, 6th and 7th fully connected layers of an AlexNet (Krizhevsky *et al.* (2012)), show remarkable adaptation properties and outperform state-of-the-art methods in classification and detection. Whereas Donahue *et al.* (2014) studied adaptation using CNNs, Glorot *et al.* (2011) studied adaptation of features extracted using stacked denoising autoencoders for text based sentiment classification.

3.4.2 Adopted Shallow Methods

These set of deep learning methods adopt shallow (non-deep learning) domain adaptation procedures in a deep neural network. In these approaches the features extracted by the layers of the deep network are learned to be domain invariant. Domain alignment is achieved either through MMD, moment alignment or a loss function that drives the source and target classifiers to be indistinguishable. In discussing these methods, the central idea is outlined leaving out the details of network architecture, optimization procedures, loss functions etc.

In Tzeng *et al.* (2014), the authors adapt an AlexNet (Krizhevsky *et al.* (2012)) to output domain invariant features in the final fully connected $fc8$ layer in the Deep Domain Confusion (DDC) algorithm. The network has two loss components, (i) softmax classification loss for the source data points and (ii) domain confusion loss.

The network minimizes a MMD loss over the source and target data outputs of the $fc8$ layer in every mini-batch during training. This is termed as the domain confusion loss. A related idea is studied in Tzeng *et al.* (2015b), where the network has a domain confusion loss along with a domain classification loss. The domain classification loss ensures the output feature representations of the source and target data are distinct. This is in contrast to the goal of the domain confusion loss which tries to learn domain invariant representations. The network is trained to alternately minimize the two losses and reach a equilibrium. Long et al. introduce the Deep Adaptation Networks (DAN) model (Long *et al.* (2015)), which extends the concept of domain confusion by incorporating a MMD loss for all the fully connected layers ($fc6$, $fc7$ and $fc8$) of the AlexNet. The MMD loss is estimated for the feature representations over every mini-batch during training. The work also introduces MMD estimation computed with an efficient linear complexity based on Gretton *et al.* (2012). The linear MMD estimation is also unbiased because the MMD for the entire source and target data can be expressed as the sum of MMD across min-batches.

An extension to DAN is achieved with the Residual Transfer Network (RTN) in Long *et al.* (2016b), which implements a residual layer as the final layer of the network in addition to the softmax loss. In the RTN, the feature adaptation is achieved with MMD loss and the source and target classifier adaptation is implemented through the residual layer (He *et al.* (2016)). The source classifier $f_S(\boldsymbol{x})$ is tightly coupled with the target classifier $f_T(\boldsymbol{x})$ varying with only a slight perturbation $\Delta f(f_T(\boldsymbol{x}))$ which is learned by the network,

$$f_S(\boldsymbol{x}) = f_T(\boldsymbol{x}) + \Delta f(f_T(\boldsymbol{x})) \tag{3.30}$$

In addition, the source classifier is constrained by the softmax loss over the source data and the target classifier is constrained with unlabeled entropy loss over the target

data.

Compacting deep neural networks and reducing the number of parameters is essential for creating smaller more manageable networks. These procedures usually replace the larger convolutional layer kernels with kernels of size 1×1 and 3×3. Although such procedures produce networks that maintain the classification accuracies, the authors Wu et al. Wu *et al.* (2017), note that the adaptability of these networks is adversely affected resulting in low accuracies for domain adaptation. Wu et al. propose a set of layers called Conv-M, which consists of multi-scale convolution and deconvolution with kernels larger than 3×3 . The proposed compact network also uses MMD to align the source and target features at multiple layers and produces state-of-the-art results on the standard *Office* and *Office-Caltech* datasets. In addition, the network is guided with a reconstruction loss that tries to reconstruct images the encoded feature representations. The Domain reconstruction and Classification Network (DRCN) developed by Ghifary et al. Ghifary *et al.* (2016), is also guided by a reconstruction loss that decodes the feature encoding along with a standard classification loss.

While the MMD is a standard non-parametric measure used to align the features of the domains, Koniusz et al. Koniusz *et al.* (2017) propose a technique to align the higher order statistics of the features. The scatter statistics of samples belonging to a class (within-class) are aligned across the two domains. These include the means, scale/shear and the orientation measures of samples belonging to a single class. The procedure also maintains good separation for between-class scatters to enhance classification accuracies. Unlike the popular unsupervised setting, this deep learning technique is trained using a few labeled data from the target domain.

In all the above deep domain adaptation approaches, the weights are shared between the source and the target network to ensure domain invariant features. The

authors in Rozantsev *et al.* (2016), argue that merely ensuring domain invariant features may be detrimental to the discriminative power of the features. Their model is a twin network (one for the source and another for the target) with a loss function over the weights for every source target layer pair. The loss term ensures the weights of the source and the target are closely related (but not identical). The source network is trained with a softmax loss over the source data and both the networks also minimize the MMD loss to extract domain invariant features.

3.4.3 Adversarial Methods

In recent years, one of the most significant advances to deep learning has been the introduction of Generative Adversarial Networks (GAN) by Goodfellow *et al.* (2014). GANs are generative networks that generate data (text, images, audio, etc.), such that the data follows a predetermined distribution $P(X)$. A vanilla GAN implementation has two deep networks, *generator* $g(.)$ and *discriminator* $f(.)$, competing against each other. The *generator* network takes in a noise vector $\boldsymbol{z} \in \mathbb{R}^d$ sampled from a uniform or normal distribution and generates an output $g(\boldsymbol{z})$. The *discriminator* takes in $\boldsymbol{x} \in P(X)$ and $g(\boldsymbol{z})$ are tries to discriminate between the two. The generator network tries to fool the discriminator network by generating data that appears to belong to $P(X)$ and the discriminator tries to distinguish between real images and fake images. The equilibrium is a saddle point in the network parameter space.

Apart from GANs, there are other generative models like the Variational Autoencoders (VAE) in Kingma and Welling (2013) and PixelRNN in Oord *et al.* (2016) and they all have their pros and cons [1]. GANs provide the sharpest images but are very difficult to optimize due to unstable training dynamics. VAEs provide complex Bayesian inference models but the generated images can be blurry. PixelRNNs have

[1] https://openai.com/blog/generative-models/

a simple and stable training process, but their sampling strategy is inefficient. GANs have had tremendous success in multiple applications. Some of these include, image super-resolution (Ledig *et al.* (2016)), text-to-image generation (Reed *et al.* (2016); Zhang *et al.* (2016)), image-to-image generation (Isola *et al.* (2016)) and conditional image generation (Nguyen *et al.* (2016); Chen *et al.* (2016)). In this section, some of the successful adversarial based domain adaptation techniques will be discussed.

In the Domain Adversarial Neural Network (DaNN) in Ganin *et al.* (2016), the authors train a deep neural network in a domain adversarial manner for image classification based domain adaptation. The bottom layers of the network act as feature extractors. The features from the bottom layers are fed into two branches of the network. The first branch is a softmax classifier trained with the labeled source data. The second branch is a domain classifier trained to distinguish between the features of the source and the target. The key to the DaNN is the gradient reversal layer connecting the bottom feature extraction layers and the domain classifier. During backpropagation, the gradient from the domain classifier is reversed when learning the feature extractor weights. In this way, the feature extractor is trained to extract domain invariant features. A closely related work is also presented in Ajakan *et al.* (2014).

Liu and Tuzel implement a Coupled Generative Adversarial Network model (CoGAN) in Liu and Tuzel (2016). The CoGAN trains a coupled network which shares weights at different layers of the GAN. The CoGAN is setup so that the lower layers of the generator and the upper layers of the discriminator share weights. A common noise vector $\boldsymbol{z}$, is fed into the two generators $g_1(.)$ and $g_2(.)$ to generate outputs $g_1(\boldsymbol{z})$ and $g_2(\boldsymbol{z})$. These inputs are fed into two discriminators $f_1(.)$ and $f_2(.)$. The discriminator $f_1(.)$ is trained to discriminate between $g_1(\boldsymbol{z})$ and the source $\boldsymbol{x}^s$. Likewise, the discriminator $f_2(.)$ is trained to discriminate between $g_2(\boldsymbol{z})$ and the target $\boldsymbol{x}^t$.

In addition, the source discriminator has an additional softmax layer to classify the source data points $\boldsymbol{x}^s$. The CoGAN was tested with MNIST and USPS data to yield impressive unsupervised domain adaptation results. It remains to be seen if these results can be replicated with object classification datasets.

The Pixel-GAN in Bousmalis *et al.* (2017), is a straightforward extension of the GAN for unsupervised domain adaptation. In this model, instead of a noise vector input $\boldsymbol{z}$, the generator is input the source images and trained to convert them into target images. The discriminator on the other hand, is trained to distinguish between real target images and generated target images (fake target images generated from the source). In addition, a separate network is trained to classify the generated target images. There have been many recent works applying adversarial training for domain adaptation. A few of the most recent procedures are, Kamnitsas *et al.* (2016), Tzeng *et al.* (2017), Sankaranarayanan *et al.* (2017) and Peng and Saenko (2017).

3.4.4 Sundry Deep Methods

One of the earliest procedures for deep learning domain adaptation was proposed by Chopra et al. Chopra *et al.* (2013). The Deep Learning for domain adaptation by Interpolation between Domains (DLID) learns a cross-domain representation by interpolating the path between the source and target domains along the lines of Gopalan *et al.* (2011). Multiple datasets with varying ratios of source and target data points are sampled to create intermediate representations between the two domains. The final cross-domain feature is a concatenation of all the intermediate feature representations.

Hu et al. (Hu *et al.* (2015)), develop a metric learning method for supervised transfer learning using clustering. The Deep Transfer Metric Learning (DTML) model trains a deep neural network to minimize intra-class distances and increase inter-class

distances. In addition, the features of the last layer of the network are learned to be domain invariant by minimizing the MMD between the source and target feature outputs. Further along clustering methods, Sener et al. in Sener *et al.* (2016), train a deep neural network to estimate the labels of target data in a transductive setting. The algorithm learns an asymmetric similarity metric relating the source data with the target data. The deep network predicts the labels so as to minimize intra-class distances and maximize inter-class distances.

Sun et al. Sun *et al.* (2015b) develop a domain transfer method called the localized action frame (LAF) for fine-grained action localization in temporally untrimmed videos. The LAF motivates domain transfer between weakly labeled web images and videos. The domain transfer works in both directions; the video frames are used to select web images that are relevant and drop non-action web images and in turn the web images are used to select action-like frames and drop non-action frames in the video. After the relevant frames and images are selected, a long short-term memory (LSTM) network to is used to train a fine-grained action detector to model the temporal evolution of actions and classify the action in the frames. The work also released a dataset of sports videos with over $130,000$ videos from 240 categories.

Bousmalis et al. (Bousmalis *et al.* (2016)), train Domain Separation Networks (DSN) to extract domain-invariant feature representations and domain-specific representations of source and target data. A shared encoder network $E_c(\boldsymbol{x})$ is trained to extract domain invariant feature representations for the source and the target data. Private encoder networks $E_p^s(\boldsymbol{x})$ and $E_p^t(\boldsymbol{x})$ for the source and target respectively, are trained to extract feature representations that are distinct from the domain-invariant representations that are the outputs of $E_c(\boldsymbol{x})$. A shared decoder network is trained to reconstruct the original input data based on the outputs from $E_c(\boldsymbol{x})$, $E_p^s(\boldsymbol{x})$ and $E_p^t(\boldsymbol{x})$ A classifier is trained with the source outputs of $E_c(\boldsymbol{x})$. The feature representations

that are the outputs of $E_c(\boldsymbol{x})$ can be declared to be domain-invariant.

3.5 Miscellaneous Methods for Domain Adaptation

This section outlines an assortment of methods that could not be classified into the above categories.

3.5.1 Manifold based Methods

Manifold methods for domain adaptation treat the subspaces of the source and target domains as points on a manifold. Since it is a manifold of subspaces, these manifolds are termed as Grassmann manifolds. The task of domain adaptation is to then learn a transformation that transforms one domain to another and can be represented as a curve on the manifold. In Gopalan *et al.* (2011) and Gopalan *et al.* (2014), the authors sample intermediate subspaces along the manifold curve connecting the source domain to the target domain. These transformations are applied to the source data to gradually transfer it to the target subspace. The authors in Gong *et al.* (2012a), sample an infinite number of such subspaces along the source-target-curve. The effect of applying a sequence of infinite such subspace transformations is captured with the Geodesic Flow Kernel.

3.5.2 Dictionary Based Methods

A sparse representation of signals and images has multiple applications. High dimensional data can often be represented using sparse combination of signals from a specified dictionary. Dictionary based domain adaptation methods seek to learn common dictionaries for the source and target and represent data from either domains as sparse vectors encoding these signals. A framework for transforming a dictionary learned on one domain to another, while maintaining a sparse domain-invariant signal

representation, is implemented in Qiu *et al.* (2012). In Shekhar *et al.* (2013), the authors learn a common domain invariant dictionary for both the domains in a semi-supervised setting. In order to ensure high correlation between the features from different domains, the model projects the data to a common low-dimensional subspace while also maintaining the data structure on the manifold. Ni et al. in Ni *et al.* (2013), bring together manifold methods along with dictionary approaches by generating a set of dictionaries connecting the source dictionary with the target dictionary along the lines of Gopalan *et al.* (2011).

3.5.3 Feature Augmentation Methods

One of the less complex approaches to domain adaptation was developed by Daumé III in Daumé III (2007). The procedure concatenates the vectors from the domains to capture domain specific representations and domain invariant representations. Every data point in the source and target domain is represented as follows: $\phi^s(\boldsymbol{x}^s) = [\boldsymbol{x}^{s\top}, \boldsymbol{x}^{s\top}, \mathbf{0}^\top]^\top$, and $\phi^t(\boldsymbol{x}^t) = [\boldsymbol{x}^{t\top}, \mathbf{0}^\top, \boldsymbol{x}^{t\top}]^\top$. It was demonstrated that for a linear classifier trained with these features, the first component captured the domain invariant attributes of the data and the next two components captured the domain specific attributes. Daumé III also introduced a kernelized version of the algorithm in the same work. In Daumé III *et al.* (2010), the authors extend the work to semi-supervised domain adaptation.

The authors in Li *et al.* (2014), extend feature augmentation by including feature transformation. The new augmented and transformed features are given by,

$$\phi^s(\boldsymbol{x}^s) = [W_1\boldsymbol{x}^{s\top}, \boldsymbol{x}^{s\top}, \mathbf{0}_{d_T}^\top]^\top \quad \text{and} \quad \phi^t(\boldsymbol{x}^t) = [W_2\boldsymbol{x}^{t\top}, \mathbf{0}_{d_S}^\top, \boldsymbol{x}^{t\top}]^\top \tag{3.31}$$

The method is applicable for heterogeneous domain adaptation where the source and target data have different dimensions. $W_1 \in \mathbb{R}^{k\times d_S}$ and $W_2 \in \mathbb{R}^{k\times d_T}$ are the

transformation matrices with d_S and d_T being the source and target data dimensions respectively. The transformation matrices W_1 and W_2 map the data to a common subspace enabling domain adaptation.

This concludes the survey on domain adaptation methods for computer vision. The survey gives a new perspective on the gamut of research in this area and will hopefully provide new insights to researchers interested in domain adaptation.

Chapter 4

LINEAR FEATURE SPACES FOR DOMAIN ADAPTATION

There are various models that attempt to achieve domain adaptation between the source and the target. Chapter (3), provides a classification of different approaches to achieving domain adaptation. This chapter describes an example from one such class of methods - namely linear approaches. Linear models for domain adaptation are built using some form of linear transformation of the source and/or the target. A standard approach is to learn a projection matrix that transforms data from one domain, so that cross domain disparity is minimized as in Saenko *et al.* (2010). Another standard linear approach is to match the higher order moments of the two domains, and this reduces domain discrepancy as discussed in Sun *et al.* (2015a). The most common approach for linear domain adaptation is based on classifier adaptation using the Support Vector Machine (SVM). The linear SVM learns a linear decision boundary, which can be expressed as a linear combination of the training data points. In an adaptive SVM, the decision boundary learned with the source training data, is modified to act as a decision boundary for the target data, as in Bruzzone and Marconcini (2010); Aytar and Zisserman (2011).

In this chapter, a linear SVM for domain adaptation is outlined. In this case the domain adaptive model is a couple of SVM decision boundaries - one for the source and the other for the target. Section (4.1) introduces the linear model for domain adaptation which is then developed in Section (4.2). This section derives the two decision boundary SVM and reduces it to a standard SVM form. Section (4.3) outlines the experiments that were conducted to test the proposed model. The final section summarizes the contributions along with proposals for extensions.

4.1 A Linear Model for Domain Adaptation

This work considers the problem of semi-supervised domain adaptation, where labeled samples from the source domain are used along with a limited number of labeled samples from the target domain, to train a linear classier for the target domain. A linear Support Vector Machine called Coupled-SVM model is outlined. The Coupled-SVM simultaneously estimates linear SVM decision boundaries $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$, for the source and target training data respectively.

Using a technique termed as instance matching, researchers sample source data points such that the difference between the means of the sampled source and target data is minimized, Duan *et al.* (2012); Long *et al.* (2014). The intuition behind the Coupled-SVM is along similar lines, where the difference between $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$ is penalized. Since the SVM decision boundaries are a linear combination of the data points, penalizing the difference between $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$, can be viewed as penalizing the difference between the weighted means of the source and target data points. Figure (4.1a), illustrates standard SVM based domain adaptation, where $\boldsymbol{w}_s$ is first learned for the source and it is perturbed to obtain the target ($\boldsymbol{w}_t$). The perturbed SVM $\boldsymbol{w}_t$ could be vastly different from $\boldsymbol{w}_s$ and can over fit the target training data. Figure (4.1b), depicts the Coupled-SVM, where $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$, are learned simultaneously. The source SVM $\boldsymbol{w}_s$, provides an anchor for the target SVM $\boldsymbol{w}_t$. The difference between $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$ is modeled based on the difference between the source and target domains. In addition, the Coupled-SVM trades training error for improved generalization, as illustrated in Figure (4.1c).

The following sections describe the Coupled-SVM model. Although the model outputs two decision boundaries, the Coupled-SVM can be reduced to a standard SVM formulation. The model is compared with other SVM based domain adaptation

models using different datasets.

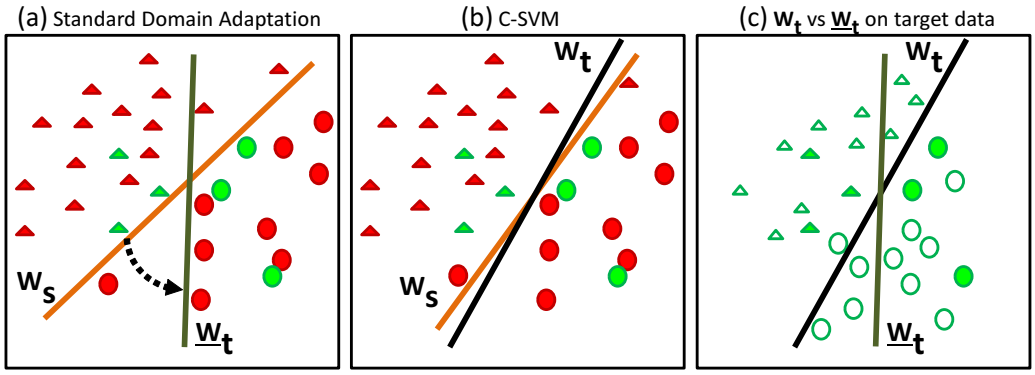


Figure 4.1: (a) Domain adaptation with a SVM. The SVM for the source $\boldsymbol{w}_s$ is modified to get the target SVM $\underline{\boldsymbol{w}}_t$. (b) In the Coupled-SVM, both $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$ are learned together. In this setting, the training error can be high. (c) The Coupled-SVM does not over fit. *Red* is source and *Green* is target data. Filled objects are train data and unfilled objects are test data. Image based on Venkateswara *et al.* (2015b).

4.2 The Coupled Support Vector Machine

A linear classifier is learned with the coupled SVM model for the source and the target simultaneously. The idea is to modify the source classifier ($\boldsymbol{w}_s$) to learn a target classifier ($\boldsymbol{w}_t$). The difference $||\boldsymbol{w}_s - \boldsymbol{w}_t||^2$, between the source and the target classifier is penalized by a weight.

4.2.1 Coupled-SVM Notation

The source data is denoted as $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s} \subset \mathscr{S}$, where $\mathscr{S}$ is the *source domain*, $\boldsymbol{x}_i^s \in \mathbb{R}^d$, are data points and $y_i^s \in \{-1, +1\}$, are their labels. The following discussion considers a binary category classification model which can be extended to a multi-category scenario. Similarly, $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t} \subset \mathscr{T}$, where $\mathscr{T}$ is the *target domain*, $\boldsymbol{x}_i^t \in \mathbb{R}^d$, are data points and $y_i^t \in \{-1, +1\}$, are their labels. The goal is to develop a classifier $f_t(.)$, that can predict the labels $\hat{y}^t = \text{sgn}\big(f_t(\boldsymbol{x}^t)\big), \forall \boldsymbol{x}^t \in \{\boldsymbol{x}^t | (\boldsymbol{x}^t, y^t) \in \mathscr{T}\}$,

where, sgn(.) is the sign function. The constraint here is that the number of target samples, n_t, is limited to a few samples from each category, i.e. $n_t \ll n_s$. Training the classifier $f_t(.)$ with only n_t samples could lead to over-fitting. The classifier $f_t(.)$, should generalize to a larger subset of $\mathscr{T}$ and must not over-fit the training data $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}$. To this end, the source data $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ is also used to train the target classifier.

4.2.2 Coupled-SVM Model

The linear domain adaptation model is outlined as follows:

$$\min_{f_s, f_t} \quad \lambda \mathcal{D}(f_s, f_t) + \mathcal{R}(f_s, f_t) + C_s \sum_i^{n_s} \mathcal{L}(y_i^s, \boldsymbol{x}_i^s, f_s) + C_t \sum_i^{n_t} \mathcal{L}(y_i^t, \boldsymbol{x}_i^t, f_t) \tag{4.1}$$

The aim is to estimate classifiers $\{f_s, f_t\}$ such that Equation (4.1) is minimized. The first term is a measure of the difference between the source classifier f_s and target classifier f_t. λ controls the importance of this difference. The hypothesis is that the source and target classifiers are related. The similarity(dissimilarity) between the datasets is expected to be captured by the similarity(dissimilarity) between the classifiers. The second term penalizes the complexity of the classifiers f_s and f_t. The third and fourth terms are a measure of the classification error over the source data and the target data respectively. C_s and C_t, control the importance of the loss terms for the source and target data respectively. A linear classifier such as a Linear SVM is considered for the discussion, with $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$, where, $\boldsymbol{w} \in \mathbb{R}^d$ is the SVM decision boundary, b is the SVM bias, which is a scalar and $\boldsymbol{w}^\top$ represents *transpose* of $\boldsymbol{w}$. The square of the L_2 norm, $||.||_2^2$ is chosen as the *Regularizer* $\mathcal{R}(.)$. The loss function is set to the standard hinge loss, $\mathcal{L}(y, \boldsymbol{x}, f) = \max\big(0, 1 - y.f(\boldsymbol{x})\big)$. The loss function can be rewritten in the form of constraints for a 'soft'-margin SVM model (see Figure (4.2)).

SVM Model: This box outlines a brief review of the constrained SVM model and its equivalence to the unconstrained SVM model. The linearly separable case for SVM requires solving the constrained optimization problem,

$$\begin{aligned} &\min_{\boldsymbol{w}} \quad ||\boldsymbol{w}||_2^2 \\ &\text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 \quad \text{for } i \in \{1, \ldots, n\} \end{aligned} \tag{4.2}$$

Here, $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ is the labeled training data and $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. The 'soft'-margin constrained SVM model addresses the case of linear inseparability and is given by,

$$\begin{aligned} &\min_{\boldsymbol{w}} \quad ||\boldsymbol{w}||_2^2 + C\sum_{i=1}^{n} \xi_i \\ &\text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \quad \text{for } i \in \{1, \ldots, n\} \end{aligned} \tag{4.3}$$

The slack variables ξ_i ease the strict constraint on separability by allowing data points to lie within the margin and also on the incorrect side of the SVM margin. The constraint $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i$ can be rephrased as, $y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i$, where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$. Along with $\xi_i \geq 0$, the constraint can be written as,

$$\xi_i = \max(0, 1 - y_i f(\boldsymbol{x}_i)) \tag{4.4}$$

Equation (4.3) can then be expressed in an unconstrained form as,

$$\min_{\boldsymbol{w}} \quad ||\boldsymbol{w}||_2^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f(\boldsymbol{x}_i)) \tag{4.5}$$

The second term in Equation (4.5) is the loss and in this case, it is called the *hinge* loss.

Figure 4.2: SVM Model: Constrained and unconstrained formulations.

The source classifier is defined by $f_s(\boldsymbol{x}) = \boldsymbol{w}_s^\top \boldsymbol{x} + b_s$. Similarly, the target classifier is, $f_t(\boldsymbol{x}) = \boldsymbol{w}_t^\top \boldsymbol{x} + b_t$. The source and target SVM decision boundaries are $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$, respectively. The source and target biases are b_s and b_t, respectively. To simplify the notation, bias values are incorporated into the decision boundaries. The decision boundaries are redefined as, $\boldsymbol{w}_s \leftarrow [\boldsymbol{w}_s^\top, b_s]^\top$ and $\boldsymbol{w}_t \leftarrow [\boldsymbol{w}_t^\top, b_t]^\top$. To account for the bias, the data variables are augmented with 1. The re-defined data variables are, $\boldsymbol{x}_i^s \leftarrow [\boldsymbol{x}_i^{s^\top}, 1]^\top$ and $\boldsymbol{x}_i^t \leftarrow [\boldsymbol{x}_i^{t^\top}, 1]^\top$. Including these definitions, Equation (4.1) can be redefined as follows,

$$\begin{aligned} \{\boldsymbol{w}_s^*, \boldsymbol{w}_t^*\} = & \underset{\boldsymbol{w}_s, \boldsymbol{w}_t}{\operatorname{argmin}} \ \frac{1}{2}\lambda ||\boldsymbol{w}_s - \boldsymbol{w}_t||_2^2 + \frac{1}{2}||\boldsymbol{w}_s||_2^2 + \frac{1}{2}||\boldsymbol{w}_t||_2^2 + \ C_s \sum_i^{n_s} \xi_i^s + C_t \sum_i^{n_t} \xi_i^t \\ & \text{s.t.} \ \ y_i^s(\boldsymbol{w}_s^\top \boldsymbol{x}_i^s) \geq 1 - \xi_i^s, \ \xi_i^s \geq 0, \quad i \in [1, \ldots, n_s] \\ & \qquad y_i^t(\boldsymbol{w}_t^\top \boldsymbol{x}_i^t) \geq 1 - \xi_i^t, \ \xi_i^t \geq 0, \quad i \in [1, \ldots, n_t] \end{aligned} \tag{4.6}$$

The terms in Equation (4.6) are rearranged for formatting reasons. The dissimilarity measure $\mathcal{D}(.)$, is defined as the square of the Euclidean distance between $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$. The rest of the terms make up a set of two SVM models, one for the source data and the other for the target data. Equation (4.6) is a modification of the standard Linear SVM having two decision boundaries along with an additional term capturing the relation the two decision boundaries. In the following subsection, Equation (4.6) is re-formulated as a standard SVM.

4.2.3 Coupled-SVM Solution

In order to reduce Equation (4.6) to a standard SVM problem, a new set of variables are defined based on the existing variables. The two SVM decision boundaries are concatenated into a single variable, $\boldsymbol{w} \in \mathbb{R}^{2(d+1)}$ defined as,

$$\boldsymbol{w} \leftarrow [\boldsymbol{w}_s^\top, \boldsymbol{w}_t^\top]^\top \tag{4.7}$$

The individual SVMs $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$ can be extracted from $\boldsymbol{w}$ using permutation matrices $I_s \in \mathbb{R}^{(d+1)\times 2(d+1)}$ and $I_t \in \mathbb{R}^{(d+1)\times 2(d+1)}$, where I_s and I_t are binary matrices such that,

$$I_s\boldsymbol{w} = \boldsymbol{w}_s \quad \text{and} \quad I_t\boldsymbol{w} = \boldsymbol{w}_t \tag{4.8}$$

For example, let $\boldsymbol{v}_1 = [a, b]^\top$, $\boldsymbol{v}_2 = [c, d]^\top$ and $\boldsymbol{v} \leftarrow [\boldsymbol{v}_1^\top, \boldsymbol{v}_2^\top]^\top$. Then the permutation matrices I_{v_1} and I_{v_2} such that, $I_{v_1}\boldsymbol{v} = \boldsymbol{v}_1$ and $I_{v_2}\boldsymbol{v} = \boldsymbol{v}_2$, are given by,

$$I_{v_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad I_{v_2} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

New variables $(\boldsymbol{x}_i, y_i, c_i)$ $\forall i \in \{1, \ldots, (n_s + n_t)\}$, are introduced, where, $\boldsymbol{x}_i \in \mathbb{R}^{2(d+1)}$ and $y_i \in \{-1, +1\}$ are the new data points and $c_i \in \{C_s, C_t\}$ is the importance of the classification error for the i^{th} data point. $\boldsymbol{x}_i$ is defined as,

$$\boldsymbol{x}_i \leftarrow \begin{cases} [\boldsymbol{x}_i^{s^\top}, \mathbf{0}^\top]^\top & 1 \le i \le n_s \\ [\mathbf{0}^\top, \boldsymbol{x}_{i-n_s}^{t^\top}]^\top & (n_s+1) \le i \le (n_s + n_t) \end{cases} \tag{4.9}$$

where, $\mathbf{0} \in \mathbb{R}^{d+1}$ is a vector of zeros. Similarly, $\{y_i, c_i\}$ are defined as,

$$\{y_i, c_i\} \leftarrow \begin{cases} \{y_i^s, C_s\} & 1 \le i \le n_s \\ \{y_{i-n_s}^t, C_t\} & (n_s+1) \le i \le (n_s + n_t) \end{cases} \tag{4.10}$$

With the definitions of $\boldsymbol{w}$ and $\boldsymbol{x}_i$, it can be noted that

$$\boldsymbol{w}^\top \boldsymbol{x}_i = \begin{cases} \boldsymbol{w}_s^\top \boldsymbol{x}_i^s & 1 \le i \le n_s \\ \boldsymbol{w}_t^\top \boldsymbol{x}_{i-n_s}^t & (n_s+1) \le i \le (n_s + n_t) \end{cases} \tag{4.11}$$

For the remaining derivation, it is assumed the data is linearly separable, i.e. slack variables $\xi_i^s = 0$ and $\xi_i^t = 0$, $\forall i$. The slack variables will be re-introduced towards the

end. The minimization problem in Equation (4.6) can now be re-formulated as,

$$\begin{aligned} \{\boldsymbol{w}^*\} \;&=\; \underset{\boldsymbol{w}}{\operatorname{argmin}} \;\frac{1}{2}\lambda||I_{st}\boldsymbol{w}||_2^2 \;+\; \frac{1}{2}||\boldsymbol{w}||_2^2 \\ &\text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i) \geq 1 \quad \forall i \in \{1, \ldots, (n_s + n_t)\} \end{aligned} \tag{4.12}$$

where, $I_{st} \leftarrow (I_s - I_t)$ with $I_s\boldsymbol{w} = \boldsymbol{w}_s$, $I_t\boldsymbol{w} = \boldsymbol{w}_t$ for the first term. The second term is obtained with, $\frac{1}{2}||\boldsymbol{w}_s||_2^2 + \frac{1}{2}||\boldsymbol{w}_t||_2^2 = \frac{1}{2}||\boldsymbol{w}||_2^2$. Since this is a linearly separable case, the constraints adhere to a margin 1. In order to solve the optimization problem, Lagrangian variables $\{\alpha_i\}$ are introduced for each of the constraints, $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i) \geq 1$. The Lagrangian is defined as,

$$L(\boldsymbol{w}, \boldsymbol{\alpha}) \;=\; \frac{1}{2}\lambda||I_{st}\boldsymbol{w}||_2^2 \;+\; \frac{1}{2}||\boldsymbol{w}||_2^2 \;-\; \sum_i^{n_s+n_t} \alpha_i\big(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - 1)\big) \tag{4.13}$$

The Lagrangian needs to be minimized w.r.t. $\boldsymbol{w}$ and maximized w.r.t. to $\boldsymbol{\alpha}$. Optimization is carried out first w.r.t. $\boldsymbol{w}$ by setting the derivative $\frac{\partial L(\boldsymbol{w},\boldsymbol{\alpha})}{\partial \boldsymbol{w}} = \mathbf{0}$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \lambda I_{st}^\top I_{st}\boldsymbol{w} + \boldsymbol{w} - \sum_i^{n_s+n_t} \alpha_i y_i \boldsymbol{x}_i = \mathbf{0} \tag{4.14}$$

$$\implies \boldsymbol{w} = D \sum_i^{n_s+n_t} \alpha_i y_i \boldsymbol{x}_i \tag{4.15}$$

where, I is an identity matrix and,

$$D = (I + \lambda I_{st}^\top I_{st})^{-1} \tag{4.16}$$

Because of the nature of the permutation matrices I_s and I_t, $(I + \lambda I_{st}^\top I_{st})$ is full rank and therefore, D exists. Following the definition of $\boldsymbol{v} \leftarrow \sum_i^{n_s+n_t} \alpha_i y_i \boldsymbol{x}_i$ and a substitution for $\boldsymbol{w}$ in Equation (4.13), the optimization problem in terms of Lagrangian

variable $\boldsymbol{\alpha}$ is,

$$
\begin{aligned}
\{\boldsymbol{\alpha}^*\} &= \underset{\boldsymbol{\alpha}}{\text{argmax}}\ \lambda\frac{1}{2}\boldsymbol{v}^\top D^\top I_{st}^\top I_{st} D\boldsymbol{v} + \frac{1}{2}\boldsymbol{v}^\top D^\top D\boldsymbol{v} - \boldsymbol{v}^\top D\boldsymbol{v} + \mathbf{1}^\top\boldsymbol{\alpha} \\
&= \underset{\boldsymbol{\alpha}}{\text{argmax}}\ \boldsymbol{v}^\top(\frac{1}{2}D^\top D + \frac{1}{2}\lambda D^\top I_{st}^\top I_{st} D - D)\boldsymbol{v} + \mathbf{1}^\top\boldsymbol{\alpha} \\
&= \underset{\boldsymbol{\alpha}}{\text{argmax}}\ \boldsymbol{v}^\top(\frac{1}{2}D^\top(I + \lambda I_{st}^\top I_{st})D - D)\boldsymbol{v} + \mathbf{1}^\top\boldsymbol{\alpha} \\
&= \underset{\boldsymbol{\alpha}}{\text{argmax}}\, \boldsymbol{v}^\top(\frac{1}{2}D^\top - D)\boldsymbol{v} + \mathbf{1}^\top\boldsymbol{\alpha} \quad \text{(Equation (4.16))} \\
&= \underset{\boldsymbol{\alpha}}{\text{argmin}}\ \frac{1}{2}\boldsymbol{v}^\top D\boldsymbol{v} - \mathbf{1}^\top\boldsymbol{\alpha} \quad (D \text{ is symmetric}) \\
&= \underset{\boldsymbol{\alpha}}{\text{argmin}}\ \frac{1}{2}\boldsymbol{v}^\top D\boldsymbol{v} - \mathbf{1}^\top\boldsymbol{\alpha} \\
&= \underset{\boldsymbol{\alpha}}{\text{argmin}}\ \frac{1}{2}\boldsymbol{\alpha}^\top\mathcal{Q}\boldsymbol{\alpha} - \mathbf{1}^\top\boldsymbol{\alpha}. \qquad (4.17)
\end{aligned}
$$

Equation (4.17) can be viewed as the standard SVM dual where $\mathcal{Q}_{ij} = y_i y_j \boldsymbol{x}_i^\top D\boldsymbol{x}_j$ and $\mathbf{1}$ is a vector of $1s$. By defining $\tilde{\boldsymbol{x}}_i = D^{1/2}\boldsymbol{x}_i$, the matrix $\mathcal{Q}$ can be expressed in a standard SVM dot product form, $\mathcal{Q}_{i,j} = y_i y_j \tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{x}}_j$ which reduces Equation (4.17) to a standard quadratic minimization problem. Any of the standard SVM libraries can then be used to arrive at a solution for $\boldsymbol{\alpha}$.

In the space of $\tilde{\boldsymbol{x}}_i$, the decision bondary is given by, $\tilde{\boldsymbol{w}} = \sum_i \alpha_i y_i \tilde{\boldsymbol{x}}_i$. In the space of $\boldsymbol{x}_i$, the decision boundary is given by $\boldsymbol{w} = D\sum_i \alpha_i y_i \boldsymbol{x}_i$. Therefore, $\boldsymbol{w} = D^{1/2}\tilde{\boldsymbol{w}}$. For the experiments in this chapter, the LIBLINEAR Fan *et al.* (2008), package was adapted. It has an implementation of a weighted SVM where weights (importance) can be specified for each training data point. The slack variables ξ_i^s and ξ_i^t will now be re-introduced through the weight term c_i. The c_i is introduced as a constraint on α_i. In solving Equation (4.17), the constraint on α_i is usually given by, $0 \leq \alpha_i \leq \infty$. This constraint is used for a linearly separable SVM solution. The linearly inseparable SVM ('soft'-margin), can be modeled by modifying the constraint to, $0 \leq \alpha_i \leq c_i$. This is equivalent to solving a 'soft'-margin SVM by introducing slack variables, ξ_i^s

and ξ_i^t.

Once $\boldsymbol{w}$ is known, $\boldsymbol{w}_s$ and $\boldsymbol{w}_t$, can be estimated using Equation (4.8). The Coupled-SVM can be easily extended to the multi-class setting using one-vs-one or one-vs-all settings. SVM packages like LIBLINEAR provide solutions to multi-class configurations. Using LIBLINEAR, the decision boundaries of the multi-class SVM with P categories will be a matrix $W \in \mathbb{R}^{(P \times 2(d+1))}$, where each row of W is a decision boundary. The first $(d+1)$ columns of each row correspond to $\boldsymbol{w}_s$ and the second $(d+1)$ columns correspond to $\boldsymbol{w}_t$ for a particular category. The Coupled-SVM algorithm for the binary case is outlined in Algorithm (1).

Algorithm 1 Coupled-SVM Domain Adaptation

Input: $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}, \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}, \lambda, C_s, C_t$

Output: $\boldsymbol{w}_s^*, \boldsymbol{w}_t^*$ as in Equation (4.6)

$D \leftarrow (I + \lambda I_{st}^\top I_{st})^{-1}$ (Equation (4.16))

for $i \leftarrow 1$ **to** $(n_s + n_t)$ **do**

    Define $\boldsymbol{x}_i$ as in Equation (4.9)

    $\tilde{\boldsymbol{x}}_i \leftarrow D^{1/2}\boldsymbol{x}_i$

    Define $\{y_i, c_i\}$ as in Equation (4.10)

$\tilde{\boldsymbol{w}} \leftarrow$ Linear Weighted SVM $\{\tilde{\boldsymbol{x}}_i, y_i, c_i\}_{i=1}^{(n_s+n_t)}$

$\boldsymbol{w} \leftarrow D^{1/2}\tilde{\boldsymbol{w}}$

$\boldsymbol{w}_s^* \leftarrow I_s\boldsymbol{w}$ and $\boldsymbol{w}_t^* \leftarrow I_t\boldsymbol{w}$ (Equation (4.8))

4.3 Experimental Analysis for the Coupled-SVM

This section discusses the extensive experiments that were conducted to study the Coupled-SVM model. The different datasets and their domains are first outlined followed by a brief introduction to the baselines that are used for camparison. This

is followed by the experimental layout and a discussion of the results.

4.3.1 Experimental Setup

The Coupled-SVM was evaluated with multiple datasets from different applications and with multiple types of features. For all the experiments (except *Office-Caltech*) the following setting has been used. For the training data, 20 examples are sampled from the source domain and 3 examples from the target domain from every category. The remaining examples in the target domain that are not used for training are considered as test data. A few sampled images from the datasets are depicted in Figure (4.3).

Figure 4.3: Top Row: Images of objects from Datasets, `Amazon`, `Dslr`, `Webcam` and `Caltech`. Second Row: Facial expression images from `CKPlus` and `MMI`. Thrid row: Digit images from `MNIST` and `USPS` followed by snapshots from `HMDB51` and `UCF50`.

***MNIST-USPS* datasets**: The digit datasets *MNIST* and *USPS* consist of images of individual digits from 0 to 9. They are benchmark datasets for handwritten digit recognition. In the following experiments, a subset of these datasets (2000 images

from *MINST* and 1800 images from *USPS*) based on Long *et al.* (2014) has been used. The images are represented as vectors of length 256 after resizing the images to 16×16 pixels. These domains are referred to as `MNIST` and `USPS` respectively.

***Office-Caltech* datasets**: The experimental setup for this dataset has been adapted from Gong *et al.* (2012a). The *Office* dataset is made up of three domains, `Amazon`, `Dslr` and `Webcam`. An additional domain called `Caltech` is included based on the *Caltech256* dataset. All of these domains are made up of images from a set of common categories viz., {*back-pack, bike, calculator, headphones, computer-keyboard, laptop, monitor, computer-mouse, coffee-mug, video-projector*}. For the experiments, the 800 dimension SURF-Bag-of-Words (SURF-BoW) features provided by Gong *et al.* (2012a) are used. In creating the training data, 8 examples are sampled from the source domain (for `Amazon` 20 are used) and 3 examples are sampled from the target domain.

***CKPlus-MMI* dataset**: The *CKPlus* Lucey *et al.* (2010) and *MMI* Pantic *et al.* (2005) are popular datasets for facial expression recognition. From these datasets, six categories were selected viz., {*anger*, *disgust*, *fear*, *happy*, *sad*, and *surprise*}, from video frames with the most intense expression (peak frames) for every facial expression video sequence. This yields around 1500 images for each dataset with around 250 images per category. These domains are referred to as `CKPlus` and `MMI`. A deep neural network was used to extract feature vectors from the images. Features extracted using pre-trained convolutional neural networks (CNN) have shown astonishingly good results across a wide array of applications Razavian *et al.* (2014). Therefore, the deep CNN developed by Simonyan and Zisserman Simonyan and Zisserman (2014) was deployed as an 'off-the-shelf' feature extractor. The outputs of the first fully connected layer from the 16 weight layer model with dimension 4096 were used as features. These were reduced to 100 dimensions using PCA.

***HMDB51-UCF50* dataset**: Eleven common categories of activity were gathered from *HMDB51* Kuehne *et al.* (2011) and *UCF50* Reddy and Shah (2013). The categories from *UCF50* are as follows with the corresponding categories from *HMDB51* in parenthesis, {*BaseballPitch(throw)*, *Basketball(shoot_ball)*, *Biking(ride_bike)*, *Diving(dive)*, *Fencing(fencing)*, *GolfSwing(golf)*, *HorseRiding(ride_horse)*, *PullUps (pullup)*, *PushUps(pushup)*, *Punch(punch)*, *WalkingWithDog(walk)*}. These domains are referred to as `HMDB51` and `UCF50`. State-of-the-art **HOG**, **HOF**, **MBHx** and **MBHy** descriptors were extracted from the videos based on the work of Kantorov and Laptev (2014). The descriptors were then pooled into a grid **1x1x1**, and Fisher Vectors were estimated with $K = 256$ Gaussians. The resulting large dimension Fishers Vectors ($202,752$ dimensions) were reduced to 100 dimensions using PCA.

4.3.2 Baselines for Comparison

The Coupled-SVM is compared with the following semi-supervised domain adaptation techniques based on SVMs. **SVM(T)**: This is a linear SVM trained on the target labeled data. It can be expected to over-fit the training data and perform poorly on the test data. **SVM(S)**: This is a linear SVM with training data from source domain. It can be expected to perform poorly when there is domain discrepancy between the source and target domains. **SVM(S+T)**: This is a linear SVM trained with a union of source and target domain training data. In this case too, domain discrepancy can lead to poor performance. The above three methods are baselines that the Coupled-SVM is expected to outperform. **MMDT**: The Max-Margin Domain Transform Hoffman *et al.* (2013), learns a single SVM model for the source and the transformed target data. In the MMDT, the target data is transformed by a transformation matrix to be similar to the source data and the transformation is learned in an optimization framework along with the SVM classifier. **AMKL**: The

Adaptive Multiple Kernel Learning Duan *et al.* (2012), implements a multiple kernel method where multiple base kernel classifiers are combined with a pre-learned average classifier obtained from fusing multiple nonlinear SVMs. Unlike in Coupled-SVM where the similarity between source and target is learned by the model, Widmer et al. Widmer *et al.* (2012) use a similar approach to solve multitask problems using graph Laplacians to model task similarity. The Coupled-SVM holds a unique position in this wide array of SVM solutions for domain adaptation. The Coupled-SVM trains a linear SVM for both the source and target domains simultaneously, thereby minimizing the chances of over-fitting, especially when there are very few labeled samples from the target domain. The Coupled-SVM is labeled **C-SVM** in the figures and tables.

4.3.3 Results

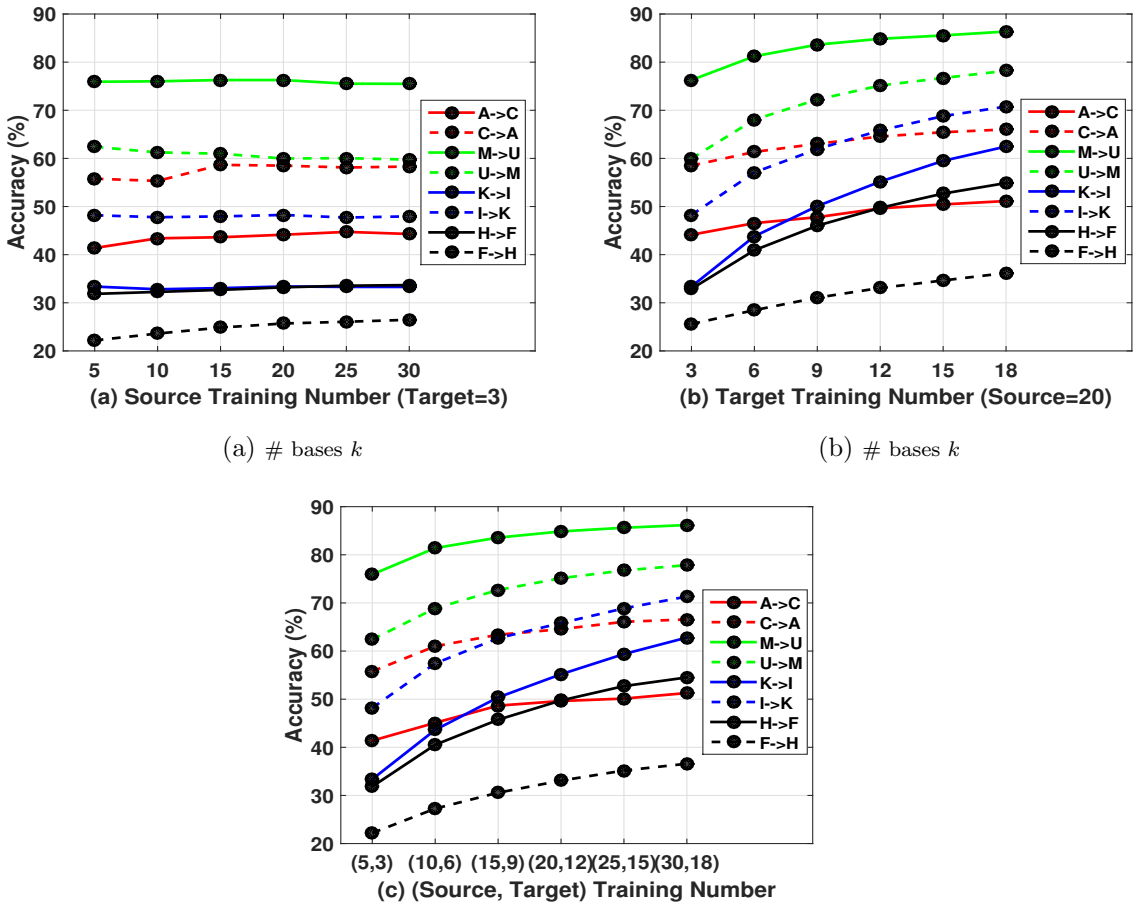
Eighteen experiments were conducted using different pairs of datasets as source and target. Table (4.1) depicts the target recognition accuracies for multiple algorithms. The results were averaged across 20 splits of data for the *Office-Caltech* dataset. For the remaining datasets, 100 splits were used. The domain difference is highlighted with the results from the SVM(S) experiments. Although the datasets comprise of the same categories, a classifier trained on the source does not perform as well on the target because of the domain disparity. This fact is further highlighted by the success of SVM(T), which shows improved accuracies even when a few of the target training data points are labeled. The SVM(S+T) illustrates that the naïve union of the source and target training data may be beneficial in some cases but not always. The Coupled-SVM outperforms the remaining algorithms but is nearly on par with the AMKL. Although there is little to choose between the two in terms of recognition accuracies, the Coupled-SVM is faster and it is easier to implement it

Table 4.1: Target recognition accuracies (%) for the object, digit, facial expression and activity datasets across multiple algorithms. {`Amazon`(A), `Webcam`(W), `Dslr`(D), `Caltech`(C), `MNIST`(M), `USPS`(U), `CKPlus`(K), `MMI`(I), `HMDB51`(H), `UCF50`(F)}. A→W implies A is source domain and W is target domain. The best results are highlighted in bold.

Expt.	SVM(T)	SVM(S)	SVM(S+T)	MMDT	AMKL	C-SVM
A→ W(1)	56.06±0.95	37.36±1.19	51.26±1.19	64.87±1.26	**67.85±1.06**	66.40±1.09
A→ D(2)	43.15±0.78	37.64±0.96	47.56±0.99	54.41±1.00	56.22±0.89	**57.13±0.98**
W→ A(3)	44.39±1.18	32.03±0.90	44.87±0.59	50.54±0.82	52.96±0.57	**53.97±0.42**
W→ D(4)	45.20±1.34	61.06±0.86	65.39±0.89	62.48±0.98	**75.95±0.94**	68.27±0.86
D→ A(5)	42.17±1.03	31.48±0.65	46.17±0.44	50.45±0.75	52.36±0.57	**54.10±0.55**
D→ W(6)	54.91±0.80	69.81±1.06	76.19±0.64	74.34±0.66	**85.94±0.44**	77.17±0.46
A→ C(7)	26.62±0.60	38.61±0.50	42.46±0.39	39.67±0.50	**44.92±0.46**	**44.74±0.57**
W→ C(8)	25.82±0.78	26.67±0.59	34.53±0.76	34.86±0.79	**39.20±0.57**	**39.77±0.59**
D→ C(9)	26.88±0.74	25.74±0.47	34.68±0.67	35.82±0.75	**41.12±0.44**	**41.27±0.51**
C→ A(10)	43.52±1.07	36.22±0.82	47.75±0.60	51.10±0.76	**55.98±0.58**	**55.56±0.76**
C→ W(11)	55.49±1.02	29.72±1.54	51.28±1.23	62.94±1.11	**68.70±1.07**	67.74±1.05
C→ D(12)	43.07±1.47	32.56±1.03	47.68±1.17	52.56±0.97	58.82±0.83	**59.72±1.01**
M→ U(13)	70.73±0.41	38.89±0.61	64.36±0.41	68.96±0.43	**79.56±0.30**	76.02±0.34
U→ M(14)	58.23±0.39	21.67±0.33	38.43±0.36	48.29±0.32	**63.80±0.32**	**63.25±0.31**
K→ I(15)	**33.31±0.27**	13.30±0.15	25.83±0.31	18.28±0.42	31.87±0.29	**33.10±0.29**
I→ K(16)	45.65±0.47	19.47±0.55	25.63±0.35	21.33±0.81	43.59±0.50	**48.54±0.47**
H→ F(17)	28.94±0.26	17.45±0.17	23.00±0.19	29.05±0.23	33.06±0.23	**35.89±0.25**
F→ H(18)	18.64±0.19	16.99±0.16	19.58±0.17	22.28±0.18	**24.28±0.16**	**24.41±0.19**

compared to the AMKL which is a multiple kernel based method.

Leave-one-out cross validation was applied for the training target data in order to determine the optimal values of the model parameters $\{C_s, C_t, \lambda\}$. To get a better idea of the role of labeled data, the number of labeled data points was varied and the recognition accuracies were studied. Since the `Webcam` and `Dslr` datasets have fewer data points compared to the other domains, they were left out of this anal-



(a) # bases k

(b) # bases k

(c) # bases k

Figure 4.4: Average target recognition accuracies (%) for the Coupled-SVM across different experiments by varying number of labeled training examples in source and target. Images based on Venkateswara *et al.* (2015b).

ysis. The results of this study are along expected lines. Increasing the number of source training data points, has little effect on improving the target test accuracies as illustrated in Figure (4.4a). The reason being, the decision boundaries do not vary significantly with additional data. To estimate the source decision boundary, the SVM relies on support vectors. Only the support vectors affect the direction of the decision boundary. Since most of the data is not made up of support vectors and lies away from the boundary, additional source training data does not change the decision boundaries. On the other hand, the effect of additional target training data

is comparatively more pronounced as seen in Figure (4.4b). Labeled target data is limited, and adding any new labeled target data points will help reshape the decision boundary. Both of the above conclusions are reinforced with the results in Figure (4.4c). The effect of increasing the number of both the source and target labeled data points is comparable to increasing the number of labeled target data alone. The additional source labeled data does not alter the decision boundary as much as the number of additional labeled target data. In other words, additional source data does not contribute to the target SVM after the decision boundary has stabilized.

4.4 Conclusions and Summary

The experimental analysis indicates that the Coupled-SVM is a promising alternative for domain adaptation. The AMKL is nearly as good as the Coupled-SVM, but the Coupled-SVM has the advantage of being simple. Being a linear model, the Coupled-SVM is fast and easy to implement using existing libraries. In its current form, the Coupled-SVM requires some labeled data in the target domain, which is not unreasonable. However, a possible extension to the Coupled-SVM could be an unsupervised version that needs no target data labels. The Coupled-SVM models the difference between the domains as the difference between their decision boundaries. In order to estimate the importance of this difference (λ), a target validation dataset is required which is not available for unsupervised domain adaptation. Chapter (5) will outline a procedure to create a validation set based on source data.

In this chapter a linear model for semi-supervised domain adaptation was introduced. The Coupled-SVM model trains two SVM classifiers - one for the source and another for the target. The linear model is efficient, elegant and easy to implement as it can be implemented using existing SVM libraries. Experimental results indicate that the Coupled-SVM performs comparably well against competitive linear methods.

Chapter 5

NONLINEAR FEATURE SPACES FOR DOMAIN ADAPTATION

A linear model for domain adaptation was introduced in Chapter (4). Sometimes, a linear model may be overly simplistic and may not effectively ameliorate the discrepancy between the distributions of the two domains. When linear approaches are ineffective in modeling domain diversity, *nonlinear* transformations of data are the next available option for achieving domain adaptation. This chapter introduces a nonlinear model for unsupervised domain adaptation.

Chapter (3) provided a survey of nonlinear approaches for domain adaptation. A nonlinear model for domain adaptation is usually a kernel method which projects the data to a high-dimensional (possibly infinite dimensional) space and aligns the domains in that space. A standard procedure for nonlinear domain alignment is using the Maximum Mean Discrepancy (MMD), as described in Equation (3.12). Many nonlinear methods for domain adaptation use some form of MMD for domain alignment. The standard nonlinear approaches include max-margin methods (nonlinear SVMs) like Duan *et al.* (2009), instance selection based on MMD like Chattopadhyay *et al.* (2013) and spectral methods like Kernel-PCA as in Long *et al.* (2014). This chapter introduces a spectral based nonlinear method for unsupervised domain adaptation. In addition to domain adaptation, the model enhances classification using manifold based embedding of the nonlinearly transformed data.

The chapter is organized as follows. Section (5.1) provides an introduction to the proposed nonlinear model with an intuitive toy example. Section (5.2) outlines the model by deriving the various components. This section also introduces a validation technique for unsupervised domain adaptation, in the absence of labeled target

data. Section (5.3) provides the results of various experiments with the proposed model. 50 different domain adaptation experiments were conducted to compare the proposed model with existing competitive procedures for unsupervised domain adaptation. The validation procedure was evaluated using 7 popular domain adaptation image datasets, including object, face, facial expression and digit recognition datasets.

5.1 A Nonlinear Model for Domain Adaptation

Nonlinear techniques are deployed in situations where the source and target domains cannot be aligned using linear transformations. These techniques apply nonlinear transformations on the source and target data in order to align them. For example, Maximum Mean Discrepancy (MMD) is applied to learn nonlinear representations, where the difference between the source and target distributions is minimized, as in Pan *et al.* (2011). Even though nonlinear transformations may align the domains, the resulting data may not be conducive to classification. If, after domain alignment, the data were to be clustered based on similarity, it can lead to effective classification.

A two domain binary classification toy problem demonstrates this intuition. Figure (5.1a), depicts the two domains of a two-moon dataset. Figure (5.1b), shows the data after it has been transformed by Kernel-PCA. Kernel-PCA projects the data along nonlinear directions ('curves') of maximum variance, however the data gets dispersed and there is no domain alignment. In Figure (5.1c), the data is presented after the domains have been aligned using Maximum Mean Discrepancy (MMD). The domains are now more aligned, however the classification enhancement is not guaranteed. Figure (5.1d), displays the data after it has been embedded using similarity-based embedding along with MMD based domain alignment. This makes the data more classification friendly. A classifier trained on the source will now provide enhanced accuracies on the target data.

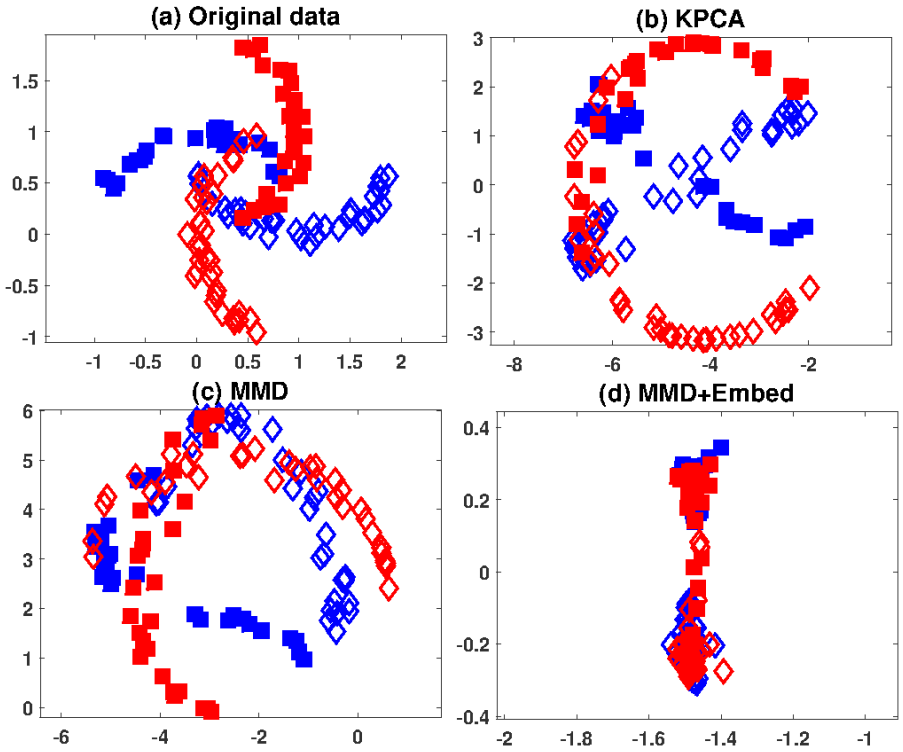


Figure 5.1: Two-class classification problem with with source data in blue and target data in red. The target labels are unknown. (a) Plot of the original 2-dimensional data, (b) The data after Kernel-PCA based that projects the data along nonlinear directions of maximum variance, (c) The data after MMD based projection that aligns the two domains, (d) The data after MMD+Similarity-based Embedding that aligns the domains and also clusters the data to ensure easy classification. Images based on Venkateswara *et al.* (2017a).

This chapter introduces the Nonlinear Embedding Transform (NET), which performs a nonlinear transformation to align the source and target domains and also cluster the data based on label-similarity. The NET algorithm is a spectral (eigen) technique that requires certain parameters (like number of eigen bases, etc.) to be pre-determined. These parameters are often given random values which need not be optimal as in Pan *et al.* (2011); Long *et al.* (2013) and Long *et al.* (2014). This chapter also outlines a validation procedure to fine-tune model parameters with a validation set created from the source data. The two major contributions of this work are outlined as follows:

- Nonlinear embedding transform (NET) algorithm for unsupervised DA.
- Validation procedure to estimate optimal parameters for an unsupervised DA algorithm.

The following sections outline the NET model and compare its performance with other state-of-the-art domain adaptation techniques across multiple datasets.

5.2 Nonlinear Embedding Transformation Model

The NET algorithm for unsupervised domain adaptation is outlined in the following section. It is followed by the description of a cross-validation procedure that can be used to estimate the model parameters for the NET algorithm.

As is usually the case for domain adaptation, two domains are considered; source domain $\mathscr{S}$ and target domain $\mathscr{T}$. The source domain is represented by $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s} \subset \mathscr{S}$ and the target domain is represented by $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t} \subset \mathscr{T}$. In matrix notation, the source data is given by $\mathbf{X}_S = [\boldsymbol{x}_1^s, \ldots, \boldsymbol{x}_{n_s}^s] \in \mathbb{R}^{d \times n_s}$ and the target data by $\mathbf{X}_T = [\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_{n_t}^t] \in \mathbb{R}^{d \times n_t}$. Similarly, the corresponding labels are represented by $Y_S = [y_1^s, \ldots, y_{n_s}^s]$ and $Y_T = [y_1^t, \ldots, y_{n_t}^t]$ for the source and target data respectively. The source and target data have the same dimensionality where, $\boldsymbol{x}_i^s$ and $\boldsymbol{x}_i^t \in \mathbb{R}^d$ and the label space for the two domains is identical, i.e., y_i^s and $y_i^t \in \{1, \ldots, C\}$. Additional terms are introduced for later use like, $\mathbf{X} \coloneqq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] = [\mathbf{X}_S, \mathbf{X}_T]$, with $n = n_s + n_t$. In the unsupervised domain adaptation setting, the target labels Y_T are unknown and the joint distributions of the two domains are different with, $P_S(X, Y) \neq P_T(X, Y)$.

In order to predict the target data labels, a classifier $f(\boldsymbol{x}) = p(y|\boldsymbol{x})$ is trained. The posterior $p(y|\boldsymbol{x})$ is the probability that data point $\boldsymbol{x}$ is assigned label y. The labels $\hat{Y}_T = [\hat{y}_1^t, \ldots, \hat{y}_{n_t}^t]$ are the estimated target data labels corresponding to $\mathbf{X}_T$ using a

classifier trained on $\mathcal{D}_s$ and $\mathbf{X}_T$. A classifier trained with only the source data $\mathcal{D}_s$, may not accurately predict the target data labels because the joint distributions of the source and target data are different. The NET algorithm modifies the source and target data by projecting them to a common subspace using nonlinear transformations. In this subspace, the cross domain disparity is reduced and the data points are clustered together based on the similarity of their labels. A classifier is then trained using the projected source data which is then used to predict the target data labels.

5.2.1 Nonlinear Domain Alignment

A standard procedure to project data to a common subspace is through Principal Component Analysis (PCA). A common basis for the source and target data is estimated and this basis is used to linearly transform the source and target data. This is often a baseline procedure applied to reduce domain disparity and can be viewed as a naïve form of domain adaptation. PCA maximizes the variance of the projected data by estimating a projection matrix $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k] \in \mathbb{R}^{d \times k}$. The variance of the projected data is given by $\sum_i^n ||\mathbf{U}^\top \boldsymbol{x}_i||_2^2$, where $\mathbf{U}$ is determined by solving,

$$\max_{\mathbf{U}^\top \mathbf{U} = \mathbf{I}} \text{tr}(\mathbf{U}^\top \mathbf{X} \mathbf{H} \mathbf{X}^\top \mathbf{U}). \tag{5.1}$$

$\mathbf{H}$ is the $n \times n$ centering matrix given by $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}$, $\mathbf{I}$ is an identity matrix and $\mathbf{1}$ is a $n \times n$ matrix of 1s. The constraint ensures the principal components are orthonormal. The Lagrangian for Equation (5.1) is given by $L(\mathbf{U}, \mathbf{\Lambda}) = \text{tr}(\mathbf{U}^\top \mathbf{X} \mathbf{H} \mathbf{X}^\top \mathbf{U}) + \mathbf{U}^\top \mathbf{U} \mathbf{\Lambda}$, where $\mathbf{\Lambda} = \mathit{diag}(\lambda_1, \ldots, \lambda_k)$ are the Lagrangian constants. Differentiating with respect to $\mathbf{U}$ and setting $\frac{\partial L}{\partial \mathbf{U}} = 0$ leads to the standard eigen-decomposition problem $\mathbf{X} \mathbf{H} \mathbf{X}^\top \mathbf{U} = \mathbf{U} \mathbf{\Lambda}$. The solution to Equation (5.1) is given by the standard eigen-value problem $\mathbf{X} \mathbf{H} \mathbf{X}^\top \mathbf{U} = \mathbf{U} \mathbf{\Lambda}$, where $\mathbf{\Lambda} = \mathit{diag}(\lambda_1, \ldots, \lambda_k)$ and $(\lambda_1, \ldots, \lambda_k)$ are the k-largest eigen-values. The PCA projected data in the k-dimensional subspace is given

by $\mathbf{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n] = \mathbf{U}^\top \mathbf{X} \in \mathbb{R}^{k \times n}$.

The PCA provides a linear projection of the source and target data to a common subspace. When linear projection does not provide a good basis for projection (i.e. cross-domain disparity is not reduced), often, Kernel-PCA (KPCA) is applied to estimate a nonlinear projection of the data. In this case, data is mapped to a high-dimensional (possibly infinite-dimensional) space defined by $\Phi(\mathbf{X}) = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]$. $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ defines a mapping function and $\mathcal{H}$ is a RKHS (Reproducing Kernel Hilbert Space). The dot product between the high-dimensional mapped vectors $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{y})$, is estimated by the kernel-trick. The dot product is given by the positive semi-definite (psd) kernel, $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$. The kernel $k(.)$ can be viewed as a similarity measure between $\boldsymbol{x}$ and $\boldsymbol{y}$. The similarity measure between all pairs of data points in $\mathbf{X}$, is represented using the kernel gram matrix and is given by, $\mathbf{K} = \Phi(\mathbf{X})^\top \Phi(\mathbf{X}) \in \mathbb{R}^{n \times n}$. The high-dimensional mapped data $\Phi(\mathbf{X})$ is projected onto a subspace of eigen-vectors (directions of maximum nonlinear variance in the RKHS). The leading k eigen-vectors in the RKHS are denoted using the representer theorem $\mathbf{U} = \Phi(\mathbf{X})\mathbf{A}$ Kimeldorf and Wahba (1970). Matrices $\mathbf{U}$ and $\Phi(\mathbf{X})$ are never estimated in practice since they are high dimensional or even infinite dimensional. The coefficient matrix $\mathbf{A}$ is instead determined and the projected data is obtained using $\mathbf{A}$. The kernelized version of Equation (5.1) with $\mathbf{A}$ as the projection matrix that is obtained by solving,

$$\max_{\mathbf{A}^\top \mathbf{A} = \mathbf{I}} \text{tr}(\mathbf{A}^\top \mathbf{K} \mathbf{H} \mathbf{K}^\top \mathbf{A}). \tag{5.2}$$

Here, $\mathbf{H}$ is the $n \times n$ centering matrix given by $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}$, $\mathbf{I}$ is an identity matrix and $\mathbf{1}$ is a $n \times n$ matrix of 1s. The matrix of coefficients is $\mathbf{A} \in \mathbb{R}^{n \times k}$ and the nonlinear projected data is given by $\mathbf{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n] = \mathbf{A}^\top \mathbf{K} \in \mathbb{R}^{k \times n}$.

Implementing a nonlinear projection of data onto a common subspace may not

necessarily account for domain disparity. In such a situation, domain alignment is achieved using Maximum Mean Discrepancy (MMD) Gretton *et al.* (2009), which is a standard nonparametric measure to estimate domain disparity. To ensure the joint distributions of the source and target are aligned, the Joint Distribution Adaptation (JDA) Long *et al.* (2013), algorithm which seeks to align both the the marginal and conditional probability distributions of the projected data, is adopted. The marginal distributions are aligned by estimating the coefficient matrix $\mathbf{A}$, which minimizes:

$$\min_{\mathbf{A}} \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{A}^\top \boldsymbol{k}_i - \frac{1}{n_t} \sum_{j=n_s+1}^{n} \mathbf{A}^\top \boldsymbol{k}_j \right\|_{\mathcal{H}}^2 = \operatorname{tr}(\mathbf{A}^\top \mathbf{K} \mathbf{M}_0 \mathbf{K}^\top \mathbf{A}). \tag{5.3}$$

$\mathbf{M}_0$, is the MMD matrix which given by,

$$(\mathbf{M}_0)_{ij} = \begin{cases} \frac{1}{n_s n_s}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_s \\ \frac{1}{n_t n_t}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_t \\ \frac{-1}{n_s n_t}, & \text{otherwise,} \end{cases} \tag{5.4}$$

Likewise, the conditional distribution difference can also be minimized by introducing matrices M_c, with $c = 1, \ldots, C$, defined as,

$$(\mathbf{M}_c)_{ij} = \begin{cases} \frac{1}{n_s^{(c)} n_s^{(c)}}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_s^{(c)} \\ \frac{1}{n_t^{(c)} n_t^{(c)}}, & \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_t^{(c)} \\ \frac{-1}{n_s^{(c)} n_t^{(c)}}, & \begin{cases} \boldsymbol{x}_i \in \mathcal{D}_s^{(c)}, \boldsymbol{x}_j \in \mathcal{D}_t^{(c)} \\ \boldsymbol{x}_j \in \mathcal{D}_s^{(c)}, \boldsymbol{x}_i \in \mathcal{D}_t^{(c)} \end{cases} \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

Here, $\mathcal{D}_s$ and $\mathcal{D}_t$ are the subsets of source and target data points respectively. $\mathcal{D}_s^{(c)}$ is the subset of source data points whose class label is c and $n_s^{(c)} = |\mathcal{D}_s^{(c)}|$. Similarly, $\mathcal{D}_t^{(c)}$ is the subset of target data points whose class label is c and $n_t^{(c)} = |\mathcal{D}_t^{(c)}|$. For the target data, since the labels are not known, the predicted target labels are used

to determine $\mathcal{D}_t^{(c)}$. The target data labels are initialized using a classifier trained on the source data and refined over iterations. Incorporating both the conditional and marginal distribution alignments, the JDA model can be written as,

$$\min_{\mathbf{A}} \sum_{c=0}^{C} \text{tr}(\mathbf{A}^\top \mathbf{K} \mathbf{M}_c \mathbf{K}^\top \mathbf{A}). \tag{5.6}$$

5.2.2 Similarity Based Embedding

Along with domain alignment, the NET algorithm seeks to project the data in a classification friendly manner (easily classifiable). Laplacian eigenmaps are used in order to cluster data points on the basis of class label similarity. For this purpose an $(n \times n)$ adjacency matrix $\mathbf{W}$ is initialized. The entries in the adjacency matrix capture the similarity between pairs of data points. These entries are used to weight the distances between pairs of data points. $\mathbf{W}$ is defined as,

$$\mathbf{W}_{ij} := \begin{cases} 1 & y_i^s = y_j^s \text{ or } i = j \\ 0 & y_i^s \neq y_j^s \text{ or labels unknown.} \end{cases} \tag{5.7}$$

The clustering is implemented by minimizing the sum of squared distances weighted by the adjacency matrix. This is expressed as a minimization problem,

$$\min_{\mathbf{Z}} \frac{1}{2} \sum_{ij} \left|\left| \frac{\boldsymbol{z}_i}{\sqrt{d_i}} - \frac{\boldsymbol{z}_j}{\sqrt{d_j}} \right|\right|^2 \mathbf{W}_{ij} = \min_{\mathbf{A}} \text{tr}(\mathbf{A}^\top \mathbf{K} \mathbf{L} \mathbf{K}^\top \mathbf{A}). \tag{5.8}$$

Here, $d_i = \sum_k \mathbf{W}_{ik}$ and $d_j = \sum_k \mathbf{W}_{jk}$. They constitute the entries of $\mathbf{D}$ the $(n \times n)$ diagonal matrix. $||\boldsymbol{z}_i/\sqrt{d_i} - \boldsymbol{z}_j/\sqrt{d_j}||^2$, is the squared normalized distance between the projected data points $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, which get clustered together when $\mathbf{W}_{ij} = 1$, (as they belong to the same category). Since the target labels are unknown, distances involving target data points are not weighted. The proposition to use predicted target labels refined over multiple iterations did not provide satisfactory results. The normalized distance provides a more robust clustering measure when compared to the standard

Euclidean distance $||\boldsymbol{z}_i - \boldsymbol{z}_j||^2$, Chung (1997). By substituting $\mathbf{Z} = \mathbf{A}^\top \mathbf{K}$, Equation (5.8) can be expressed in terms of $\mathbf{A}$, where $\mathbf{L}$, denotes the symmetric positive semi-definite graph laplacian matrix with $\mathbf{L} \coloneqq \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, and $\mathbf{I}$ being an identity matrix.

5.2.3 Optimization Problem

The different components, namely, nonlinear projection in Equation (5.2), joint distribution alignment in Equation (5.6) and similarity based embedding in Equation (5.8) are now included to design the domain adaptation model. Maximizing Equation (5.2) and minimizing Equations (5.6) and (5.8) is equivalent to maintaining Equation (5.2) constant and minimizing Equations (5.6) and (5.8), according to the generalized Raleigh quotient. However, minimizing the similarity embedding in Equation (5.8) can result in a trivial solution with the projected vectors being embedded in a low dimensional subspace. A new constraint is introduced in place of $\mathbf{A}^\top \mathbf{K}\mathbf{H}\mathbf{K}^\top \mathbf{A} = \mathbf{I}$, in order to enforce subspace dimensionality. The NET is introduced as an optimization problem obtained by minimizing Equations (5.6) and (5.8). The goal is still to determine the $(n \times k)$ projection matrix, $\mathbf{A}$. By including Frobenius norm based regularization for a smooth solution, along with the dimensionality constraint, the NET optimization problem is given by,

$$\min_{\mathbf{A}^\top \mathbf{K}\mathbf{D}\mathbf{K}^\top \mathbf{A} = \mathbf{I}} \alpha.\mathrm{tr}(\mathbf{A}^\top \mathbf{K} \sum_{c=0}^{C} \mathbf{M}_c \mathbf{K}^\top \mathbf{A}) + \beta.\mathrm{tr}(\mathbf{A}^\top \mathbf{K}\mathbf{L}\mathbf{K}^\top \mathbf{A}) + \gamma||\mathbf{A}||_F^2. \tag{5.9}$$

The first term controls the domain alignment and its importance is given by α. The second term weighted by β captures similarity based embedding. The third term is the standard regularization (Frobenius norm) that ensures a smooth projection matrix $\mathbf{A}$ and its importance is denoted by γ. As mentioned earlier, the constraint on $\mathbf{A}$ (in place of $\mathbf{A}^\top \mathbf{K}\mathbf{H}\mathbf{K}^\top \mathbf{A} = \mathbf{I}$), prevents a trivial solution where the projection

could collapse onto a subspace with dimensionality less than k, Belkin and Niyogi (2003). In order to solve Equation (5.9) the Lagrangian is introduced,

$$L(\mathbf{A}, \mathbf{\Lambda}) = \alpha.\mathrm{tr}\big(\mathbf{A}^\top \mathbf{K} \sum_{c=0}^{C} \mathbf{M}_c \mathbf{K}^\top \mathbf{A}\big) + \beta.\mathrm{tr}(\mathbf{A}^\top \mathbf{K}\mathbf{L}\mathbf{K}^\top \mathbf{A}) + \gamma||\mathbf{A}||_F^2 + \mathrm{tr}((\mathbf{I} - \mathbf{A}^\top \mathbf{K}\mathbf{D}\mathbf{K}^\top \mathbf{A})\mathbf{\Lambda}), \tag{5.10}$$

where the Lagrangian constants are captured in the diagonal matrix $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_k)$. With the derivative set to zero, $\frac{\partial L}{\partial \mathbf{A}} = 0$, a generalized eigen-value problem is obtained,

$$\Big(\alpha \mathbf{K} \sum_{c=0}^{C} \mathbf{M}_c \mathbf{K}^\top + \beta \mathbf{K}\mathbf{L}\mathbf{K}^\top + \gamma \mathbf{I}\Big)\mathbf{A} = \mathbf{K}\mathbf{D}\mathbf{K}^\top \mathbf{A}\mathbf{\Lambda}. \tag{5.11}$$

The coefficient matrix $\mathbf{A}$ in Equation (5.9) is given by the k-smallest eigen-vectors of Equation (5.11). The domain-aligned and embedded data points are then given by $\mathbf{Z} = \mathbf{A}^\top \mathbf{K}$. The NET algorithm is outlined in Algorithm 2.

5.2.4 Model Selection

In a supervised learning setting, validation data (subset of the training data with labels) is used to estimate the optimal value of the model parameters. In unsupervised domain adaptation the target labels are treated as unknown. When estimating the optimum value for the model parameters, current domain adaptation methods appear to inherently assume the availability of target labels Long *et al.* (2013), Long *et al.* (2014) for validation. However, in the case of real world applications, when target labels may not available, it becomes challenging to ensure optimal model parameters. In the case of the NET model, there are 4 parameters $(k, \alpha, \beta, \gamma)$, that need to be pre-determined. Since the source data is available and it contains labels, a subset of the source data can be used for validation purposes. A technique using Kernel Mean Matching (KMM) is introduced to sample the source data to create a validation set. The KMM is based on the MMD and it is used to weight the source data points in

Algorithm 2 Nonlinear Embedding Transform

Input: $\mathbf{X}$, Y_S, constants α, β, regularization γ and projection dimension k. Number of iterations to converge, T.

Output: Projection matrix $\mathbf{A}$, projected data $\mathbf{Z}$.

Compute kernel matrix $\mathbf{K}$, for predefined kernel $k(.,.)$
Define the adjacency matrix $\mathbf{W}$ (Equation (5.7))
Compute $\mathbf{D} = \text{diag}(d_1, \ldots, d_n)$, where $d_i = \sum_j \mathbf{W}_{ij}$
Compute normalized graph laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
Train a classifier with source data $\{[\boldsymbol{x}_1^s, \ldots, \boldsymbol{x}_{n_s}^s], Y_S\}$
Estimate initial target labels $\hat{y}_i^t$ for $t \in \{1, \ldots, n_t\}$ with source classifier
Construct $\mathbf{M}_c$ for $c \in \{0, 1, \ldots, C\}$ based on Equations (5.4) and (5.5)
for $i = 1$ to T **do**
    Solve Equation (5.11) and select k smallest eigen-vectors as columns of $\mathbf{A}$
    Estimate $\mathbf{Z} \leftarrow \mathbf{A}^\top \mathbf{K}$
    Train a source classifier with modified data $\{[\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_s}], Y_S\}$
    Re-estimate labels $\hat{y}_i^t$ for $t \in \{1, \ldots, n_t\}$ with source classifier
    Re-construct $\mathbf{M}_c$ for $c \in \{0, 1, \ldots, C\}$ based on Equations (5.4) and (5.5)
Solve Equation (5.11) and select k smallest eigen-vectors as columns of $\mathbf{A}$
Estimate $\mathbf{Z} \leftarrow \mathbf{A}^\top \mathbf{K}$
Train a classifier with modified data $\{[\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_s}], Y_S\}$

order to reduce the domain disparity between the source and target data Fernando *et al.* (2013), Gong *et al.* (2013a). The KMM learns a set of weights for each of the source data points. The source data points with large weights can be considered to have a marginal distribution similar to that of the target data. These data points are then chosen to constitute subset that can be used for cross validation purposes. The

weights w_i, $i = 1, \ldots, n_s$, are estimated by minimizing,

$$\left\| \frac{1}{n_s} \sum_{i=1}^{n_s} w_i \phi(\boldsymbol{x}_i^s) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\boldsymbol{x}_j^t) \right\|_{\mathcal{H}}^2. \tag{5.12}$$

In order to simplify the equation, $\kappa_i := \frac{n_s}{n_t} \sum_{j=1}^{n_t} k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^t)$, $i = 1, \ldots, n_s$ and $\mathbf{K}_{S_{ij}} = k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^s)$ are defined. The minimization can now be represented as a quadratic programming problem,

$$\begin{aligned} \min_{\boldsymbol{w}} = & \frac{1}{2} \boldsymbol{w}^\top \mathbf{K}_S \boldsymbol{w} - \kappa^\top \boldsymbol{w}, \\ & \text{s.t. } w_i \in [0, B], \ \left| \sum_{i=1}^{n_s} w_i - n_s \right| \leq n_s \epsilon. \end{aligned} \tag{5.13}$$

In the first constraint the scope of discrepancy between source and target distributions is limited, with $B \to 1$, leading to an unweighted solution. The second constraint ensures that $w(x)P_S(x)$, remains a probability distribution Gretton *et al.* (2009). In conducting the experiments, 10% of the source data with the largest weights was chosen to form the validation subset. The optimal values of $(\alpha, \beta, \gamma, k)$, were estimated using the validation set. For fixed values of $(\alpha, \beta, \gamma, k)$, the NET model is trained using the source data (without the validation set) and the target data. The model is tested on the validation subset since it has labels. A grid search for the parameters is conducted to estimate the optimal values yielding the highest validation accuracies.

5.3 Experimental Analysis of the NET Model

In the Experiments section, the NET algorithm is compared with other nonlinear domain adaptation procedures. The NET algorithm is evaluated across various datasets like the *Office*, *Office-Caltech*, *COIL*, *MMI* and *CKPlus*, *MNIST* and *USPS*.

5.3.1 Experimental Setup

The NET model was evaluated extensively with 7 different datasets. The details of the datasets are outlined in Table (5.1).

Table 5.1: Statistics for the benchmark datasets

Dataset	Type	#Samples	#Features	#Classes	Subsets
MNIST	Digit	2,000	256	10	MNIST
USPS	Digit	1,800	256	10	USPS
CKPlus	Face Exp.	1,496	4096	6	CKPlus
MMI	Face Exp.	1,565	4096	6	MMI
COIL20	Object	1,440	1,024	20	COIL1, COIL2
PIE	Face	11,554	1,024	68	P05, ..., P29
Ofc-Cal SURF	Object	2,533	800	10	A, C, W, D
Ofc-Cal Deep	Object	2,505	4096	10	A, C, W, D

***MNIST-USPS* datasets**: This dataset has already been discussed in the previous chapter. It is presented here again for the sake of completeness. The digit datasets *MNIST* and *USPS* consist of images of individual digits from 0 to 9. They are benchmark datasets for handwritten digit recognition. In the following experiments, a subset of these datasets (2000 images from *MINST* and 1800 images from *USPS*) based on Long *et al.* (2014) has been used. The images are represented as vectors of length 256 after resizing the images to 16×16 pixels. These domains are referred to as `MNIST` and `USPS` respectively.

***CKPlus-MMI* dataset**: This dataset has already been discussed in the previous chapter. It is presented here again for the sake of completeness. The *CKPlus* Lucey *et al.* (2010) and *MMI* Pantic *et al.* (2005) are popular datasets for facial expression recognition. From these datasets, six categories were selected viz., {*anger*, *disgust*, *fear*, *happy*, *sad*, and *surprise*}, from video frames with the most intense expression (peak frames) for every facial expression video sequence. This yields around 1500 images for each dataset with around 250 images per category. These domains are referred to as `CKPlus` and `MMI`. A deep neural network was used to extract feature vectors from the images. Features extracted using pre-trained convolutional neural

networks (CNN) have shown astonishingly good results across a wide array of applications Razavian *et al.* (2014). Therefore, the deep CNN developed by Simonyan and Zisserman Simonyan and Zisserman (2014) was deployed as an 'off-the-shelf' feature extractor. The outputs of the first fully connected layer from the 16 weight layer model with dimension 4096 were used as features. These were reduced to 500 dimensions using PCA.

***COIL20* dataset**: It is an object recognition dataset which consists of 20 categories with data belonging to two domains, `COIL1` and `COIL2`. The domains consist of images of objects captured from views that are 5 degrees apart. The images are 32×32 pixels with gray scale values Long *et al.* (2013) that are vectorized to 1024 dimensions.

***PIE* dataset**: The "Pose, Illumination and Expression" (PIE) dataset consists of face images (32×32 pixels) of 68 individuals. The images were captured with different head-pose, illumination and expression. Along the lines of Long *et al.* (2013), 5 subsets were selected with differing head-pose to create 5 domains, namely, `P05` (C05, left pose), `P07` (C07, upward pose), `P09` (C09, downward pose), `P27` (C27, frontal pose) and `P29` (C29, right pose).

***Office-Caltech* dataset**: This dataset has already been discussed in the previous chapter. It is presented here again for the sake of completeness. This is currently the most popular benchmark dataset for object recognition in the domain adaptation computer vision community. The dataset consists of images of everyday objects. It consists of 4 domains; `Amazon`, `Dslr` and `Webcam` from the *Office* dataset and `Caltech` domain from the *Caltech-256* dataset. The `Amazon` domain has images downloaded from the www.amazon.com website. The `Dslr` and `Webcam` domains have images captured using a DSLR camera and a webcam respectively. The `Caltech` domain is a subset of the Caltech-256 dataset that was created by selecting categories common with the *Office* dataset. The *Office-Caltech* dataset has 10 categories of objects and

a total of 2533 images (data points). Two feature types were used for evaluation with the *Office-Caltech* dataset; (i) 800-dimensional SURF features Gong *et al.* (2012a), (ii) Deep features. The deep features are extracted using a pre-trained network similar to the *CKPlus-MMI* datasets.

5.3.2 Baselines for comparison

The NET algorithm is compared with the following baseline and state-of-the-art methods. Similar to NET, the TCA, TJM and JDA are all spectral (eigen) methods.

Table 5.2: Baseline methods that are compared with the NET.

Method	**Reference**
SA	Subspace Alignment Fernando *et al.* (2013)
CA	Correlation Alignment Sun *et al.* (2015a)
GFK	Geodesic Flow Kernel Gong *et al.* (2012a)
TCA	Transfer Component Analysis Pan *et al.* (2011)
TJM	Transfer Joint Matching Long *et al.* (2014)
JDA	Joint Distribution Adaptation Long *et al.* (2013)

All the above four algorithms apply MMD to align the source and target datasets, however the NET, in addition uses nonlinear embedding to enhance classification. In a setting similar to Equation (5.11), TCA, TJM and JDA, solve for $\mathbf{A}$. But unlike NET, they do not incorporate the similarity based embedding term. Also, $\alpha = 1$, is fixed for all the three algorithms. Therefore, these models only have 2 free parameters (γ and k), that need to be pre-determined in contrast to NET, which has 4 parameters, $(\alpha, \beta, \gamma, k)$. Since TCA, TJM and JDA, are all quite similar to each other and they have the same model parameters, for the sake of brevity model selection (estimating optimal model parameters) is evaluated using cross validation for only JDA and NET. The other algorithms, SA, CA and GFK, do not have any critical free model parameters that need to be pre-determined.

In the experiments, NET_v is treated as a special case of the NET, with model parameters $(\alpha, \beta, \gamma, k)$, being determined using a validation set derived from Equation (5.13). Likewise, JDA_v is a special case of JDA, where (γ, k), are determined using a validation set derived from Equation (5.13). There is no theoretical guarantee to the effectiveness of this validation procedure. However, its effectiveness can be measured empirically by comparing it with a procedure that uses target data as a validation set. The results obtained using target data as a validation subset are represented by NET in the figures and tables. For the rest of the algorithms (SA, CA, GFK, TCA, TJM and JDA), the parameter settings described in their respective works were used.

5.3.3 Experimental Details

The same experimental protocol as in Gong *et al.* (2012a); Long *et al.* (2014) is followed to ensure a fair comparison with existing methods. 50 different domain adaptation experiments were conducted with the previously mentioned datasets. In each of these unsupervised domain adaptation experiments, there is one source domain (data points and labels) and one target domain (data points only). Since $\mathbf{M}_c$ is refined over multiple iterations, 10 iterations were run to converge to the predicted test/validation labels. For the kernel function $k(.)$, a Gaussian kernel was used with a standard width equal to the median of the squared distances over the dataset as described in Gretton *et al.* (2009). For all the experiments, the projected target data was tested using a 1-Nearest Neighbor (NN) classifier that was trained using the projected source data. Since it does not require tuning of cross-validation parameters, a NN classifier was chosen as in Gong *et al.* (2012a); Long *et al.* (2014). The percentage of correctly classified target data points are indicated as target recognition accuracies.

Table 5.3: Target recognition accuracies (%) for domain adaptation experiments on the digit and face datasets. {`MNIST`(M), `USPS`(U), `CKPlus`(CK), `MMI`(MM), `COIL1`(C1) and `COIL2`(C2). M→U implies M is source domain and U is target domain. The best and second best results in every experiment (row) are highlighted in **bold** and *italic* respectively. The shaded columns indicate accuracies obtained using model selection.

Expt.	SA	CA	GFK	TCA	TJM	JDA	JDA_v	NET	NET_v
M→U	67.39	59.33	66.06	60.17	64.94	67.28	71.94	**75.39**	*72.72*
U→M	51.85	50.80	47.40	39.85	52.80	59.65	59.65	**62.60**	*61.35*
C1→C2	85.97	84.72	85.00	90.14	91.67	92.64	**95.28**	*93.89*	90.42
C2→C1	84.17	82.78	84.72	88.33	89.86	*93.75*	**93.89**	92.64	88.61
CK→MM	*31.12*	**31.89**	28.75	32.72	30.35	29.78	25.82	29.97	30.54
MM→CK	39.75	37.74	37.94	31.33	*40.62*	28.39	26.79	**45.83**	40.08
P05→P07	26.64	40.33	26.21	40.76	10.80	58.81	*77.53*	**77.84**	69.00
P05→P09	27.39	41.97	27.27	41.79	7.29	54.23	*66.42*	**70.96**	57.41
P05→P27	30.28	55.36	31.15	59.60	15.14	84.50	*90.78*	**91.86**	84.68
P05→P29	19.24	29.04	17.59	29.29	4.72	49.75	**52.70**	*52.08*	45.40
P07→P05	25.42	41.51	25.27	41.78	16.63	57.62	**74.70**	*74.55*	57.92
P07→P09	47.24	53.43	47.37	51.47	21.69	62.93	**79.66**	*77.08*	54.60
P07→P27	53.47	63.77	54.22	64.73	26.04	75.82	81.14	*83.84*	**86.09**
P07→P29	26.84	35.72	27.02	33.70	10.36	39.89	*63.73*	**69.24**	47.30
P09→P05	23.26	35.47	21.88	34.69	14.98	50.96	**77.16**	*73.98*	68.67
P09→P07	41.87	47.08	43.09	47.70	27.26	57.95	*78.39*	**79.01**	67.34
P09→P27	44.97	53.71	46.38	56.23	27.55	68.45	*84.92*	83.48	**87.47**
P09→P29	28.13	34.68	26.84	33.09	8.15	39.95	65.93	**70.04**	*67.65*
P27→P05	35.62	51.17	34.27	55.61	25.96	80.58	*92.83*	**93.07**	92.44
P27→P07	63.66	66.05	62.92	67.83	28.73	82.63	*90.18*	89.99	**93.68**
P27→P09	72.24	73.96	73.35	75.86	38.36	87.25	*90.14*	89.71	**90.20**
P27→P29	36.03	40.50	37.38	40.26	7.97	54.66	72.18	*76.84*	**79.53**
P29→P05	23.05	26.89	20.35	27.01	9.54	46.46	*60.20*	**67.32**	52.67
P29→P07	26.03	31.74	24.62	29.90	8.41	42.05	**71.39**	*70.23*	57.52
P29→P09	27.76	31.92	28.49	29.90	6.68	53.31	*74.02*	**74.63**	62.81
P29→P27	30.31	34.70	31.27	33.67	10.06	57.01	*76.66*	75.43	**80.98**
Average	41.14	47.55	40.65	47.59	26.79	60.63	*72.85*	**74.67**	68.73

Table 5.4: Target recognition accuracies (%) for domain adaptation experiments on the *Office-Caltech* dataset with SURF and Deep features. {`Amazon`(A), `Webcam`(W), `Dslr`(D), `Caltech`(C)}. A→W implies A is source and W is target. The best and second best results in every experiment (row) are highlighted in **bold** and *italic* respectively. The shaded columns indicate accuracies obtained using model selection.

Expt:	SURF Features								
	SA	**CA**	**GFK**	**TCA**	**TJM**	**JDA**	**JDA$_v$**	**NET**	**NET$_v$**
C→A	43.11	36.33	45.72	44.47	**46.76**	44.78	45.41	*46.45*	46.24
D→A	29.65	28.39	26.10	31.63	32.78	33.09	29.85	**39.67**	*35.60*
W→A	32.36	31.42	27.77	29.44	29.96	32.78	29.33	**41.65**	*39.46*
A→C	38.56	33.84	39.27	39.89	39.45	39.36	39.27	**43.54**	*43.10*
D→C	31.88	29.56	30.45	30.99	31.43	31.52	31.08	**35.71**	*34.11*
W→C	29.92	28.76	28.41	32.15	30.19	31.17	31.43	**35.89**	*32.77*
A→D	37.58	36.94	34.40	33.76	**45.22**	39.49	31.85	*40.76*	36.31
C→D	43.95	38.22	43.31	36.94	44.59	*45.22*	40.13	**45.86**	36.31
W→D	*90.45*	85.35	82.17	85.35	89.17	89.17	88.53	89.81	**91.72**
A→W	37.29	31.19	41.70	33.90	*42.03*	37.97	38.98	**44.41**	35.25
C→W	36.27	29.49	35.59	32.88	38.98	*41.69*	37.97	**44.41**	33.56
D→W	87.80	83.39	79.66	85.42	85.42	*89.49*	86.78	87.80	**90.51**
Average	44.90	41.07	42.88	43.07	*46.33*	46.31	44.22	**49.66**	46.24

Expt:	Deep Features								
	SA	**CA**	**GFK**	**TCA**	**TJM**	**JDA**	**JDA$_v$**	**NET**	**NET$_v$**
C→A	88.82	*91.12*	90.60	89.13	91.01	90.07	89.34	**92.48**	90.70
D→A	84.33	86.63	88.40	88.19	88.72	91.22	90.18	**91.54**	*91.43*
W→A	84.01	82.76	88.61	86.21	88.09	91.43	87.04	**92.58**	*91.95*
A→C	80.55	*82.47*	81.01	75.53	78.08	83.01	78.27	**83.01**	82.28
D→C	76.26	75.98	78.63	74.43	76.07	80.09	78.17	*82.10*	**83.38**
W→C	78.90	74.98	76.80	76.71	79.18	**82.74**	78.90	*82.56*	82.28
A→D	82.17	*87.90*	82.80	82.17	87.26	89.81	77.07	**91.08**	80.89
C→D	80.89	82.80	77.07	75.80	82.80	89.17	80.25	**92.36**	*90.45*
W→D	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	*99.36*	**100.00**
A→W	82.37	80.34	84.41	76.61	87.12	87.12	79.32	**90.85**	*87.46*
C→W	77.29	79.32	78.64	78.31	*88.48*	85.76	77.97	**90.85**	84.07
D→W	98.98	*99.32*	98.31	97.97	98.31	98.98	98.98	**99.66**	**99.66**
Average	84.55	85.30	85.44	83.42	87.09	*89.12*	84.63	**90.70**	88.71

5.3.4 Parameter Estimation Study

The procedure for model selection is evaluated below. There are 4 parameters $(k, \alpha, \beta, \gamma)$, for the NET algorithm and 2 parameters (k, γ), for the JDA that need to be pre-determined. To determine these parameters a validation subset is created from the source data by weighting them using Equation (5.13) and selecting 10% of the source data points with the largest weights. This validation subset has a distribution similar to the target and it can be used to validate the optimal values for the model parameters $(\alpha, \beta, \gamma, k)$ since the source data points have labels. A grid search is conducted in the parameter space of $k \in \{10, 20, \ldots, 100, 200\}$ and α, β, γ from the set $\{0, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$. Although a unique set of parameters can be evaluated for every domain adaptation experiment, for the sake of brevity, one set of parameters is presented for every dataset. Once the optimum parameters are evaluated using cross-validation, the NET model is applied on the entire source data (data and labels) and the target data (data only) to estimate the projected source and target data points. The target recognition accuracies obtained are represented as shaded columns JDA_v and NET_v in Tables (5.3) and (5.4).

In order to evaluate the proposed model selection method, the parameters were also determined using the target data as a validation set. The NET column in Tables (5.3) and (5.4) depicts these results. The target recognition accuracies for the NET columns can be considered as the best accuracies for the NET model. The JDA_v and NET_v should be compared with the NET column to evaluate the proposed cross-validation procedure. For the rest of the column values, SA, CA, GFK, TCA, TJM and JDA, their model parameters were fixed based on their respective papers. The target recognition accuracies for NET_v is higher than those of the other domain adaptation methods and is nearly comparable to the NET. This is an empirical vali-

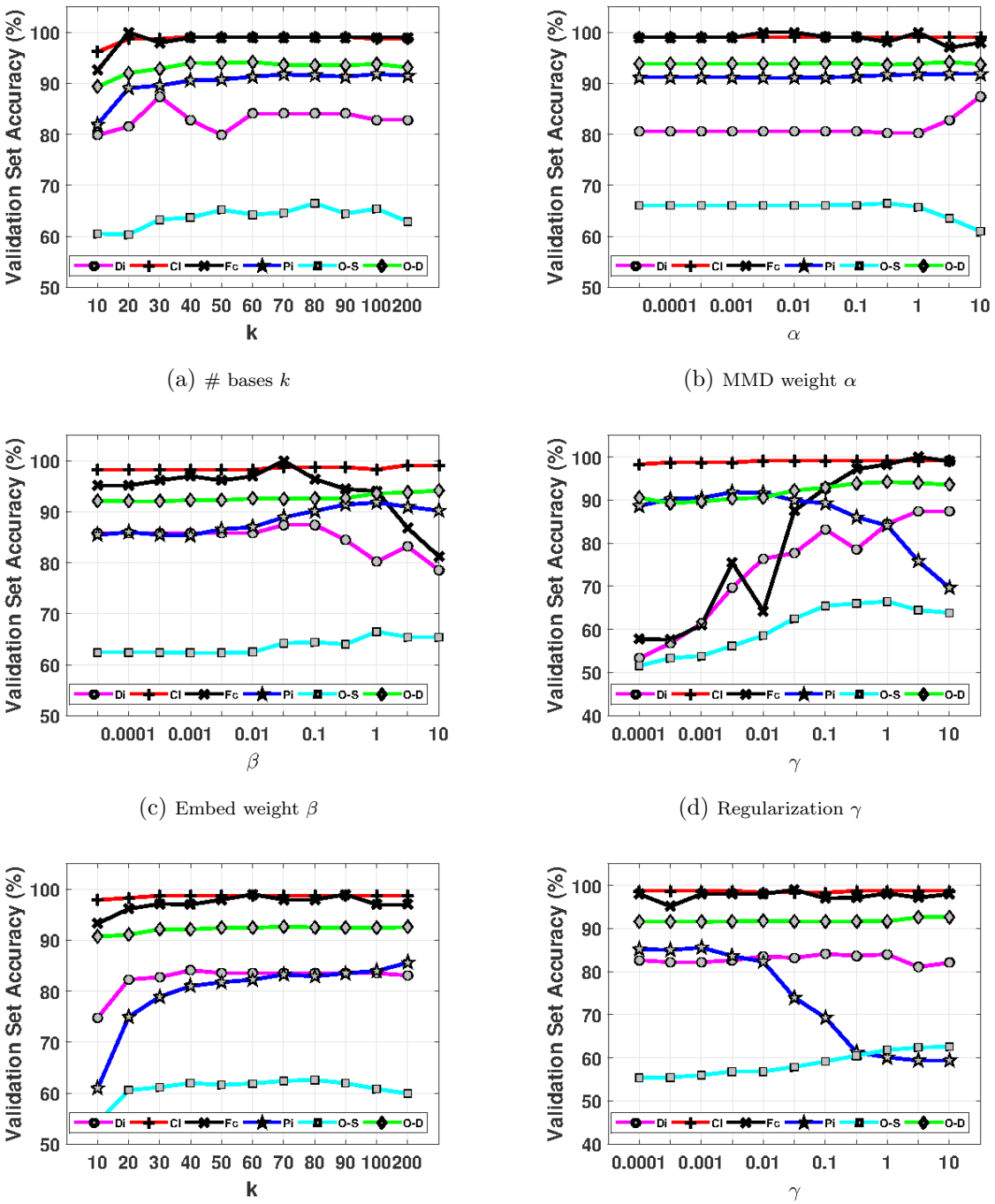
dation for the effectiveness of the cross-validation procedure. Interestingly, in Table (5.3), the JDA$_v$ has better performance than the JDA. This highlights the fact that a validation procedure helps select the optimum model parameters. Both these results demonstrate that the proposed model selection procedure is a valid technique for evaluating an unsupervised domain adaptation algorithm in the absence of target data labels. Figures (5.2a) to (5.2d), depict the variation of average validation set accuracies for the model parameters in the NET model. Similarly, Figures (5.2e) and (5.2f), depict the variation of average validation set accuracies for the model parameters in the JDA model. The optimal parameters were chosen based on the highest validation set accuracies for each of the datasets.

5.3.5 NET Algorithm Evaluation

The results of the NET algorithm are depicted under the NET column in Tables (5.3) and (5.4). The parameters used to obtain these results are depicted in Table (5.5). NET consistently outperforms non-spectral methods like SA, CA and GFK. Also, the accuracies obtained with the NET algorithm are better than any of the other spectral methods (TCA, TJM and JDA). These results confirm that embedding data based on similarity along with domain alignment improves target data classification.

Table 5.5: Parameters Used for the NET Model

Dataset	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\gamma}$	$\boldsymbol{k}$
MNIST & USPS	1.0	0.01	1.0	20
MMI & CK+	0.01	0.01	1.0	20
COIL	1.0	1.0	1.0	60
PIE	10.0	0.001	0.005	200
Ofc-SURF	1.0	1.0	1.0	20
Ofc-Deep	1.0	1.0	1.0	20



(a) # bases k

(b) MMD weight α

(c) Embed weight β

(d) Regularization γ

(e) # bases k

(f) Regularization γ

Figure 5.2: NET and JDA cross-validation study. Each of the figures depicts the recognition accuracies over the source-based validation set. When studying a parameter (say k), the remaining parameters (α, β, γ) are fixed at the optimum value. The legend is, Digit (Di), Coil (Cl), MMI&CK+ Face (Fc), PIE (Pi), Office-Caltech SURF (O-S) and Office-Caltech Deep (O-D). The top 4 figures are for the NET cross-validation study for $(k, \alpha, \beta, \gamma)$. The bottom 2 figures are for the JDA cross-validation study (k, γ). Images based on Venkateswara *et al.* (2017a)

5.4 Conclusions and Summary

The average accuracies obtained with JDA and NET using the validation set are comparable to the best accuracies with JDA and NET. This empirically validates the model selection proposition. However, there is no theoretical guarantee that the parameters selected are the best. In the absence of theoretical validation, further empirical analysis is advised when using the proposed technique for model selection.

This chapter introduced the Nonlinear Embedding Transform algorithm for unsupervised domain adaptation. The NET algorithm implemented domain adaptation by aligning the joint distributions of the source and the target using MMD. The aligned distributions were embedded onto a manifold to ensure enhanced classification. The chapter also introduced a validation procedure to estimate model parameters in the absence of labeled target data. The experimental analysis demonstrated that the NET performs favorably when compared with competitive visual domain adaptation methods across multiple datasets.

Chapter 6

HIERARCHICAL FEATURE SPACES FOR DOMAIN ADAPTATION

Deep learning models extract hierarchical patterns from large amounts of data as demonstrated in Krizhevsky *et al.* (2012). These feature representations have been found to be highly discriminative in nature. Since a deep network has a hierarchical set of layers with nonlinear functions at multiple layers, the deep network can be viewed as a function with very high nonlinearity. Since they are multilayer networks, they are also termed as hierarchical methods.

Deep learning is a relatively new area in computer vision when compared to hand-crafted (shallow) feature extraction methods like SIFT, HOG or SURF. Transfer learning based on deep networks is still in its formative years, although there has been a significant amount of work in the last few years. Deep learning has been applied to domain adaptation and has shown remarkable results compared to non-deep learning procedures. Chapter (3) discusses the different approaches to deep learning based domain adaptation. These include a direct extension of shallow methods to deep networks, as in Tzeng *et al.* (2014) and Long *et al.* (2015) and adversarial methods like Ganin *et al.* (2016).

This chapter introduces a novel unsupervised domain adaptation procedure based on a deep network that determines hash values. The deep network is trained with labeled source data and unlabeled target data to learn hash values for the inputs. The source data is trained using a supervised hash based loss and the target data is trained with an unsupervised hash based entropy loss. The chapter is organized as follows. Section (6.1) provides an introduction to the proposed hierarchical model for unsupervised domain adaptation. It outlines the main contributions of the approach

and motivates the reason for a hashing based approach. Section (6.2) describes the deep learning model, highlighting the different components and their role. This is followed by a discussion on the experiments that were conducted to evaluate the model, in Section (6.3). Two classes of experiments were conducted, (i) unsupervised domain adaptation experiments and, (ii) unsupervised hashing experiments, to evaluate all aspects of the model.

6.1 A Hierarchical Feature Model for Domain Adaptation

Conventional shallow transfer learning methods develop their models in two stages, feature extraction followed by domain adaptation. The features are fixed and then a model is trained to align the source and target domains, as in Duan *et al.* (2009) and Saenko *et al.* (2010). On the other hand, deep transfer learning procedures exploit the feature learning capabilities of deep networks to learn transferable feature representations for domain adaptation and have demonstrated impressive empirical performance. This chapter outlines a deep hashing network for unsupervised domain adaptation. In unsupervised domain adaptation, there are no labels for the target data. It is therefore difficult to train a deep network with target data in a supervised manner. The proposed model overcomes this challenge by learning hash values for the data inputs.

The explosive growth of digital data in the modern era has posed fundamental challenges regarding their storage, retrieval and computational requirements. Against this backdrop, hashing has emerged as one of the most popular and effective techniques due to its fast query speed and low memory cost Wang *et al.* (2014). Hashing techniques transform high dimensional data into compact binary codes and generate similar binary codes for similar data items. Motivated by this fact, a deep neural network is trained to output binary hash codes (instead of probability values), which

can be used for classification. There are two advantages to estimating a hash value instead of a standard probability vector in the final layer of the network:

1. Hash values enable efficient storage and retrieval of data due to their fast query speed and low memory costs.

2. During prediction, the hash code of a test sample can be compared against the hash codes of the training samples to arrive at a more robust prediction.

In this chapter, a novel deep learning framework is proposed called Domain Adaptive Hashing (DAH), to learn informative hash codes to address the problem of unsupervised domain adaptation. A unique loss function is described to train the deep network with the following components:

1. Supervised hash loss for labeled source data, which ensures that source samples belonging to the same class have similar hash codes

2. Unsupervised entropy loss for unlabeled target data, which imposes each target sample to align closely with exactly one of the source categories and be distinct from the other categories

3. A loss based on multi-kernel Maximum Mean Discrepancy (MK-MMD), which seeks to learn transferable features within the layers of the network to minimize the distribution difference between the source and target domains.

Figure (6.1) illustrates the different layers of the DAH and the components of the loss function.

6.2 Domain Adaptation Through Hashing

In unsupervised domain adaptation, as in the previous chapters, two domains are considered; *source* and *target*. The source consists of labeled data, $\mathcal{D}_s = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{n_s}$

and the target has only unlabeled data $\mathcal{D}_t = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$. The data points $\boldsymbol{x}_i^*$ belong to $\mathcal{X}$, where $\mathcal{X}$ is some input space. The corresponding labels are represented by $y_i^* \in \mathcal{Y} \coloneqq \{1, \ldots, C\}$. The paradigm of domain adaptive learning attempts to address the problem of *domain-shift* in the data, where the data distributions of the source and target are different, i.e. $P_s(X, Y) \neq P_t(X, Y)$ for random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$. The domain-disparity notwithstanding, the goal is to train a deep neural network classifier $\psi(.)$, that can predict the labels $\{\hat{y}_i^t\}_{i=1}^{n_t}$, for the target data.

The neural network is implemented as a deep CNN which consists of 5 convolution layers *conv*1 - *conv*5 and 3 fully connected layers *fc*6 - *fc*8 followed by a loss layer. In this model, a hashing layer *hash-fc*8 is introduced in place of the standard fully connected *fc*8 layer to output a binary code $\boldsymbol{h}_i$, for every data point $\boldsymbol{x}_i$, where $\boldsymbol{h}_i \in \{-1, +1\}^d$. Two loss functions direct the *hash-fc*8 layer, (i) *supervised hash loss* for the source data, (ii) *unsupervised entropy loss* for the target data. The supervised hash loss is meant to ensure the hash values are distinct and discriminatory, i.e. if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same category, their hash values $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are similar and different otherwise. The unsupervised entropy loss aligns the target hash values with source hash values based on the similarity of their feature representations. The output of the network is represented as $\psi(\boldsymbol{x})$, where $\psi(\boldsymbol{x}) \in \mathbb{R}^d$, which is converted to a hash code $\boldsymbol{h} = \text{sgn}(\psi(\boldsymbol{x}))$, where sgn(.) is the sign function. Once the network has been trained, the probability of $\boldsymbol{x}$ being assigned a label y is given by $f(\boldsymbol{x}) = p(y|\boldsymbol{h})$. The network was trained using $\mathcal{D}_s$ and $\mathcal{D}_t$ and the target data labels $\hat{y}_*^t$ were predicted using $f(.)$.

In order to address the issue of domain-shift, the feature representations of the target and the source need to be aligned. It was achieved by reducing the domain discrepancy between the source and target feature representations at multiple layers of the network. In the following subsections, the design of the domain adaptive hash

(DAH) network is discussed in detail.

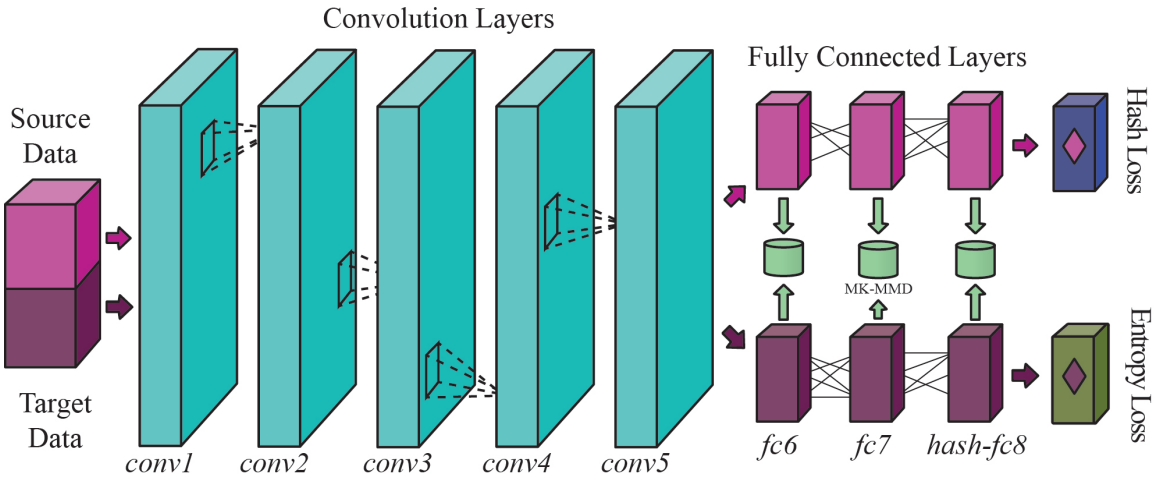


Figure 6.1: The Domain Adaptive Hash (DAH) network that outputs hash codes for the source and the target. The network is trained with a batch of source and target data. The convolution layers *conv*1 - *conv*5 and the fully connected layers *fc*6 and *fc*7 are fine tuned from the VGG-F network. The MK-MMD loss trains the DAH to learn feature representations which align the source and the target. The *hash-fc*8 layer is trained to output vectors of d dimensions. The supervised hash loss drives the DAH to estimate a unique hash value for each object category. The unsupervised entropy loss aligns the target hash values to their corresponding source categories. Best viewed in color. Image based on Venkateswara *et al.* (2017b).

6.2.1 Addressing Domain Disparity

Deep learning methods have been very successful in domain adaptation with state-of-the-art algorithms Ganin *et al.* (2016); Long *et al.* (2015, 2016b); Tzeng *et al.* (2015a) in recent years. The feature representations transition from generic to task-specific as one goes up the layers of a deep CNN Yosinski *et al.* (2014). The convolution layers *conv*1 to *conv*5 have been shown to be generic feature extractors and so, the extracted features are readily transferable, whereas the feature extractors in the fully connected layers are more task-specific and need to be adapted before they can be transferred. In the DAH algorithm, the MK-MMD loss is minimized to reduce the domain difference between the source and target feature representations for fully

connected layers, $\mathcal{F} = \{fc6, fc7, fc8\}$. Such a loss function has been used in previous research Long *et al.* (2015, 2016b). The multi-layer MK-MMD loss is given by,

$$\mathcal{M}(\mathcal{U}_s, \mathcal{U}_t) = \sum_{l \in \mathcal{F}} d_k^2(\mathcal{U}_s^l, \mathcal{U}_t^l), \tag{6.1}$$

where, $\mathcal{U}_s^l = \{\boldsymbol{u}_i^{s,l}\}_{i=1}^{n_s}$ and $\mathcal{U}_t^l = \{\boldsymbol{u}_i^{t,l}\}_{i=1}^{n_t}$ are the set of output representations for the source and target data at layer l, where $\boldsymbol{u}_i^{*,l}$ is the output representation of $\boldsymbol{x}_i^*$ for the l^{th} layer. The final layer outputs are denoted as $\mathcal{U}_s$ and $\mathcal{U}_t$. The MK-MMD measure $d_k^2(.)$ is the multi-kernel maximum mean discrepancy between the source and target representations, Gretton *et al.* (2012). For a nonlinear mapping $\phi(.)$ associated with a reproducing kernel Hilbert space $\mathcal{H}_k$ and kernel $k(.)$, where $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle$, the MMD is defined as,

$$d_k^2(\mathcal{U}_s^l, \mathcal{U}_t^l) = \Big|\Big| \mathbb{E}[\phi(\boldsymbol{u}^{s,l})] - \mathbb{E}[\phi(\boldsymbol{u}^{t,l})] \Big|\Big|_{\mathcal{H}_k}^2. \tag{6.2}$$

The characteristic kernel $k(.)$, is determined as a convex combination of κ PSD kernels, $\{k_m\}_{m=1}^{\kappa}$, $\mathcal{K} \coloneqq \left\{k : k = \sum_{m=1}^{\kappa} \beta_m k_m, \sum_{m=1}^{\kappa} \beta_m = 1, \beta_m \geq 0, \forall m\right\}$. According to Long *et al.* (2016b), β_m is to $1/\kappa$ and it works well in practice.

6.2.2 Supervised Hash Loss

The Hamming distance for a pair of hash values $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ has a unique relationship with the dot product $\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle$, given by: $\text{dist}_H(\boldsymbol{h}_i, \boldsymbol{h}_j) = \frac{1}{2}(d - \boldsymbol{h}_i^\top \boldsymbol{h}_j)$, where d is the hash length. The dot product $\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle$ can be treated as a similarity measure for the hash codes. Larger the value of the dot product (high similarity), smaller is the distance dist_H and smaller the dot product (low similarity), larger is the distance dist_H. Let $s_{ij} \in \{0, 1\}$ be the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. If $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same category, $s_{ij} = 1$ and 0, otherwise. The probability of similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ given the corresponding hash values $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, can be expressed as a

likelihood function, given by,

$$p(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j) = \begin{cases} \sigma(\boldsymbol{h}_i^\top \boldsymbol{h}_j), & s_{ij} = 1 \\ 1 - \sigma(\boldsymbol{h}_i^\top \boldsymbol{h}_j), & s_{ij} = 0, \end{cases} \tag{6.3}$$

where, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. As the dot product $\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle$ increases, the probability of $p(s_{ij} = 1|\boldsymbol{h}_i, \boldsymbol{h}_j)$ also increases, i.e., $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same category. As the dot product decreases, the probability $p(s_{ij} = 1|\boldsymbol{h}_i, \boldsymbol{h}_j)$ also decreases, i.e., $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to different categories. The $(n_s \times n_s)$ similarity matrix $\mathcal{S} = \{s_{ij}\}$, is constructed for the source data with the provided labels, where $s_{ij} = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same category and 0, otherwise. Let $\mathbf{H} = \{\boldsymbol{h}_i\}_{i=1}^{n_s}$ be the set of source data hash values. If the elements of $\mathbf{H}$ are assumed to be i.i.d., the negative log likelihood of the similarity matrix $\mathcal{S}$ given $\mathbf{H}$ can be written as,

$$\begin{aligned} \min_{\mathbf{H}} \mathcal{L}(\mathbf{H}) &= -\log\, p(\mathcal{S}|\mathbf{H}) \\ &= -\sum_{s_{ij} \in \mathcal{S}} \Big(s_{ij} \boldsymbol{h}_i^\top \boldsymbol{h}_j - \log\big(1 + \exp(\boldsymbol{h}_i^\top \boldsymbol{h}_j)\big) \Big). \end{aligned} \tag{6.4}$$

By minimizing Equation (6.4), the hash values $\mathbf{H}$ can be determined for the source data which are consistent with the similarity matrix $\mathcal{S}$. The hash loss has been used in previous research for supervised hashing Li *et al.* (2016); Zhu *et al.* (2016). Equation (6.4) is a discrete optimization problem that is challenging to solve. A relaxation is introduced on the discrete constraint $\boldsymbol{h}_i \in \{-1, +1\}^d$ by instead solving for $\boldsymbol{u}_i \in \mathbb{R}^d$, where $\mathcal{U}_s = \{\boldsymbol{u}_i\}_{i=1}^{n_s}$ is the output of the network and $\boldsymbol{u}_i = \psi(\boldsymbol{x}_i)$ (the superscript denoting the domain is dropped for ease of representation). However, the continuous relaxation gives rise to (i) approximation error, when $\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle$ is substituted with $\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle$ and, (ii) quantization error, when the resulting real codes $\boldsymbol{u}_i$ are binarized Zhu *et al.* (2016). The approximation error is accounted for by having a tanh(.) as the final activation layer of the neural network, so that the components of $\boldsymbol{u}_i$ are

bounded between -1 and $+1$. In addition, a quantization loss $||\boldsymbol{u}_i - \text{sgn}(\boldsymbol{u}_i)||_2^2$ is introduced along the lines of Gong *et al.* (2013c), where sgn(.) is the sign function. The continuous optimization problem for supervised hashing can now be outlined as;

$$\min_{\mathcal{U}_s} \mathcal{L}(\mathcal{U}_s) = -\sum_{s_{ij}\in\mathcal{S}} \Big(s_{ij}\boldsymbol{u}_i^\top \boldsymbol{u}_j - \log\big(1+\exp(\boldsymbol{u}_i^\top \boldsymbol{u}_j)\big)\Big) + \sum_{i=1}^{n_s} \big|\big|\boldsymbol{u}_i - \text{sgn}(\boldsymbol{u}_i)\big|\big|_2^2. \tag{6.5}$$

6.2.3 Unsupervised Entropy Loss

In the absence of target data labels, the similarity measure $\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle$, is used to guide the network to learn discriminative hash values for the target data. An ideal target output $\boldsymbol{u}_i^t$, needs to be similar to many of the source outputs from the j^{th} category $\big(\{\boldsymbol{u}_k^{s_j}\}_{k=1}^K\big)$. It is assumed without loss of generality, that there exist K source data points for every category j where, $j \in \{1,\ldots,C\}$ and $\boldsymbol{u}_k^{s_j}$ is the k^{th} source output from category j. In addition, $\boldsymbol{u}_i^t$ must be dissimilar to most other source outputs $\boldsymbol{u}_k^{s_l}$ belonging to a different category ($j \neq l$). Enforcing similarity with all the K data points makes for a more robust target data category assignment. A probability measure to capture this intuition is outlined as follows. Let p_{ij} be the probability that input target data point $\boldsymbol{x}_i$ is assigned to category j where,

$$p_{ij} = \frac{\sum_{k=1}^K \exp(\boldsymbol{u}_i^{t\top} \boldsymbol{u}_k^{s_j})}{\sum_{l=1}^C \sum_{k=1}^K \exp(\boldsymbol{u}_i^{t\top} \boldsymbol{u}_k^{s_l})} \tag{6.6}$$

The exp(.) is introduced for ease of differentiability and the denominator ensures $\sum_j p_{ij} = 1$. When the target data point output is similar to one category only and dissimilar to all the other categories, the probability vector $\boldsymbol{p}_i = [p_{i1},\ldots,p_{iC}]^T$ tends to be a one-hot vector. A one-hot vector can be viewed as a low entropy realization of $\boldsymbol{p}_i$. It can therefore be envisaged that all the $\boldsymbol{p}_i$ are one-hot vectors (low entropy probability vectors), where the target data point outputs are similar to source data

point outputs in one and only one category. To this end a loss is introduced to capture the entropy of the target probability vectors. The entropy loss for the network outputs is given by,

$$\mathcal{H}(\mathcal{U}_s, \mathcal{U}_t) = -\frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{j=1}^{C} p_{ij} \log(p_{ij}) \tag{6.7}$$

Minimizing the entropy loss yields probability vectors $\boldsymbol{p}_i$ that tend to be one-hot vectors, i.e., the target data point outputs are similar to source data point outputs from any one category only. Enforcing similarity with K source data points from a category, guarantees that the hash values are determined based on a common similarity between multiple source category data points and the target data point.

6.2.4 The Domain Adaptive Hash (DAH) Network

A model for deep unsupervised domain adaptation is proposed based on hashing that incorporates unsupervised domain adaptation between the source and the target in Equation (6.1), the supervised hashing for the source in Equation (6.5) and unsupervised hashing for the target in Equation (6.7) in a deep convolutional neural network. The DAH network is trained to minimize

$$\min_{\mathcal{U}} \mathcal{J} = \mathcal{L}(\mathcal{U}_s) + \gamma \mathcal{M}(\mathcal{U}_s, \mathcal{U}_t) + \eta \mathcal{H}(\mathcal{U}_s, \mathcal{U}_t), \tag{6.8}$$

where, $\mathcal{U} \coloneqq \{\mathcal{U}_s \cup \mathcal{U}_t\}$ and (γ, η) control the importance of domain adaptation (6.1) and target entropy loss (6.7) respectively. The hash values $\mathbf{H}$ are obtained from the output of the network using $\mathbf{H} = \text{sgn}(\mathcal{U})$. The loss terms in Equation (6.5) and Equation (6.7) are determined in the final layer of the network with the network output $\mathcal{U}$. The MK-MMD loss in Equation (6.1) is determined between layer outputs $\{\mathcal{U}_s^l, \mathcal{U}_t^l\}$ at each of the fully connected layers $\mathcal{F} = \{fc6, fc7, fc8\}$, where the linear time estimate for the unbiased MK-MMD was adopted as described in Gretton *et al.*

(2012) and Long *et al.* (2015). The DAH is trained using standard back-propagation. The detailed derivation of the derivative of Equation (6.8) w.r.t. $\mathcal{U}$ is provided in Appendix B.

6.2.5 Network Architecture

Owing to the paucity of images in a domain adaptation setting, the need to train a deep CNN with millions of images was circumvented by adapting the pre-trained VGG-F Chatfield *et al.* (2014) network to the DAH. The VGG-F was trained on the ImageNet 2012 dataset and it consists of 5 convolution layers (*conv*1 - *conv*5) and 3 fully connected layers (*fc*6, *fc*7, *fc*8). The hashing layer *hash-fc*8 was introduced that outputs vectors in $\mathbb{R}^d$ in the place of *fc*8. A tanh() layer was introduced To account for the hashing approximation. However, the issue of vanishing gradients Hochreiter *et al.* (2001) was encountered when using tanh() as it saturates with large inputs. Therefore the tanh() is prefaced with a batch normalization layer which prevents the tanh() from saturating. In effect, the *fc*8 is replaced by $\textit{hash-fc8} \coloneqq \{\textit{fc8} \rightarrow \textit{batch-norm} \rightarrow \textit{tanh}()\}$. The *hash-fc*8 provides greater stability when fine-tuning the learning rates than the deep hashing networks Li *et al.* (2016); Zhu *et al.* (2016). Figure (6.1) illustrates the proposed DAH network.

6.3 Experimental Analysis of the DAH Model

This section discusses the experiments that were conducted to evaluate the DAH algorithm. Since a domain adaptation technique based on hashing has been proposed, the DAH is evaluated for objection recognition accuracies for unsupervised domain adaptation and the discriminatory capabilities of the learned hash codes for unsupervised domain adaptive hashing are also studied.

6.3.1 Experimental Datasets

Office Saenko *et al.* (2010): This is currently the most popular benchmark dataset for object recognition in the domain adaptation computer vision community. The dataset consists of images of everyday objects in an office environment. It has 3 domains; `Amazon` (**A**), `Dslr` (**D**) and `Webcam` (**W**). The `Amazon` domain has images downloaded from amazon.com. The `Dslr` and `Webcam` domains have images captured using a DSLR camera and a webcam respectively. The dataset has around $4,100$ images with a majority of the images (2816 images) in the `Amazon` domain. The common evaluation protocol of different pairs of transfer tasks for this dataset Long *et al.* (2015, 2016b) was adopted. 6 transfer tasks were considered for all combinations of source and target pairs for the 3 domains. $\mathbf{A} \rightarrow \mathbf{D}$, $\mathbf{D} \rightarrow \mathbf{A}$, $\mathbf{A} \rightarrow \mathbf{W}$, $\mathbf{W} \rightarrow \mathbf{A}$, $\mathbf{D} \rightarrow \mathbf{W}$ and $\mathbf{W} \rightarrow \mathbf{D}$. $\mathbf{A} \rightarrow \mathbf{D}$ implies, **A** is the source and **D** is the target.

Office-Home[1]: This is a new dataset that was designed, developed and released to the research community as part of this dissertation. It consists of 4 domains `Art` (**Ar**), `Clipart` (**Cl**), `Product` (**Pr**) and `Real-World` (**Rw**) and 12 pairs of transfer were evaluated in a manner similar to the *Office* dataset. More details about the dataset are provided in Chapter (2).

6.3.2 Implementation Details for the DAH

The DAH was implemented using the MatConvnet framework Vedaldi and Lenc (2015). Since a pre-trained VGG-F was deployed, the weights of layers *conv*1-*conv*5, *fc*6 and *fc*7 were fine tuned. Their learning rates were set to $1/10^{th}$ the learning rate of *hash-fc*8. The learning rate was varied between 10^{-4} to 10^{-5} over 300 epochs with a momentum 0.9 and weight decay 5×10^{-4}. $K = 5$ was the number of samples from

[1] `https://hemanthdv.github.io/officehome-dataset/`

a category. Since there are 31 categories in the *Office* dataset, this results in a source batch size of $31 \times 5 = 155$. For the target batch, 155 samples were randomly selected. The total batch size turns out to be 310. For the *Office-Home* dataset, with $K = 5$ and 65 categories, the batch size turns out to be 650. The hash code length was set to $d = 64$ for most of the experiments, although other hash length were also studied. Since there is imbalance in the number of like and unlike pairs in $\mathcal{S}$, the values in the similarity matrix were set to $\mathcal{S}_{i,j} \in \{0, 10\}$ with large values for like pairs. Increasing the similarity weight of like-pairs improves the performance of DAH. The entropy loss was set to $\eta = 1$. For the MK-MMD loss, the heuristics mentioned in Gretton *et al.* (2012), were followed to determine the parameters. A Gaussian kernel was used for MMD with a bandwidth σ given by the median of the pairwise distances in the training data. To incorporate the multi-kernel, the bandwidth was varied with $\sigma_m \in [2^{-8}\sigma, 2^8\sigma]$ with a multiplicative factor of 2. γ, was estimated by validating a binary domain classifier to distinguish between source and target data points and selecting γ which gives largest error on a validation set while also comparing the performance of a source classifier on a source validation set. The target classifier $f(\boldsymbol{x}_i^t) = p(y|\boldsymbol{h}_i^t)$ is represented in terms of Equation (6.6). The target data point was assigned to the class with the largest probability, with $\hat{y}_i = \max_j(p_{ij})$ using the hash codes for the source and the target.

6.3.3 Unsupervised Domain Adaptation with DAH

The DAH was compared with state-of-the-art domain adaptation methods: (i) Geodesic Flow Kernel (**GFK**) Gong *et al.* (2012a), (ii) Transfer Component Analysis (**TCA**) Pan *et al.* (2011), (iii) Correlation Alignment (**CORAL**) Sun *et al.* (2015a) and (iv) Joint Distribution Adaptation (**JDA**) Long *et al.* (2013). The DAH was also compared with state-of-the-art deep learning methods for domain adaptation: (v)

Deep Adaptation Network (**DAN**) Long *et al.* (2015) and (vi) Domain Adversarial Neural Network (**DANN**) Ganin *et al.* (2016). For all of the shallow learning methods, deep features were used that were extracted from the $fc7$ layer of the VGG-F network that was pre-trained on the ImageNet 2012 dataset. The effect of the entropy loss on hashing for the DAH was also studied. The **DAH-e** is the DAH algorithm where η is set to zero, which implies that the target hash values are not driven to align with the source categories. The standard protocol for unsupervised domain adaptation was followed, where all the labeled source data and all the unlabeled target data was used for training.

Results and Discussion: The results are reported for the target classification in each of the transfer tasks in Tables (6.1) and (6.2), where accuracies denote the percentage of correctly classified target data samples. The hash length was set to $d = 64$ bits. The DAH algorithm consistently outperforms the baselines across all the domains for the *Office-Home* dataset. However, DANN marginally surpasses DAH for the *Office* dataset, which may be due to domain adversarial training being more effective than DAH when the categories are fewer in number. Since domain alignment is category agnostic, it is possible that the aligned domains are not classification friendly in the presence of a large number of categories. When the number of categories is large, as in *Office-Home*, DAH does best at extracting transferable features to achieve higher accuracies. It is also noted that DAH delivers better performance than DAH-e; thus, minimizing the entropy on the target data with Equation (6.7) aids in improved alignment of the source and target samples, which boosts the accuracy.

Feature Analysis: The feature representations of the penultimate layer ($fc7$) outputs using t-SNE embeddings, were also studied as in Donahue *et al.* (2014). Figure (6.2a) depicts the $\mathcal{A}$-distance between domain pairs using Deep (VGG-F), DAN and DAH features. Ben-David et al. Ben-David *et al.* (2010) defined $\mathcal{A}$-distance as the

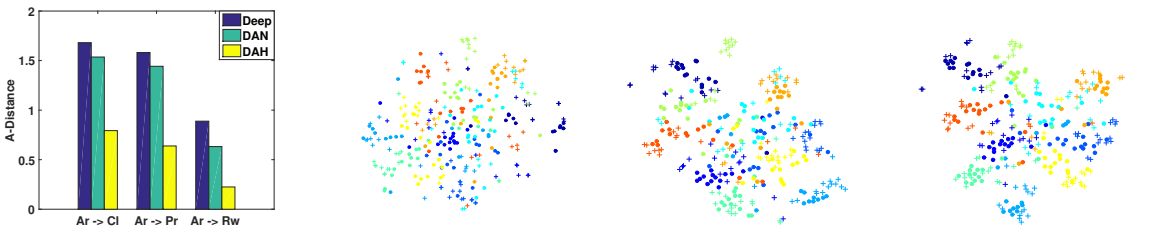
Table 6.1: Recognition accuracies (%) for domain adaptation experiments on the *Office* dataset. {`Amazon` (A), `Dslr` (D), `Webcam` (W)}. A→W implies A is source and W is target.

Expt.	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
GFK	48.59	52.08	41.83	89.18	49.04	93.17	62.32
TCA	51.00	49.43	48.12	93.08	48.83	96.79	64.54
CORAL	54.42	51.70	48.26	95.97	47.27	98.59	66.04
JDA	59.24	58.62	51.35	96.86	52.34	97.79	69.37
DAN	67.04	67.80	50.36	95.85	52.33	99.40	72.13
DANN	72.89	72.70	56.25	96.48	53.20	99.40	75.15
DAH-e	66.27	66.16	55.97	94.59	53.91	96.99	72.31
DAH	66.47	68.30	55.54	96.10	53.02	98.80	73.04

Table 6.2: Recognition accuracies (%) for domain adaptation experiments on the *Office-Home* dataset. {`Art` (Ar), `Clipart` (Cl), `Product` (Pr), `Real-World` (Rw)}. Ar→Cl implies Ar is source and Cl is target.

Expt.	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg.
GFK	21.60	31.72	38.83	21.63	34.94	34.20	24.52	25.73	42.92	32.88	28.96	50.89	32.40
TCA	19.93	32.08	35.71	19.00	31.36	31.74	21.92	23.64	42.12	30.74	27.15	48.68	30.34
CORAL	27.10	36.16	44.32	26.08	40.03	40.33	27.77	30.54	50.61	38.48	36.36	57.11	37.91
JDA	25.34	35.98	42.94	24.52	40.19	40.90	25.96	32.72	49.25	35.10	35.35	55.35	36.97
DAN	30.66	42.17	54.13	32.83	47.59	49.78	29.07	34.05	56.70	43.58	38.25	62.73	43.46
DANN	33.33	42.96	54.42	32.26	49.13	49.76	30.49	38.14	56.76	44.71	42.66	64.65	44.94
DAH-e	29.23	35.71	48.29	33.79	48.23	47.49	29.87	38.76	55.63	41.16	44.99	59.07	42.69
DAH	31.64	40.75	51.73	34.69	51.93	52.79	29.91	39.63	60.71	44.99	45.13	62.54	45.54

distance between two domains that can be viewed as the discrepancy between two domains. Although it is difficult to estimate its exact value, an approximate distance measure is given by $2(1-2\epsilon)$, where ϵ is the generalization error for a binary classifier trained to distinguish between the two domains. A LIBLINEAR SVM, Fan *et al.* (2008), classifier with 5-fold cross-validation was applied to estimate ϵ. Figure (6.2a) indicates that the DAH features have the least discrepancy between the source and target compared to DAN and Deep features. This is also confirmed with the t-SNE embeddings in Figures (6.2b-6.2d). The Deep features show very little overlap between the domains and the categories depict minimal clustering. Domain overlap and



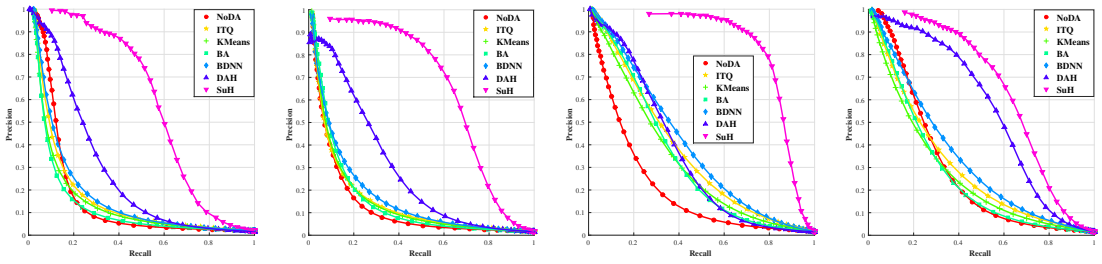
(a) $\mathcal{A}$-Distance (b) Deep Features (**Ar**,**Cl**) (c) DAN Features (**Ar**,**Cl**) (d) DAH Features (**Ar**,**Cl**)

Figure 6.2: Feature analysis of *fc*7 layer. (a) $\mathcal{A}$-distances for Deep, DAN and DAH, (b), (c) and (d) t-SNE embeddings for 10 categories from `Art` (•) and `Clipart`(+) domains. Best viewed in color. Images based on Venkateswara *et al.* (2017b).

clustering improves in DAN and DAH features, with DAH providing the best visualizations. This corroborates the efficacy of the DAH algorithm to exploit the feature learning capabilities of deep neural networks to learn representative hash codes and achieve domain adaptation.

6.3.4 Unsupervised Domain Adaptive Hashing

This subsection demonstrates the performance of the DAH algorithm to generate compact and efficient hash codes from the data, for classifying unseen test instances, when no labels are available. This problem has been addressed in the literature, with promising empirical results by Carreira-Perpinán and Raziperchikolaei (2015); Do *et al.* (2016) and Gong and Lazebnik (2011). However, in a real-world setting, labels may be available from a different, but related (source) domain; a strategy to utilize the labeled data from the source domain, to learn representative hash codes for the target domain, is therefore of immense practical importance. The following scenarios were considered to address this real-world challenge: (i) No labels are available for a given dataset and the hash codes need to be learned in a completely unsupervised manner. DAH was evaluated against baseline unsupervised hashing methods (**ITQ**) by Gong and Lazebnik (2011) and (**KMeans**) by He *et al.* (2013) and also state-of-the-art

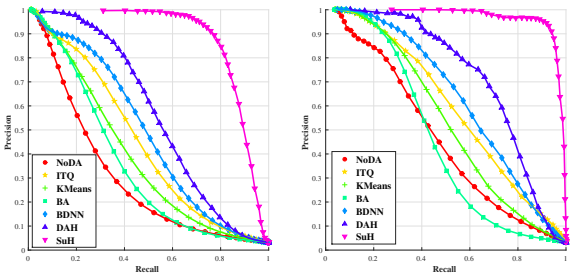


(a) Art (b) Clipart (c) Product (d) Real-World

Figure 6.3: Precision-Recall curves @64 bits for the ***Office-Home*** dataset. Comparison of hashing without domain adaptation (**NoDA**), shallow unsupervised hashing (**ITQ**, **KMeans**), state-of-the-art deep unsupervised hashing (**BA**, **BDNN**), unsupervised domain adaptive hashing (**DAH**) and supervised hashing (**SuH**). Best viewed in color. Images based on Venkateswara *et al.* (2017b).

methods for unsupervised hashing (**BA**) by Carreira-Perpinán and Raziperchikolaei (2015) and (**BDNN**) by Do *et al.* (2016). (ii) Labeled data is available from a different, but related source domain. A hashing model was trained on the labeled source data and was used to learn hash codes for the target data. This method is referred to as **NoDA**, as no domain adaptation is performed. The deep pairwise-supervised hashing (DPSH) algorithm by Li *et al.* (2016), was deployed to train a deep network with the source data and the network was applied to generate hash codes for the target data. (iii) Labeled data is available from a different, but related source domain and the DAH formulation is applied to learn hash codes for the target domain, by reducing domain disparity. (iv) Labeled data is available in the target domain. This method falls under supervised hashing (**SuH**) (as it uses labeled data in the target domain to learn hash codes in the same domain) and denotes the upper bound on the performance. It was included to compare the performance of unsupervised hashing algorithms relative to the supervised algorithm. The DPSH algorithm by Li *et al.* (2016), was used to train a deep network on the target data and generate hash codes on a validation subset.

Results and Discussion: Precision-Recall curves and the mean average precision



(a) Amazon (b) Webcam

Figure 6.4: Precision-Recall curves @64 bits for the ***Office*** dataset. Comparison of hashing without domain adaptation (**NoDA**), shallow unsupervised hashing (**ITQ**, **KMeans**), state-of-the-art deep unsupervised hashing (**BA**, **BDNN**), unsupervised domain adaptive hashing (**DAH**) and supervised hashing (**SuH**). Best viewed in color. Images based on Venkateswara *et al.* (2017b).

Table 6.3: Mean average precision @64 bits. For the NoDA and DAH results, `Art` is the source domain for `Clipart`, `Product` and `Real-World` and `Clipart` is the source domain for `Art`. Similarly, `Amazon` and `Webcam` are source target pairs.

Expt.	NoDA	ITQ	KMeans	BA	BDNN	DAH	SuH
`Amazon`	0.324	0.465	0.403	0.367	0.491	0.582	0.830
`Webcam`	0.511	0.652	0.558	0.480	0.656	0.717	0.939
`Art`	0.155	0.191	0.170	0.156	0.193	0.302	0.492
`Clipart`	0.160	0.195	0.178	0.179	0.206	0.333	0.622
`Product`	0.239	0.393	0.341	0.349	0.407	0.414	0.774
`Real-World`	0.281	0.323	0.279	0.273	0.336	0.533	0.586
Avg.	0.278	0.370	0.322	0.301	0.382	0.480	0.707

(mAP) measures were used to evaluate the efficacy of the hashing methods, similar to previous research in Carreira-Perpinán and Raziperchikolaei (2015); Do *et al.* (2016) and Gong and Lazebnik (2011). The results are depicted in Figures (6.3) and (6.4) (precision-recall curves) and Table (6.3) (mAP values), for hashing with code length $d = 64$ bits. For the sake of brevity, the results with `Dslr` are dropped, as it is very similar to `Webcam`, with little domain difference. It can be verified that the NoDA has the poorest performance due to domain mismatch. This demonstrates the fact that domain disparity needs to be considered before deploying a hashing network to ex-

tract hash codes. The unsupervised hashing methods ITQ, KMeans, BA and BDNN perform slightly better compared to NoDA. The proposed DAH algorithm encompasses hash code learning and domain adaptation in a single integrated framework. It is thus able to leverage the labeled data in the source domain in a meaningful manner in order to learn efficient hash codes for the target domain. This accounts for its improved performance, as is evident in Figures (6.3) and (6.4) and Table (6.3). The supervised hashing technique (SuH), uses labels from the target and therefore depicts the best performance. The proposed DAH framework consistently delivers the best performance relative to SuH, when compared with the other hashing procedures. This demonstrates the merit of the DAH framework in learning representative hash codes by utilizing labeled data from a different domain. Such a framework is bound to be immensely useful in a real-world setting.

6.3.5 Effect of Batch-size for Linear-MMD

The domain alignment between the source and the target is achieved using multi-kernel maximum mean discrepancy (MK-MMD). The outputs of the fully connected layers $\mathcal{F} = \{fc6, fc7, fc8\}$ are aligned using a linear MK-MMD loss. The MMD loss outlined in Section (3.3) is based on Gretton *et al.* (2007). The MMD gives a measure of discrepancy between two i.i.d. datasets. However, the MMD measure described in Gretton *et al.* (2007) requires all the samples from the source and the target to estimate the discrepancy and it is also quadratic in complexity. The DAH is trained using back-propagation, an iterative algorithm where only a batch of data points are available at any given instant. A batch-wise MMD would be a biased estimate of the MMD for the entire data and also expensive for back-propagation (since it is quadratic). The DAH therefore uses an online linear version of the MMD. The online linear version of the MMD outlined in Gretton *et al.* (2012), provides an

empirical estimate for the MMD with linear complexity and works in an online setting; where all the data samples are not available at once. The linear-MMD estimates the discrepancy between the source and the target using only a batch of data points at a time unlike the MMD which needs all the data samples from the two domains. The DAH applies the linear-MMD over every batch of data for the fully connected layers. An experiment was conducted to study the effect of batch size on domain adaptation using the linear-MMD. In the DAH the batch size is controlled by the value of K (the number of source data points for every category). Varying the values of K is bound to vary the target recognition accuracies since the number of source samples available for supervised hashing will vary. In order to study the effect of batch size on the recognition accuracies, K has to remain constant across all batch sizes. However, it is not possible to vary the batch size without changing K. Since the goal is to study the effect of batch size on domain alignment, hashing and entropy loss can be replaced with regular cross-entropy loss. The resulting model is equivalent to the DAN in Long *et al.* (2015), where domain alignment is achieved with MK-MMD and the final layer has a standard cross-entropy loss. Table (6.4) outlines the target recognition accuracies when different batch sizes are used for this setting. It can be observed that varying the batch size has no effect on the recognition accuracies when using the linear-MMD for domain alignment. It can be concluded that the linear-MMD is a consistent online empirical estimate for MMD.

6.3.6 Classification Experiments with Varying Hash Size

The previous subsections discussed the results for unsupervised domain adaptation based object recognition with $d = 64$ bits. Here, the classification results with $d = 16$ (DAH-16) and $d = 128$ (DAH-128) bits for the *Office-Home* dataset are outlined in Table (6.5). The (DAH-64), DAN and DANN results are also presented for

Table 6.4: Effect of batch size on domain alignment when using linear MMD. The values represent recognition accuracies (%) for domain adaptation experiments on the *Office-Home* dataset. {`Art` (Ar), `Clipart` (Cl), `Product` (Pr), `Real-World` (Rw)}. Ar→Cl implies Ar is source and Cl is target.

Batch Size	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg.
32	31.68	42.27	52.57	32.46	48.09	49.39	30.79	32.80	56.82	44.08	38.32	62.16	43.45
64	30.66	42.17	54.13	32.83	47.59	49.78	29.07	34.05	56.70	43.58	38.25	62.73	43.46
128	30.32	43.59	53.59	32.96	47.36	50.00	30.21	33.48	56.11	44.38	38.23	63.57	43.65
512	31.41	42.61	53.61	32.13	48.14	50.45	29.88	33.43	55.16	44.33	37.80	62.91	43.49
1024	31.30	42.80	54.07	32.38	48.11	50.18	30.17	33.75	56.41	43.87	37.95	63.41	43.70

Table 6.5: Recognition accuracies (%) for domain adaptation experiments on the *Office-Home* dataset. {`Art` (Ar), `Clipart` (Cl), `Product` (Pr), `Real-World` (Rw)}. Ar→Cl implies Ar is source and Cl is target.

Expt.	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg.
DAN	30.66	42.17	54.13	32.83	47.59	49.78	29.07	34.05	56.70	43.58	38.25	62.73	43.46
DANN	33.33	42.96	54.42	32.26	49.13	49.76	30.49	38.14	56.76	44.71	42.66	64.65	44.94
DAH-16	23.83	30.32	40.14	25.67	38.79	33.26	20.11	27.72	40.90	32.63	25.54	37.46	31.36
DAH-64	31.64	40.75	51.73	34.69	51.93	52.79	29.91	39.63	60.71	44.99	45.13	62.54	45.54
DAH-128	32.58	40.64	52.40	35.72	52.80	52.12	30.94	41.31	59.31	45.65	46.67	64.97	46.26

comparison. There is an increase in the average recognition accuracy for $d = 128$ bits compared to $d = 64$ bits because of the increased capacity in representation. As expected, $d = 16$ has a lower recognition accuracy.

6.3.7 Hashing Experiments with Varying Hash Size

The unsupervised domain adaptive hashing results for $d = 16$ and $d = 128$ bits are provided in Figures (6.5) and (6.6) respectively. In Tables (6.6) and (6.7), the corresponding mAP values are outlined. Similar trends are observed for both $d = 16$ and $d = 128$ bits compared to $d = 64$ bits. It is interesting to note that with increase in bit size d, the mAP does not necessarily increase. Table (6.7) ($d = 128$) has its mAP values lower than those for $d = 64$ (see Table (6.3)) for all the hashing methods. It can be surmised that merely increasing the hash code length does not always improve mAP scores. Also, the mAP values for `Real-World` for $d = 128$ bits has DAH performing better than SuH. This indicates that in some cases domain

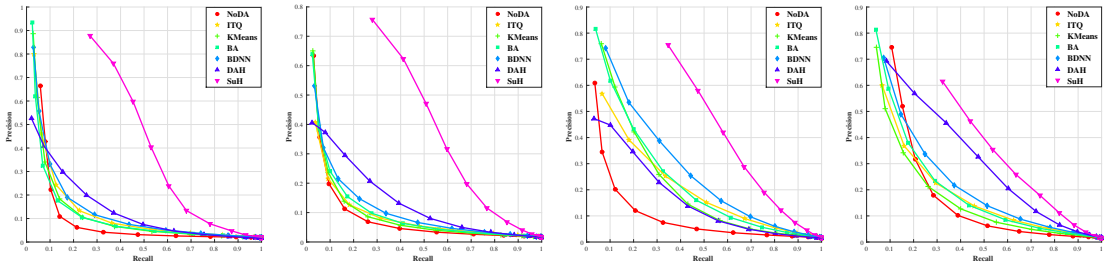
Table 6.6: Mean average precision @16 bits. Notation similar to Table (6.3).

Expt.	NoDA	ITQ	KMeans	BA	BDNN	DAH	SuH
Art	0.102	0.147	0.133	0.131	0.151	0.207	0.381
Clipart	0.110	0.120	0.116	0.123	0.138	0.211	0.412
Product	0.134	0.253	0.241	0.253	0.313	0.257	0.459
Real-World	0.193	0.225	0.195	0.216	0.248	0.371	0.400
Avg.	0.135	0.186	0.171	0.181	0.212	0.262	0.413

Table 6.7: Mean average precision @128 bits. Notation similar to Table (6.3).

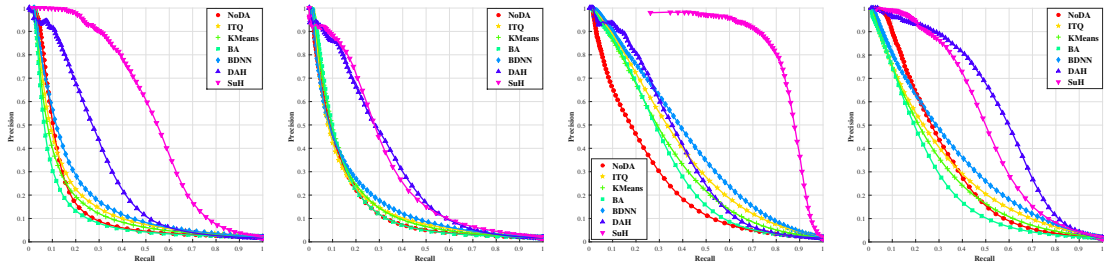
Expt.	NoDA	ITQ	KMeans	BA	BDNN	DAH	SuH
Art	0.154	0.202	0.175	0.148	0.207	0.314	0.444
Clipart	0.186	0.210	0.196	0.187	0.213	0.350	0.346
Product	0.279	0.416	0.356	0.336	0.432	0.424	0.792
Real-World	0.308	0.343	0.289	0.258	0.348	0.544	0.458
Avg.	0.232	0.293	0.254	0.232	0.300	0.408	0.510

adaptation helps in learning a better generalized model.



(a) Art (b) Clipart (c) Product (d) Real-World

Figure 6.5: Precision-Recall curves @16 bits for the ***Office-Home*** dataset. Notation similar to Figure (6.3).



(a) Art (b) Clipart (c) Product (d) Real-World

Figure 6.6: Precision-Recall curves @128 bits for the ***Office-Home*** dataset. Notation similar to Figure (6.3).

6.4 Conclusions and Summary

The DAH model introduced hashing in order to determine a loss for the target data. There is also another advantage to hashing that was not stressed upon in the chapter. The hash values can be used to boost predictions. The hash output for a test sample can be compared with the hash values of multiple training samples in order to determine its label very accurately. This perhaps makes the prediction more robust than having a softmax layer based prediction.

Another interesting direction in future work for domain adaptation, would be to attempt domain alignment in a category cognizant manner. Current approaches align the marginal probability distributions $P(X)$, of the domains. A deep learning framework that can impute the labels of the target data and align the joint distributions

$P(X, Y)$ (data and labels), could potentially be a breakthrough in deep learning based domain adaptation.

In summary, this chapter introduced a novel domain adaptive hashing (DAH) framework which can exploit the feature learning capabilities of deep neural networks to learn efficient hash codes for unsupervised domain adaptation. The DAH framework solves two important practical problems: the problem of weak supervision or insufficient labels (through domain adaptation) and the problem of memory and computation requirements (through hashing). Thus, two practical challenges are addressed through a single integrated framework. This research is the first of its kind to integrate hash code learning with unsupervised domain adaptation. The extensive empirical results corroborate the competency of the framework for real-world image recognition applications.

Chapter 7

FEATURE SELECTION BASED DOMAIN ADAPTATION

One of the strategies for domain adaptation is *feature matching* through dimensionality reduction, where the distribution difference between the source and target datasets is minimized by discovering a shared feature representation between the two datasets (Pan *et al.* (2008) and Long *et al.* (2014)). Along similar lines, this chapter discusses the idea of selecting features between the source and target dataset, such that their distributions become more similar. The idea of feature selection for domain adaptation has not been explored except in a handful of cases like Helleputte and Dupont (2009) and Uguroglu and Carbonell (2011). In the area of multitask learning, feature selection has been studied in great detail through sparsity inducing norms, in order to increase the overlap between tasks (domains), Evgeniou and Pontil (2007), Obozinski *et al.* (2010), and Liu *et al.* (2009). Since multitasking is a form of transfer learning closely related to domain adaptation, the concept of feature selection can be extended to domain adaptation.

The chapter is organized in two parts. The first part outlines a standard feature selection procedure based on information gain. In Section (7.1), a feature selection technique based on conditional mutual information, is proposed. This section arrives at an efficient approximate solution for the NP-hard problem of feature selection based on solutions from related NP-hard problems like k-Sparse-PCA and Densest-k-Subgraph. This is followed by experiments, described in Section (7.2), of feature selection on standard publicly available datasets. The second part proposes a non-linear feature selection model for unsupervised domain adaptation in Section (7.3). Some preliminary results for this proposed model are outlined in Section (7.4).

7.1 Feature Selection Based on Information Gain

Mutual information (MI) is a probabilistic measure that captures the 'correlation' between random variables (see Figure (7.1)). Whereas standard correlation captures linear relationships between variables, mutual information can capture non-linear dependencies between variables Rodriguez-Lujan *et al.* (2010). The steps involved in arriving at a mutual information based formulation for feature selection is discussed below for a classification setting.

In the standard classification setting, data is independent and identically distributed (i.i.d), with $\mathcal{D} = \{(\boldsymbol{x}^i, y^i); i = 1 \dots n\}$. Each data point $\boldsymbol{x}^i \in \mathbb{R}^d$, is regarded as an instance of a set of d continuous random variables $X = \{X_1, X_2, \dots, X_d\}$. The sample index i is treated as a superscript for the initial part of the discussion and the subscript will be used to refer to the feature index. The terms *features* or *variables* are used interchangeably to denote X or its subsets. The dependent class label y^i, is considered to be an instance of a discrete random variable Y, that takes values in $[1, \dots, c]$. $\mathbb{S}$ is defined to be a subset of k feature indices, where $1 \leq k \leq d$ and $X_{\mathbb{S}}$, the subset of k features indexed by $\mathbb{S}$. Likewise, $\tilde{\mathbb{S}}$ is the subset of left over indices, i.e. $\{1, \dots, d\} \backslash \mathbb{S}$ and $X_{\tilde{\mathbb{S}}}$ is the subset of leftover $(d - k)$ features indexed by $\tilde{\mathbb{S}}$, i.e. $X \backslash X_{\mathbb{S}}$. The posterior probability distribution on the dataset $\mathcal{D}$, is given by $p(Y|X)$. The problem of feature selection is stated as follows.

Problem 7.1.1. *To estimate the optimal subset $\mathbb{S}^* \subset \{1, \dots, d\}$, of feature indices, such that $p(Y|X_{\mathbb{S}^*})$ is a good approximation to $p(Y|X)$.*

Theorem 7.1.1. *$p(Y|X) = p(Y|X_{\mathbb{S}})$, if and only if mutual information $I(X;Y) = I(X_{\mathbb{S}};Y)$*

Proof. Joint probability $p(X, Y)$ can be factored as

$$\begin{aligned} p(X, Y) &= p(X)p(Y|X) = p(X)p(Y|X_{\mathbb{S}}) \\ &= p(X_{\bar{\mathbb{S}}})p(X_{\mathbb{S}}|X_{\bar{\mathbb{S}}})p(Y|X_{\mathbb{S}}) \end{aligned} \tag{7.1}$$

When viewed as elements in a Bayesian network, the random variables can be thought to form a Markov Chain $X_{\bar{\mathbb{S}}} \rightarrow X_{\mathbb{S}} \rightarrow Y$ where Y is conditionally independent of $X_{\bar{\mathbb{S}}}$ given $X_{\mathbb{S}}$. Therefore, the conditional mutual information $I(X_{\bar{\mathbb{S}}}; Y|X_{\mathbb{S}}) = 0$. With $I(X; Y) = I(X_{\mathbb{S}}; Y) + I(X_{\bar{\mathbb{S}}}; Y|X_{\mathbb{S}})$, the necessary condition is proved. The statements in the proof are also true in the reverse order. ■

$I(X; Y)$ depends only on $\mathcal{D}$ and is a constant that is shared between $I(X_{\mathbb{S}}; Y)$ and $I(X_{\bar{\mathbb{S}}}; Y|X_{\mathbb{S}})$. It is very likely that $I(X_{\bar{\mathbb{S}}}; Y|X_{\mathbb{S}})$ is a non-zero value for every $\mathbb{S}$ except for the most trivial case with $X_{\bar{\mathbb{S}}} = \emptyset$. For a fixed value of k, the optimal subset $\mathbb{S}$ is estimated to maximize $I(X_{\mathbb{S}}; Y)$. Theorem (7.1.1) provides the reason for choosing a mutual information based approach to determining the optimal $\mathbb{S}^*$. In the following subsection the challenges to estimating $\mathbb{S}^*$ are discussed and the assumptions made to arrive at an approximate solution are outlined.

7.1.1 The Binary Quadratic Problem

Given a subset of features $X_{\mathbb{S}}$, in order to evaluate $I(X_{\mathbb{S}}; Y)$, the joint probability distribution $p(X_{\mathbb{S}}, Y)$ needs to be estimated. If the random variables X_i were binary, it would still require $m = 2^d$ data points to estimate $p(X_{\mathbb{S}}, Y)$, which is impractical even for a moderate feature size d. To simplify the estimation process of the joint distribution and make it tractable, an assumption of conditional independence between features in the spirit of Naïve Bayes is introduced Brown *et al.* (2012). A new term $\mathbb{S}_i$ is defined as the subset $\mathbb{S}$ without the index i also denoted as $\mathbb{S}\backslash\{i\}$. The assumption that will help make the joint distribution tractable is defined as:

Assumption 7.1.1. *For a set of selected features* $\{X_{\mathbb{S}_i} \cup X_i\}$*, the features* $X_{\mathbb{S}_i}$ *are conditionally independent and class-conditionally independent given* X_i

$$p(X_{\mathbb{S}_i}|X_i) = \prod_{j \in \mathbb{S}_i} p(X_j|X_i) \tag{7.2}$$

$$p(X_{\mathbb{S}_i}|X_i, Y) = \prod_{j \in \mathbb{S}_i} p(X_j|X_i, Y) \tag{7.3}$$

Theorem 7.1.2. *If Assumptions in (7.2) and (7.3) are true, and* $X_i \in X_{\mathbb{S}}$ *then,* $I(X_{\mathbb{S}}; Y) = I(X_i; Y) + \sum_{j \in \mathbb{S}_i} I(X_j; Y|X_i)$

Proof. The mutal information between $X_{\mathbb{S}}$ and Y is given by:

$$\begin{aligned}
I(X_{\mathbb{S}}; Y) &= I(X_{\mathbb{S}_i}, X_i; Y) \\
&= I(X_{\mathbb{S}_i}; Y) + I(X_i; Y|X_{\mathbb{S}_i}) \\
&= I(X_{\mathbb{S}_i}; Y) + I(X_i; Y) - I(X_i; X_{\mathbb{S}_i}) + I(X_i; X_{\mathbb{S}_i}|Y) \\
&= I(X_{\mathbb{S}_i}; Y) + I(X_i; Y) - H(X_{\mathbb{S}_i}) + H(X_{\mathbb{S}_i}|X_i) \\
&\quad + H(X_{\mathbb{S}_i}|Y) - H(X_{\mathbb{S}_i}|X_i, Y) \\
&= I(X_i; Y) + H(X_{\mathbb{S}_i}|X_i) - H(X_{\mathbb{S}_i}|X_i, Y)
\end{aligned} \tag{7.4}$$

In the above derivation, the mutual information chain rule $I(A, B; C) = I(A; C) + I(B; C|A)$ is first applied followed by the application of the mutual information rule $I(A; B|C) - I(A; B) = I(A; C|B) - I(A; C)$, and the two trailing mutual information terms are then expressed in terms of entropy. In the proof statements below, applying Assumption in (7.2) and (7.3) to (7.4), the high order entropy terms can be replaced with summations and reduced further using $H(X_j|X_i) - H(X_j|X_i, Y) = I(X_j, Y|X_i)$

$$\begin{aligned}
I(X_i X_{\mathbb{S}_i}; Y) &\approx I(X_i; Y) + \sum_{j \in \mathbb{S}_i} H(X_j|X_i) - \sum_{j \in \mathbb{S}_i} H(X_j|X_i, Y) \\
&= I(X_i; Y) + \sum_{j \in \mathbb{S}_i} I(X_j; Y|X_i)
\end{aligned} \tag{7.5}$$

■

Since $\mathbb{S}$ needs to be selected such that it maximizes Equation (7.5), for every X_i, the global feature selection problem can be formulated as below,

$$\mathbb{S} = \underset{\{\mathbb{S}|X_{\mathbb{S}} \subset X\}, |\mathbb{S}|=k}{\operatorname{argmax}} \sum_{i \in \mathbb{S}} \big[I(X_i; Y) + \sum_{j \in \mathbb{S}_i} I(X_j; Y | X_i) \big] \tag{7.6}$$

This is equivalent to the constrained Binary Quadratic problem,

$$\max_{\mathbf{x}} \{ \mathbf{x}^\top \mathbf{Q} \mathbf{x} \} \;\; \text{s.t. } \mathbf{x} \in \{0,1\}^d, \;\; ||\mathbf{x}||_1 = k, \tag{7.7}$$

where, $\mathbf{Q}$ is a $[d \times d]$ non-negative matrix with $Q_{ii} = I(X_i; Y)$ and $Q_{ij} = I(X_j; Y | X_i)$ and the non-zero indices of the solution $\mathbf{x}$ constitute $\mathbb{S}$. Applying the conditional mutual information criteria, $\mathbb{S}$ can also be estimated using a greedy feature selection process. Beginning with $\mathbb{S} = \operatorname{argmax}_i Q_{ii} \; \forall i$, $\mathbb{S}$, is updated iteratively until $|\mathbb{S}| = k$ using,

$$\mathbb{S} \leftarrow \big\{ \mathbb{S} \cup \underset{i, \; i \notin \mathbb{S}}{\operatorname{argmax}} [Q_{ii} + \sum_{j \in \mathbb{S}} Q_{ij}] \big\} \tag{7.8}$$

7.1.2 Solution to the Binary Quadratic Problem

The aim is to solve the following constrained binary quadratic problem:

$$\max_{\mathbf{x}} \mathbf{x}^\top \mathbf{Q} \mathbf{x} \;\; \text{s.t. } \mathbf{x} \in \{0,1\}^d, \;\; \|\mathbf{x}\|_1 = k, \tag{BQP}$$

where $\mathbf{Q}$ is a symmetric and possibly indefinite matrix with non-negtiave elements, i.e., $Q_{ij} \geq 0$ for all $1 \leq i, j \leq d$. Note that the upper bound of the maximum value could be approximated by the largest eigenvalue of $\mathbf{Q}$. Although this problem is well defined, it is highly nonconvex due to the nonconvex constraint on $\mathbf{x}$. And it is also known that this binary quadratic problem is NP-hard Garey and Johnson (1990). So it is difficult to find the optimal value in practice. Therefore, an approximate solution to the BQP is estimated. The proposed binary quadratic problem is also closely

related to several other NP-hard problems, including k-Sparse-PCA Yuan and Zhang (2013); Papailiopoulos *et al.* (2013), Densest-k-Subgraph Papailiopoulos *et al.* (2014), Best Principal Sub-matrix and MAX-CUT Goemans and Williamson (1995). Many approximation methods have been proposed to solve these problems. Those methods include linear relaxation, spectral relaxation, semidefinite programming, truncated power method and low rank bilinear approximation. A linear approximation to Equation (7.7) is discussed below along with bounds for the approximation followed by other approximations to this NP-hard problem.

Linear Relaxation: The proposed BQP problem can be rewritten as follows:

$$\begin{aligned} &\max \sum_{i=1}^{d}\sum_{j=1}^{d} Q_{ij}x_i x_j \\ &\text{s.t.} \quad x_i \in \{0,1\}, \quad \forall 1 \leq i \leq d, \quad \sum_{i=1}^{d} x_i = k. \end{aligned} \tag{7.9}$$

By introducing a new variable w_{ij}, the quadratic term can be linearized. Specifically, it can formulated as follows:

$$\begin{aligned} &\max \sum_{i=1}^{d}\sum_{j=1}^{d} Q_{ij}w_{ij} \\ &\text{s.t.} \quad 2w_{ij} \leq x_i + x_j, \quad \sum_{i=1}^{d} x_i = k, \quad x_i, w_{ij} \in \{0,1\}, \quad \forall 1 \leq i,j \leq d. \end{aligned} \tag{LP1}$$

It can be shown that LP1 is equivalent to BQP, which is also NP-hard. Note that LP1 increases the size of the problem by adding $O(d^2)$ variables and $O(d^2)$ constraints. Most linear programs approximate the solution. Global optimal solutions can be estimated applying Branch and Bound techniques (as in CPLEX IBM-ILOG-CPLEX (2013)). These procedures are however not computationally efficient. By relaxing $w_{ij} \in [0,1]$, it is not difficult to show that one of the optimality conditions is $2w_{ij} = x_i + x_j$. This dramatically reduces the redundant constraints. The following linear

relaxation formulation can then be proposed:

$$\begin{aligned} \textbf{Linear} = \max \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{1}{2} Q_{ij}(x_i + x_j) = \|\mathbf{Q}\mathbf{x}\|_1 \\ \text{s.t.} \quad \|\mathbf{x}\|_1 = k, \quad \mathbf{x} \in \{0,1\}^d. \end{aligned} \tag{LP2}$$

Since $Q_{ij} \geq 0$, the maximum value for **Linear** is equivalent to the k largest column (or row) sum of $\mathbf{Q}$. Although LP2 is not equivalent to LP1, it is a very good approximation to LP1 when k is small and it can be solved efficiently. It can be proved that **Linear** guarantees a good lower bound to BQP (proof in Appendix A). The solution from **Linear** can also be used to initialize the input for more advanced algorithms.

Truncated Power Method: Truncated power (**TPower**) method aims to find the largest k-sparse eigenvector. Given a positive semidefinite matrix A, the largest k-sparse eigenvalue can be defined as follows Yuan and Zhang (2013):

$$\lambda_{\max}(A, k) = \max \ \mathbf{x}^\top A\mathbf{x}, \quad \text{s.t.} \ \ \|\mathbf{x}\| = 1, \ \ \|\mathbf{x}\|_0 \leq k \tag{7.10}$$

Matrix A is required to be positive semidefinte, but **TPower** method can be extended to deal with general symmetric matrices by setting $A \leftarrow (A + \tilde{\lambda} I_{d\times d})$ where $\tilde{\lambda} > 0$ such that $(A + \tilde{\lambda} I_{d\times d}) \in \mathbb{S}_+^d$. The truncated power method can be solved in an iterative manner, as follows. Starting from an initial k-sparse vector $\mathbf{x}_0$, at each iteration t, A is multiplied by the vector $\mathbf{x}_{t-1}$ and the entries of $A\mathbf{x}_{t-1}$ are truncated to zeros and the largest k entries are set to 1. **TPower** can benefit from a good starting point. The solution from **Linear** is used as the initial sparse vector $\mathbf{x}_0$.

Low Rank Bilinear Approximation: The low rank bilinear approximation (**LowRank**) procedure has been applied to solve the k-Sparse-PCA Papailiopoulos *et al.* (2013) and the Densest-k-Subgraph Papailiopoulos *et al.* (2014). It approximates the solu-

tion to a BQP by applying a bilinear relaxation which is given as,

$$\text{BQP}_b = \max_{\mathbf{x},\mathbf{y}} \{\mathbf{x}^\top \mathbf{Q}\mathbf{y}\} \text{ s.t. } \mathbf{x} \in \{0,1\}^d,\ \mathbf{y} \in \{0,1\}^d, \tag{7.11}$$
$$||\mathbf{x}||_1 = ||\mathbf{y}||_1 = k$$

BQP_b provides a good (2ρ-approximation Papailiopoulos *et al.* (2014)) approximation to BQP and can be solved in polynomial time using a r-rank approximation of $\mathbf{Q}$. The authors in Papailiopoulos *et al.* (2014) have developed the Spannogram algorithm to estimate a candidate set (termed the Spannogram $\mathscr{S}$) of vector pairs $(\mathbf{x}, \mathbf{y})$ with k features. One of the vectors from the pair that maximizes BQP_b, is the bilinear approximate solution to BQP.

Related BQP Approximation Methods: For the sake of completeness, two other techniques will be mentioned, that have been applied to approximate the BQP in the domain of feature selection. The authors in Nguyen *et al.* (2014) propose two relaxation techniques, (1) Spectral relaxation and (2) Semidefinite Programming relaxation. In Spectral relaxation (**Spectral**), the constraint on the values of $\mathbf{x}$ are relaxed to continuous values. The values of $\mathbf{x}$ being positive, can be interpreted as feature weights. The solution to **Spectral** has been shown to be the largest eigenvector of $\mathbf{Q}$ Nguyen *et al.* (2014). Semidefinite Programming relaxation (**SDP**) is a much more effective approximation to the BQP than **Spectral**. The BQP is approximated by a trace maximization problem using semidefinite relaxation. The approximate solution $\mathbf{x}$, to the BQP, is obtained from the **SDP** solution through random projection rounding based on Cholesky decomposition Goemans and Williamson (1995). In the experiments conducted, random rounding with 100 projections was implemented. For further details, please refer to Nguyen *et al.* (2014).

7.1.3 Other Mutual Information Based Methods

This subsection outlines other mutual information based feature selection methods that are compared with the techniques discussed above.

Greedy Selectors: These procedures usually begin with an empty set $\mathbb{S}$, and iteratively add to it the most important feature indices, until a fixed number of feature indices are selected or a stopping criterion is reached. Mutual information between the random variables (features and label) provides the ranking for the features. The most basic form of the scoring function is *Maximum Relevance* (**MaxRel**) Lewis (1992), where the score is simply the mutual information between the feature and the class variable. To account for the redundancy $I(X_i; X_j)$ between features, Peng *et al.* (2005), introduced the *Maximum Relevance Minimum Redundancy* (**MRMR**) criterion, which selects features with maximum relevance to the label and minimum redundancy between each other. A greedy procedure very closely related to the proposed technique is the *Joint Mutual Information* (**JMI**), that was developed by Yang and Moody (1999), and later by Meyer *et al.* (2008).

Global Selectors: There is limited work on mutual information based global feature selection. In Rodriguez-Lujan *et al.* (2010), the *Quadratic Programming Feature Selection* (**QPFS**) was introduced. This method can be viewed as a global alternative to **MRMR**. The second global technique proposed by Nguyen *et al.* (2014), is related to the proposed method presented in Equation (7.7). Nguyen *et al.* (2014), model a global feature selection problem based on the CMI matrix **Q**, and apply **Spectral** and **SDP** methods to approximate the solution.

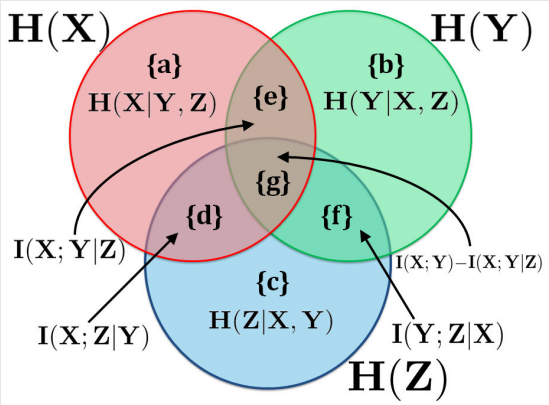


Figure 7.1: Venn diagram depicting entropy interaction H(X) = {a,e,g,d}, H(Y)={b,e,g,f}, H(Z)={c,d,g,f}, I(X;Y)={e,g}, I(X;Z) = {d,g}, I(Y;Z) = {g,f}, H(X,Y,Z) = {a,b,c,d,e,f,g}. Image based on Venkateswara *et al.* (2015a).

Information Theory Basics: Figure (7.1) provides a pictorial representation of these concepts through Venn diagrams. Entropy $H(X)$, is the expected information content for a random variable X. $H(X) = \mathbb{E}[I(X)]$, where $I(X) = -\log p(X)$. Therefore, $H(X) = -\sum_{\boldsymbol{x}} p(X = \boldsymbol{x}) \log p(X = \boldsymbol{x})$. Entropy $H(X)$, characterizes the uncertainty about the random variable X. Mutual Information (MI) between two random variables X and Y, is a measure of information shared between them and is represented as $I(X;Y)$. It is symmetric with $I(X;Y) = I(Y;X)$. In terms of entropy, mutual information is defined as $I(X;Y) = H(X) - H(X|Y)$, where $H(X|Y)$ is the conditional entropy. Mutual information between random variables X and Y can also be understood as the reduction in entropy of X (or Y) due to the presence of Y (or X). Conditional Mutual Information (CMI) denoted as $I(X;Y|Z)$, is the expected mutual information of two random variables X and Y, given a third random variable Z. Areas in entropy based Venn diagrams do not always correspond to positive quantities. In case of two variable overlap it is true as in $\{e\}$, $\{d\}$ or $\{f\}$. But, $\{g\}$ need not be positive. $I(X;Y) - I(X;Y|Z)$ can be less than 0.

7.2 Experiments

This section outlines some of the experiments conducted to test the feature selection model. The experiments were conducted using MATLAB on an Intel Core-i7 2.3 GHz processor with 16GB of memory.

7.2.1 Feature Selectors: A Test of Scalability

Greedy feature selectors have time complexities of the order $O(dk)$, which is negligible compared to the time complexities of the global feature selectors. Table (7.1) lists the time complexities for the global algorithms. To study time complexities, multiple experiments (d, k), were conducted, where a CMI matrix $\mathbf{Q}$ was simulated by a random positive symmetric matrix of size $[d \times d]$, and k features were selected. The time complexity for experiment (d, k), is the average time of convergence over 10 runs. The same set of random matrices were used for each of the algorithms in the experiment. Figure (7.2) depicts the convergence times for different experiments. **Linear** algorithm is the most efficient, followed closely by **Spectral** and **TPower** methods. CVX Grant and Boyd (2014) implementation with SDPT3 solver Toh *et al.* (1999), was used for all the **SDP** experiments. The **SDP** solver has a huge memory footprint and with matrix sizes $d \geq 700$, the computer ran into 'Out of Memory' errors. For the **LowRank** method, the following parameters were used, $r = 3, \epsilon = 0.1, \delta = 0.1$ for all the experiments.

Table 7.1: Time complexities for the Global approximate solutions for BQP in number of features d

Linear	**Spectral**	**SDP**	**LowRank***	**TPower****
$O(dk)$	$O(d^2)$	$O(d^{4.5})$	$O(d^{(r+1)})$	$O(td^2)$

$^*r$ is approximation rank, $^{**}t$ is number of iterations

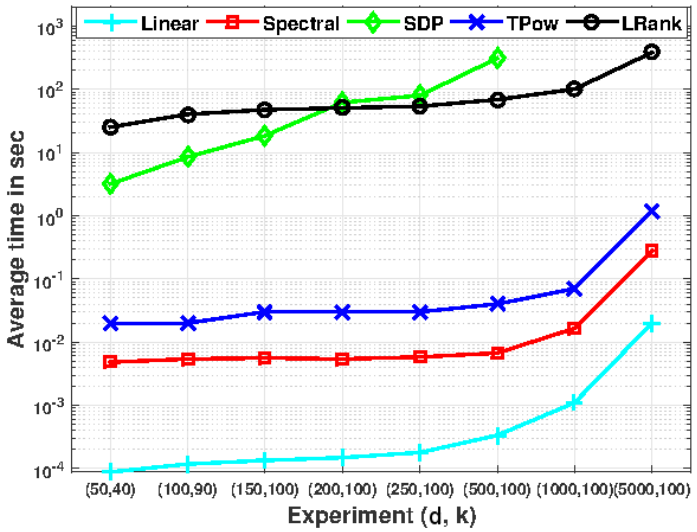


Figure 7.2: Average time in seconds for an algorithm to select k features from data containing d features in experiment (d, k). Image based on Venkateswara *et al.* (2015a).

7.2.2 BQP Methods: A Test of Approximation

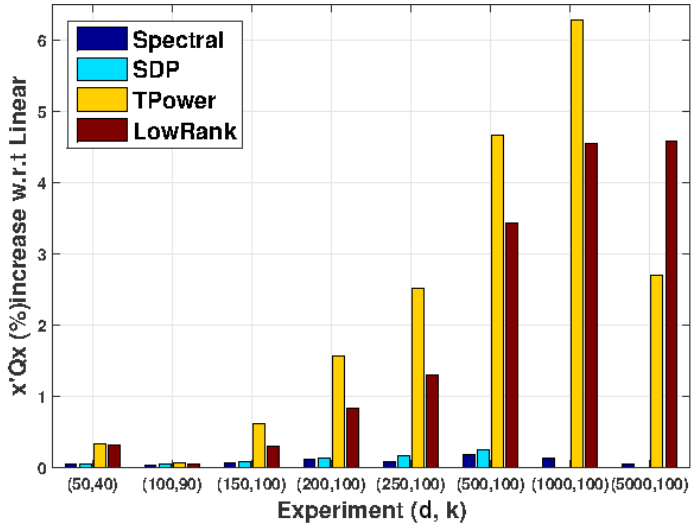


Figure 7.3: The average percentage difference of the BQP objective values compared with the **Linear** BQP objective value. In experiment (d, k), d is the matrix dimension and k is the number of features selected. Image based on Venkateswara *et al.* (2015a).

For the next set of experiments, the degree of approximation of the global algorithms was examined. Since the optimal solution is unknown for the BQP, the methods on evaluated by their relative objective values. The binary feature vector $\mathbf{x}$

was estimated after applying each of the methods and then the objective value $\mathbf{x}^\top \mathbf{Q}\mathbf{x}$ was evaluated. The percentage difference of every algorithm's objective value with the **Linear** method's objective was compared. Similar to the experimental evaluation used for time complexity, random data was generated for each experiment (d, k) and the values were averaged over 10 random runs. Figure (7.3) presents the results of the experiment. **TPower** and **LowRank** displayed the largest percentage increase from the **Linear**. Since **TPower** and **LowRank** approximate the BQP better than other methods, they must therefore be better feature selectors compared to **Linear**, **Spectral** and **SDP**. The **TPower** is also very efficient in terms of execution time and would therefore be the ideal feature selector when considering both speed and accuracy.

7.2.3 Feature Selectors: A Test of Classification Error

In these experiments, the **TPower** and the **LowRank** are compared with other algorithms in terms of classification accuracies. 13 publicly available datasets are chosen that are widely used to study mutual information based feature selection as in Rodriguez-Lujan *et al.* (2010); Brown *et al.* (2012); Nguyen *et al.* (2014); Peng *et al.* (2005). The details of the datasets are captured in Table (7.2). The feature selection was performed for a set of k values and classification performance was tested across all values of k. Starting at $k = 10$, the feature selection was incremented in steps 1 till d or 100, whichever was smaller. The classifier performance was evaluated using Leave-One-Out cross validation (if $n \leq 100$) or 10-fold cross validation and the cross validation errors(%) for each fold were obtained. Since the average error across all values of k is not a good measure of the classifier performance, the paired t-test was applied, as also mentioned in Rodriguez-Lujan *et al.* (2010); Nguyen *et al.* (2014); Herman *et al.* (2013), across the cross validation folds. For a fixed dataset

and a fixed value of k, to compare **TPower** with say, **MaxRel**, the one sided paired t-test was applied at 5% significance over the error(%) of the cross validation folds for the two algorithms. The performance of **TPower** vs. **MaxRel** set to win = w, tie = t and loss = l, based on the largest number of t-test decisions across all the k values. Along the lines of earlier studies in feature selection, a linear SVM classifier was used. To estimate the conditional mutual information, the features were discretized. The role of discretization is not unduly critical as long as it is consistent across all the experiments. The features were discretized using the Class Attribute Interdependence Maximization (CAIM) algorithm developed by Kurgan and Cios (2004). Feature selection was performed on discretized data but the classification (after feature selection) was performed on the original feature space.

Table 7.2: Datasets details: d is number of features, n is number of samples, c is number of categories, **Error**: is average cross validation error (%) using all features.

Data	d	n	c	**Error**	**Ref.**
Arrhythmia	258	420	2	31.1	Lichman (2013)
Colon	2000	62	2	37.0	Ding and Peng (2005)
Gisette	4995	6000	2	2.5	Lichman (2013)
Leukemia	7070	72	2	26.4	Ding and Peng (2005)
Lung	325	73	7	9.6	Ding and Peng (2005)
Lymphoma	4026	96	9	81.3	Ding and Peng (2005)
Madelon	500	2000	2	45.5	Lichman (2013)
Multi-Feat	649	2000	10	1.5	Lichman (2013)
Musk2	166	6598	2	4.6	Lichman (2013)
OptDigits	62	3823	10	3.3	Lichman (2013)
Promoter	57	106	2	26.0	Lichman (2013)
Spambase	57	4601	2	7.5	Lichman (2013)
Waveform	21	5000	3	13.1	Lichman (2013)

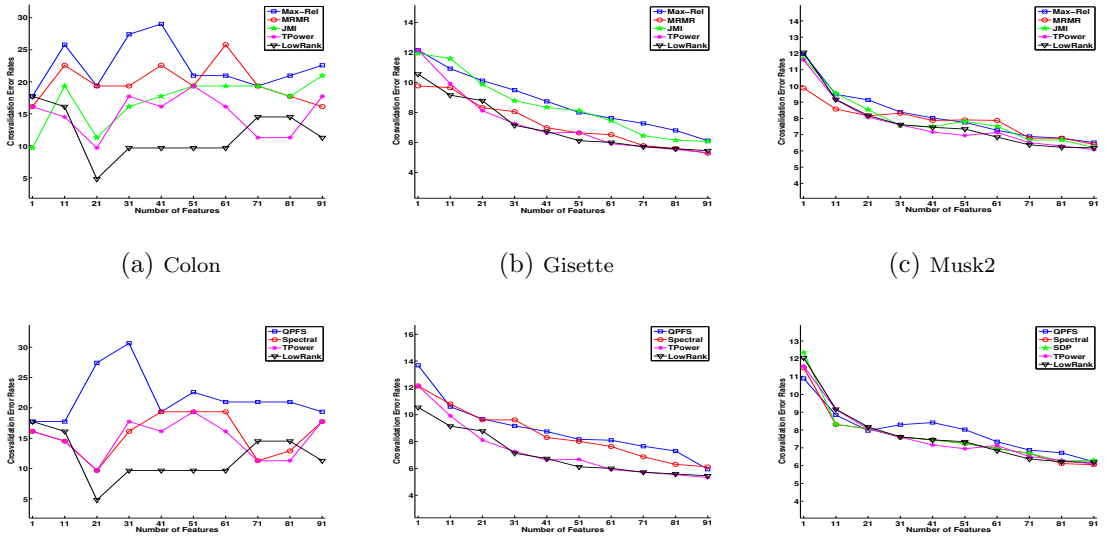
Using the above procedure, the performance of **TPower** and **LowRank** were compared with all the other algorithms. Tables (7.3) and (7.4) display the results of the experiment. The values in Table (7.3) (likewise Table (7.4)) correspond to the difference in the average of classification error(%) between **TPower** (likewise **LowRank**) and all the other algorithms. From the results in these tables, it can be gathered that **TPower** and **LowRank** outperform most of the datasets across all the algorithms. When compared against each other **LowRank** outperforms **TPower**. For the sake of brevity, the comparison between other pairs of algorithms has not displayed The win/tie/loss numbers by themselves do not provide a complete picture of the comparison. The difference in the average error also needs to be taken into account to assess the performance. A large percentage of negative values in the columns and their magnitudes indicate the low error values in classification for **TPower** and **LowRank**. Figure (7.4) displays the average classification error(%) trends for varying values of k for 3 datasets. Figures (7.4a) and (7.4d) for the Colon dataset, suggest that the addition of more features does not necessarily reduce classification error. Classification error trends also guide in validating the best value of k for a dataset. For a given dataset, the error trends between the global and greedy procedures follow a similar pattern. This perhaps indicates that nearly similar features are being selected using both types of methods. It should be noted that for huge datasets with large values of d, greedy methods may not be a bad choice for feature selection.

Table 7.3: Comparison of **TPower** with other algorithms. The table values measure the difference in average classification accuracies of **TPower** with other algorithms. w, t and l indicate one-sided paired t-test results. The last row displays the total number of Wins(W), Ties(T) and Loss(L). N/A indicates comparison data was unavailable for large datasets using **SDP**.

Data	MaxRel	MRMR	JMI	QPFS	Spectral	SDP	LowRank
Arrythmia	-0.37 ± 1.4 ^{t}	0.32 ± 1.0 ^{l}	0.02 ± 1.0 ^{l}	0.20 ± 1.8 ^{l}	-0.08 ± 1.1 ^{t}	-0.18 ± 1.0 ^{w}	-0.05 ± 0.8 ^{l}
Colon	-7.28 ± 4.6 ^{w}	-4.42 ± 4.2 ^{w}	-2.47 ± 3.8 ^{w}	-6.70 ± 4.6 ^{w}	-0.60 ± 2.8 ^{w}	N/A	4.03 ± 4.5 ^{l}
Gisette	-1.32 ± 0.6 ^{w}	0.00 ± 0.7 ^{w}	-1.12 ± 0.6 ^{w}	-1.38 ± 0.7 ^{w}	-1.26 ± 0.6 ^{w}	N/A	0.33 ± 0.6 ^{l}
Leukemia	0.11 ± 1.4 ^{w}	1.40 ± 1.6 ^{l}	1.59 ± 1.8 ^{l}	0.41 ± 1.1 ^{t}	-0.03 ± 0.6 ^{w}	N/A	1.49 ± 1.3 ^{l}
Lung	-9.43 ± 4.1 ^{w}	-2.52 ± 4.2 ^{w}	-3.83 ± 4.2 ^{w}	0.60 ± 2.8 ^{l}	-0.88 ± 2.2 ^{w}	-0.88 ± 2.1 ^{w}	-1.59 ± 2.4 ^{w}
Lymphoma	-2.76 ± 4.8 ^{w}	3.35 ± 4.7 ^{l}	2.93 ± 5.3 ^{l}	4.99 ± 3.3 ^{l}	-1.86 ± 2.5 ^{w}	N/A	3.29 ± 4.2 ^{l}
Madelon	0.32 ± 0.5 ^{l}	0.80 ± 0.9 ^{l}	0.01 ± 0.4 ^{w}	-0.22 ± 0.7 ^{w}	-0.01 ± 0.6 ^{w}	0.15 ± 0.6 ^{l}	-0.11 ± 0.4 ^{t}
MultiFeatures	0.02 ± 0.3 ^{w}	0.24 ± 0.3 ^{l}	0.17 ± 0.3 ^{l}	-0.42 ± 0.3 ^{w}	0.10 ± 0.3 ^{w}	0.11 ± 0.3 ^{w}	0.01 ± 0.3 ^{l}
Musk2	-0.45 ± 0.6 ^{w}	-0.22 ± 0.7 ^{w}	-0.18 ± 0.5 ^{w}	-0.31 ± 0.6 ^{w}	0.06 ± 0.4 ^{w}	0.03 ± 0.5 ^{w}	0.05 ± 0.4 ^{w}
OptDigits	-0.19 ± 0.5 ^{w}	-0.01 ± 0.6 ^{t}	0.16 ± 0.6 ^{l}	-0.65 ± 1.0 ^{w}	0.03 ± 0.3 ^{l}	-2.53 ± 13.0 ^{w}	0.08 ± 0.4 ^{l}
Promoter	0.73 ± 3.0 ^{l}	-0.04 ± 3.2 ^{w}	0.19 ± 3.0 ^{l}	-1.29 ± 3.8 ^{w}	-0.48 ± 2.8 ^{w}	-0.56 ± 2.9 ^{w}	-0.17 ± 3.1 ^{w}
Spambase	-0.34 ± 0.3 ^{w}	0.06 ± 0.2 ^{l}	-0.23 ± 0.3 ^{w}	0.03 ± 0.4 ^{l}	-0.09 ± 0.3 ^{w}	-0.10 ± 0.3 ^{w}	0.02 ± 0.1 ^{l}
Waveform	-0.13 ± 0.3 ^{w}	0.06 ± 0.3 ^{l}	-0.01 ± 0.0 ^{t}	0.04 ± 0.2 ^{t}	0.04 ± 0.1 ^{t}	0.00 ± 0.2 ^{t}	-0.01 ± 0.2 ^{t}
$\#W/T/L$:	10/1/2	5/1/7	6/1/6	7/2/4	10/2/1	7/1/1	3/2/8

Table 7.4: Comparison of **LowRank** with other algorithms. Table structure similar to Table 7.3

Data	MaxRel	MRMR	JMI	QPFS	Spectral	SDP	TPower
Arrythmia	-0.32 ± 1.3 ^{l}	0.36 ± 1.0 ^{l}	0.07 ± 1.0 ^{l}	0.25 ± 1.7 ^{l}	-0.03 ± 1.1 ^{t}	-0.13 ± 1.0 ^{w}	0.05 ± 0.8 ^{w}
Colon	-11.31 ± 4.7 ^{w}	-8.45 ± 4.3 ^{w}	-6.50 ± 3.9 ^{w}	-10.73 ± 5.3 ^{w}	-4.63 ± 5.0 ^{w}	N/A	-4.03 ± 4.5 ^{w}
Gisette	-1.65 ± 0.5 ^{w}	-0.32 ± 0.5 ^{w}	-1.44 ± 0.7 ^{w}	-1.70 ± 0.6 ^{w}	-1.58 ± 0.6 ^{w}	N/A	-0.33 ± 0.6 ^{w}
Leukemia	-1.39 ± 1.4 ^{w}	-0.09 ± 1.5 ^{t}	0.09 ± 1.7 ^{t}	-1.09 ± 1.3 ^{w}	-1.52 ± 1.2 ^{w}	N/A	-1.49 ± 1.3 ^{w}
Lung	-7.83 ± 4.1 ^{w}	-0.92 ± 4.5 ^{w}	-2.23 ± 4.5 ^{w}	2.19 ± 3.7 ^{l}	0.71 ± 2.5 ^{l}	0.71 ± 2.3 ^{l}	1.59 ± 2.4 ^{l}
Lymphoma	-6.06 ± 3.5 ^{w}	0.06 ± 2.1 ^{l}	-0.36 ± 2.1 ^{l}	1.70 ± 2.5 ^{l}	-5.15 ± 3.8 ^{w}	N/A	-3.29 ± 4.2 ^{w}
Madelon	0.43 ± 0.5 ^{l}	0.91 ± 0.8 ^{l}	0.12 ± 0.5 ^{l}	-0.11 ± 0.7 ^{w}	0.10 ± 0.6 ^{l}	0.26 ± 0.5 ^{l}	0.11 ± 0.4 ^{t}
MultiFeatures	0.01 ± 0.4 ^{l}	0.23 ± 0.3 ^{l}	0.16 ± 0.4 ^{l}	-0.43 ± 0.3 ^{w}	0.10 ± 0.4 ^{l}	0.10 ± 0.4 ^{l}	-0.01 ± 0.3 ^{w}
Musk2	-0.50 ± 0.4 ^{w}	-0.27 ± 0.7 ^{w}	-0.23 ± 0.5 ^{w}	-0.36 ± 0.5 ^{w}	0.02 ± 0.3 ^{l}	-0.02 ± 0.4 ^{l}	-0.05 ± 0.4 ^{l}
OptDigits	-0.26 ± 0.6 ^{w}	-0.09 ± 0.4 ^{w}	0.08 ± 0.4 ^{l}	-0.72 ± 1.2 ^{w}	-0.04 ± 0.4 ^{t}	-2.61 ± 13.1 ^{w}	-0.08 ± 0.4 ^{w}
Promoter	0.90 ± 3.1 ^{l}	0.13 ± 2.7 ^{w}	0.35 ± 2.9 ^{w}	-1.13 ± 3.7 ^{w}	-0.31 ± 2.8 ^{t}	-0.40 ± 2.5 ^{t}	0.17 ± 3.1 ^{l}
Spambase	-0.36 ± 0.3 ^{w}	0.04 ± 0.2 ^{l}	-0.24 ± 0.3 ^{w}	0.02 ± 0.4 ^{t}	-0.11 ± 0.3 ^{w}	-0.12 ± 0.3 ^{w}	-0.02 ± 0.1 ^{w}
Waveform	-0.12 ± 0.4 ^{w}	0.07 ± 0.1 ^{l}	0.00 ± 0.2 ^{t}	0.05 ± 0.1 ^{t}	0.05 ± 0.1 ^{t}	0.02 ± 0.1 ^{t}	0.01 ± 0.2 ^{t}
$\#W/T/L$:	9/0/4	6/1/6	6/2/5	8/2/3	5/4/4	3/2/4	8/2/3



(a) Colon (b) Gisette (c) Musk2

(d) Colon (e) Gisette (f) Musk2

Figure 7.4: Average cross validation error(%) vs. Number of features. First Row: Comparison of Greedy methods with **TPower** and **LowRank** for 3 datasets. Second Row: Comparison of Global methods across 3 datasets. Images based on Venkateswara *et al.* (2015a).

7.3 Nonlinear Feature Selection for Domain Adaptation

The following subsections outline a domain adaptation model that reduces cross domain disparity by selecting samples (*instance selection*) and also by selecting features (*feature selection*). The hypothesis is that performing both instance and feature selection can align the source and target domains. As in standard unsupervised domain adaptation, two domains are considered; *source* and *target*. The source consists of labeled data, $\mathcal{D}_s = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{n_s}$ and the target has only unlabeled data $\mathcal{D}_t = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$. The data points $\boldsymbol{x}_i^*$ belong to $\mathbb{R}^d$, where d is the dimension of each feature. The corresponding labels are represented by $y_i^* \in Y \coloneqq \{1, \ldots, c\}$. The main idea behind feature selection based domain adaptation is to select $k < d$ features to reduce domain disparity between $\mathcal{D}_s$ and $\mathcal{D}_t$.

7.3.1 Instance Selection

The Maximum Mean Discrepancy (MMD) measure, proposed by Borgwardt et al., is used to determine if two distributions are similar (Borgwardt *et al.* (2006)). The MMD is a non-parametric criterion that measures the distances between distributions in terms of distances between their means in a Reproducing Kernel Hilbert Space (RKHS). It has been discussed in detail in previous chapters. Researchers have adapted the MMD measure to perform *instance selection* for domain adaptation Long *et al.* (2014). Here, data points are sampled from the source domain such that their distribution is similar to the target distribution. A classifier trained on these sampled source data points can then be used for target data prediction. The MMD

for instance selection of source data points is formulated below,

$$\begin{aligned}
&\min_{\boldsymbol{\beta}} \left\| \frac{1}{n_s} \sum_{i}^{n_s} \beta_i \phi(\boldsymbol{x}_i^s) - \frac{1}{n_t} \sum_{j}^{n_t} \phi(\boldsymbol{x}_j^t) \right\|_{\mathcal{H}}^2 \\
&= \min_{\boldsymbol{\beta}} \frac{1}{n_s^2} \boldsymbol{\beta}^\top \hat{K}^s \boldsymbol{\beta} - \frac{2}{n_s n_t} \boldsymbol{\beta}^\top \hat{K}^{st} \mathbf{1} + \frac{1}{n_t^2} \mathbf{1}^\top \hat{K}^t \mathbf{1} \\
&\text{s.t.} \quad \boldsymbol{\beta} \in [0, B]^{n_s} \text{ and } \left| \frac{1}{n_s} \sum_{i}^{n_s} \beta_i - 1 \right| \leq \epsilon
\end{aligned} \tag{7.12}$$

In Equation (7.12), the weights $\boldsymbol{\beta}$ indicate the importance of the source data points. The kernel $\hat{K}^s$ is the $n_s \times n_s$ source data kernel matrix, $\hat{K}^t$ is the $n_t \times n_t$ target kernel matrix and $\hat{K}^{st}$ is the $n_s \times n_t$ source target kernel matrix. The kernel entries correspond to a nonlinear mapping where $k(\boldsymbol{x}_i, \boldsymbol{x}_j) =< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >$. B defines the scope bounding discrepancy between the source and target distributions. $B \to 1$ gives an unweighted solution. The second constraint ensures that $\beta_i p(\boldsymbol{x}_i^s)$ is still a probability distribution. The indices of $\boldsymbol{\beta}$ with the largest values are the indices of the sampled data points from the source. A nonlinear classifier trained with these source instances can be used to classify the target data.

7.3.2 Nonlinear Feature Selection

In Equation (7.12), instance sampling is performed by transforming the data points into a high dimensional (possibly infinite dimensional) space. In addition, feature selection must also be performed in a high dimensional space. Mutual information based feature selection discussed in the previous sections is infeasible when confronted with high dimensional (infinite dimensional) spaces. To perform feature selection in infinite dimensional spaces, the data can be visualized in terms of individual features. The source data can also be represented as $\mathbf{X}^s = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d]^\top$, where $\mathbf{u}_p \in \mathbb{R}^{n_s}$ for $p \in \{1, \ldots, d\}$. In order to do feature selection, the source kernel K^s is redefined as the sum over individual feature kernels, $K^s = K_1^s + K_2^s \ldots, K_d^s$,

where, $K_p^s \in \mathbb{R}^{n_s \times n_s}$ is the kernel using just the p^{th} feature of $\mathbf{X}^s$. Feature selection is implemented as selection over feature kernels as outlined in Venkateswara *et al.* (2013). The elements $K_{i,j}^s = k(\mathbf{u}_{pi}, \mathbf{u}_{pj})$, where $\mathbf{u}_{pl}$ is the l^{th} element of $\mathbf{u}_p$. Similarly, $K^{st} = K_1^{st} + K_2^{st} \ldots, K_d^{st}$ and $K^t = K_1^t + K_2^t \ldots, K_d^t$. Nonlinear feature selection is implemented by selecting the most important features by assigning weights to the respective feature kernels K_p^s, K_p^{st} and K_p^t. The optimization problem is outlined below,

$$\begin{aligned} \min_{\boldsymbol{\beta},\boldsymbol{\gamma}} \ & \frac{1}{n_s^2}\boldsymbol{\beta}^\top \Big(\sum_{p=1}^{d} \gamma_p K_p^s \Big) \boldsymbol{\beta} - \frac{2}{n_s n_t} \boldsymbol{\beta}^\top \Big(\sum_{p=1}^{d} \gamma_p K_p^{st} \Big) \mathbf{1} + \frac{1}{n_t^2} \mathbf{1}^\top \Big(\sum_{p=1}^{d} \gamma_p K_p^t \Big) \mathbf{1} \\ \text{s.t.} \quad & \boldsymbol{\beta} \in [0, B]^{n_s}, \quad \left| \frac{1}{n_s} \sum_{i}^{n_s} \beta_i - 1 \right| \leq \epsilon \\ & \boldsymbol{\gamma} \in [0, 1]^d, \quad \sum_{i}^{d} \gamma_i = k \end{aligned} \tag{7.13}$$

The above equation needs to be minimized in two variables $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. Equation (7.13) is an example of a bi-convex problem. A function $f : X \times Y \to \mathbb{R}$ is bi-convex, if $f(x, y)$ is convex in y for a fixed $x \in X$, and $f(x, y)$ is convex in x for a fixed $y \in Y$. Equation (7.13) is quadratic in $\boldsymbol{\beta}$ when $\boldsymbol{\gamma}$ is a constant and is therefore convex in $\boldsymbol{\beta}$. Similarly, Equation (7.13) is linear in $\boldsymbol{\gamma}$ when $\boldsymbol{\beta}$ is constant and is therefore convex in $\boldsymbol{\gamma}$. Therefore, Equation (7.13) is a bi-convex problem. Under these conditions, an alternating minimization approach is bound to converge to a critical point Gorski *et al.* (2007). Therefore, an alternating optimization procedure is applied to estimate $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. When $\boldsymbol{\gamma}$ is fixed, the solution for $\boldsymbol{\beta}$ is a standard MMD solution which is solved by quadratic programming. When $\boldsymbol{\beta}$ is fixed, the solution for $\boldsymbol{\gamma}$ can be expressed as a linear problem given by,

$$\min_{\substack{\boldsymbol{\alpha}:\ 0 \leq \alpha_p \leq 1, \\ \sum_p \alpha_p = k}} \sum_{p=1}^{d} \alpha_p v_p \tag{7.14}$$

where v_p is given by:

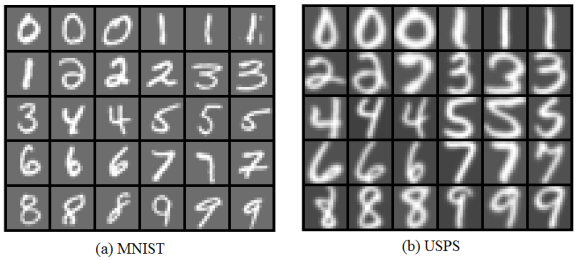
$$v_p = \boldsymbol{\beta}^\top K_p^s \boldsymbol{\beta} - \frac{2}{n_t} \boldsymbol{\beta}^\top K_p^{st} \mathbf{1} + \frac{1}{n_t^2} \mathbf{1}^\top K_p^t \mathbf{1}$$

The constraint $\sum_p \alpha_p = k$ ensures that at least k features are selected. The k features to be chosen are given by the indices of the k largest elements of $\mathbf{v} = [v_1, \ldots, v_d]$.

7.4 Experiments

For the experiments, digit datasets MNIST (LeCun *et al.* (1998)) and USPS (Roweis (2000)) datasets were used. Some examples from both the datasets are depicted in Figure (7.5). To the naked eye, there is a clear difference between the images in the two datasets. 500 samples were chosen from both the datasets. The digit image sizes were resized to 16×16 pixels resulting in images of 256 dimensions. Equation (7.13) was solved using alternate minimization.

Figure (7.6a) depicts the original datasets. The data points have been reduced to 2 dimensions using the unsupervised tSNE (Van der Maaten and Hinton (2008)). The two datasets cluster separately showing the distinct difference between the data points in the two domains. Equation (7.13) was solved with MNIST as source and USPS as target dataset and 100 features were selected out of 256 to reduce the domain disparity. Figure (7.6b) depicts the tSNE embeddings of two datasets using the reduced features. The size of the source data points indicates their importance



(a) MNIST (b) USPS

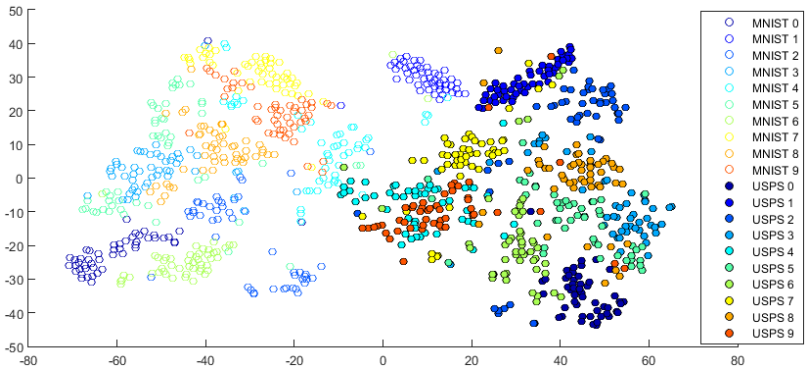
Figure 7.5: Example digits from the MNIST and the USPS datasets. There is a visible distinction between the digits from the two datasets.

($\boldsymbol{\beta}$) for domain adaptation. Similarly, Equation (7.13) was solved with USPS as the source dataset and MNIST as the target dataset and 100 features were selected out of 256 to reduce the domain disparity. The quadratic problem (estimating $\boldsymbol{\beta}$) was solved using MATLAB's `quadprog` function and the linear problem (estimating $\boldsymbol{\gamma}$) was solved using MATLAB's `linprog` function. In both the cases, the alternating optimization converges in a few iterations. Figure (7.6c) depicts the tSNE embeddings of two datasets using the reduced features. In both Figures (7.6b), (7.6c), the datasets seem to have much more overlap compared to Figure (7.6a). Table (7.5) compares the classification accuracies for unsupervised domain adaptation using the two datasets.

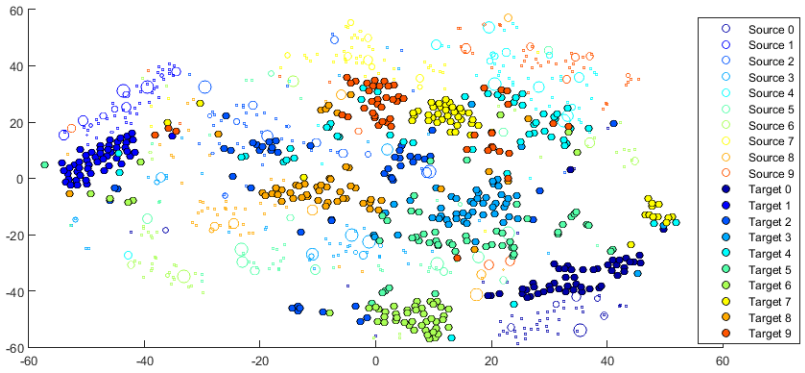
Table 7.5: Unsupervised domain adaptation experiments with digit data. In all cases, the classifier is trained with source data and tested on the target. MNIST→ USPS means, MNIST is source and USPS is target. **SVM** - regular linear SVM. **SVM**($\boldsymbol{\beta}$) - source data points are weighted by $\boldsymbol{\beta}$, **SVM**($\boldsymbol{\gamma}$) - source and target features are selected with $\boldsymbol{\gamma}$, **SVM**($\boldsymbol{\beta}, \boldsymbol{\gamma}$) - source data points are weighted by $\boldsymbol{\beta}$ and source and target features are selected with $\boldsymbol{\gamma}$.

Expt.	**SVM**	**SVM**($\boldsymbol{\beta}$)	**SVM**($\boldsymbol{\gamma}$)	**SVM**($\boldsymbol{\beta}, \boldsymbol{\gamma}$)
MNIST→ USPS	36.6%	36.2%	35.2%	41.8%
USPS→ MNIST	21.8%	26.6%	23.8%	24.0%

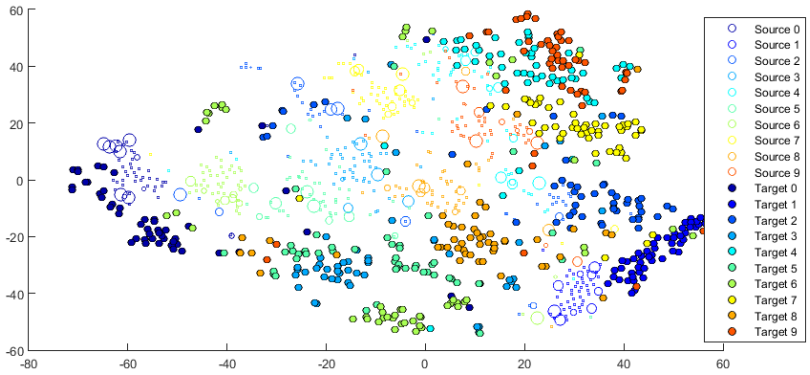
In the MNIST → USPS experiment, there is marked improvement using both the weighted source and feature selection. The results in Table (7.5) show that feature selection with instance sampling gives better accuracies than just doing feature selection or instance sampling in the case of MNIST → USPS. In the USPS → MNIST experiment, there is not much improvement compared to other methods and doing only instance sampling appears to perform better than feature selection and instance sampling. In these experiments, the model estimation ($\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$) and classification are carried out in two steps. A more rigorous implementation would be to include both model estimation and classification in the same step, thereby learning an efficient classifier for the source with weighted training data points and reduced dimensions.



(a)



(b)



(c)

Figure 7.6: The tSNE embeded images depicting both the datasets in two dimensions. (a) The two datasets are clearly different distributions. There appears to be little overlap between the clustered digits in the two datasets. (b) tSNE image after solving Equation (7.13) with MNIST as source and USPS as target and selecting 100 features. The importance of the source data points $\boldsymbol{\beta}$ is denoted by the size of the unfilled circles. There is more overlap between the source and target datasets when compared to (a). (c) tSNE image after solving Equation (7.13) with USPS as source and MNIST as target and selecting 100 features.

7.5 Conclusions and Summary

There is scope for improvement in the area of mutual information based feature selection. Feature selection is a NP-hard problem and newer methods to approximate the solution will help drive research in this area. Currently, information gain based feature selection is based on conditional independence assumptions that is similar to Naïve Bayesian models. While mutual information and conditional mutual information are the existing criteria for determining the importance of 2 or 3 features at a time, there is need to derive measures to better approximate the importance of a group of selected features. In the area of feature selection for domain adaptation, the proposed model is a promising start. Although the preliminary results on feature selection are encouraging, more experiments need to be conducted to firmly establish the hypothesis that feature selection can reduce domain disparity.

In summary, there were two parts to this chapter. In the first part, two global procedures for feature selection were introduced, namely the Truncated Power Method **TPower** and the Low Rank Bilinear Approximation **LowRank**. The experiments compared the proposed technique with existing feature selectors and showed that both **TPower** and **LowRank** perform better than existing global and iterative techniques across most of the datasets. While **LowRank** slightly outperforms **TPower**, it does not compare well with regards to time. The role of conditional mutual information in feature selection was also demonstrated by the theorems. In the second half of the chapter, a nonlinear feature selection and instance selection model for domain adaptation was introduced. Preliminary experiments were conducted with digit datasets *MNIST* and *USPS*. These results indicate that feature selection can be used to increase domain overlap.

Chapter 8

DOMAIN ADAPTATION - FUTURE DIRECTIONS

This chapter proposes some directions for future research in domain adaptation.

8.1 Understanding Domain Shift

The concept of a domain has been defined vaguely in computer vision. Images from different datasets are viewed as belonging to different domains. It is true that datasets have an inherent bias and images from a dataset have certain properties that can help identify the dataset (Torralba and Efros (2011)). However, there has been limited effort in understanding what creates this bias and on modeling the domain shift between datasets. The authors in Tommasi *et al.* (2016) attempt to identify the *domainness* - a measure for domain specificity of an image.

The difficult problem of modeling domain shift in computer vision has been rarely addressed. There has been work on identifying domains from a mixture of multiple datasets and then studying transfer of knowledge between the domains (Gong *et al.* (2013b)). Although this does not necessarily model a domain, it provides some direction towards identifying a domain through analysis. The difficulties of modeling domain shift in computer vision mostly arise due to variations in representation and not merely variations in the data being represented. The very process of imaging (camera perspective, occlusion), storage (resolution, size) and representation (color, brightness, contrast) can lead to variations. Image background (context) is another cause for variation. Finally, the diversity in the 'real data' itself could also lead variations in their images.

Most domain adaptation systems create adaptive models that perform generic

domain adaptation. The models are often guided by the datasets that are used. On the other hand, it might be beneficial to tailor the adaptation model to a specific variation in the data. This would however need a comprehensive understanding of domain shift. It might also lead to task-specific domain adaptation models based on the nature of domain-shift, leading to increased adoption of domain adaptation in real world applications.

8.2 Datasets

The standard datasets for computer vision based domain adaptation are, facial expression datasets *CKPlus* (Lucey *et al.* (2010)) and *MMI* (Pantic *et al.* (2005)), digit datasets *SVHN* (Netzer *et al.* (2011)), *USPS* and *MNIST* (Jarrett *et al.* (2009)), head pose recognition datasets *PIE* (Long *et al.* (2013)), object recognition datasets *COIL* (Long *et al.* (2013)), *Office* (Saenko *et al.* (2010)) and *Office-Caltech* (Gong *et al.* (2012a)).

The following reasons outline the need for new datasets.

1. The currently available datasets for domain adaptation are not suitable for deep learning. These datasets were created before deep learning became popular and are insufficient for training and evaluating deep learning based domain adaptation approaches. These datasets are small with a limited number of categories. For instance, the most popular object-recognition dataset *Office*, has 4110 images across 31 categories and the newly introduced *Office-Home* has $15,000$ images across 65 categories. A deep learning model with millions of parameters requires millions of images for training. Current approaches fine-tune pre-trained deep networks avoiding over-fitting issues. In order to train large scale adaptive systems, deep learning is the preferred model which in turn demands large datasets.

2. The datasets were developed for evaluating generic domain adaptation. As also discussed in the previous section, these datasets were created without modeling domain shift. The domain boundaries are determined by dataset identity and not by the nature of domain shift. When datasets are modeled on a specific domain shift it will lead to the generation of task-specific domain adaptation models leading to a rich family of domain adaptation models. This will in turn lead to widespread adoption of domain adaptation for real world problems.

Modeling of domain-shift and creation of new datasets complement each other.

8.3 Generative Models

Generative models, like Generative Adversarial Networks (GANs) are currently very popular in the research community. They have a wide range of applications from image super-resolution (Ledig *et al.* (2016)), text-to-image generation (Reed *et al.* (2016); Zhang *et al.* (2016)), image-to-image generation (Isola *et al.* (2016)) and conditional image generation (Nguyen *et al.* (2016); Chen *et al.* (2016)). The intuition behind the idea of generative models is based on a quote from physicist Richard Feynman [1],

'What I cannot create, I do not understand.'

It is common knowledge that when humans see an object for the first time, they form a mental image of the object and call upon that generated image when referencing the object at a later point in time. They are also able to understand the object based on their previous learning of similar looking objects. For e.g. they can guess the texture and weight of an object or 'imagine' how the object would appear when viewed from a different perspective. All of this is perhaps due to the generative models and transfer

[1] `https://openai.com/blog/generative-models/`

learning models in the brain that have evolved and perfected over millions of years. Based on this common understanding of the human brain and encouraged by the current success of generative adversarial models for domain adaptation, it would not be farfetched to hypothesize that transfer learning and generative models are closely related. In order to advance transfer learning and domain adaptation to the next level, generative models appear to hold the most promise.

8.4 Aligning Joint Distributions

Current forms of adaptation merely align the marginal distributions of the source and the target $P_S(X)$ and $P_T(X)$. The popular Maximum Mean Discrepancy (MMD) measure from Borgwardt *et al.* (2006), is often applied to align the marginal distributions of the source and target data, as described in this *instance selection* approach Gong *et al.* (2013a). The goal of domain adaptation is not merely aligning the domains but also being able to use the models trained on the source upon the target. In most cases, the domain adaptive models are created for classification. It would therefore make more sense to align the joint distributions $P_S(X, Y)$ with $P_T(X, Y)$ rather than merely the marginal distributions. The alignment of joint distributions will make a classifier trained on the source, an effective classifier for the target.

The challenge with this approach is that target labels are not available in unsupervised domain adaptation. The workaround is to impute the target data labels and refine them iteratively. There has been work in this regard as in Long *et al.* (2013), where the joint distributions are aligned in a spectral method using Kernel-PCA, by imputing the labels and refining them over multiple iterations. A deep learning approach has also been attempted in this regard in Long *et al.* (2016a), using a transductive approach to learn the target labels while also minimizing the joint domain discrepancy. Conditional generative models (discussed the Section 8.3) along with

joint distribution alignment could be the next wave of domain adaptation models.

8.5 Person-Centered Domain Adaptation

Very soon, computing is going to become all pervasive. The environment is plugged with computing devices and an average person carries quite a few smart-devices like a phone, watch, wristband, etc. Can this computing be adapted to every user ? Computing that adapts to a user's needs and idiosyncrasies can be called Person-Centered Computing (PCC) (Panchanathan *et al.* (2012)). This would mean that personal devices would model their interaction and response based on the user's needs, rather than a one-size-fits-all approach where users train themselves to adapt to their devices, in order to effectively interact with them. This paradigm, where the user and the device adapt to each other, is termed as *co-adaptation*.

These personalized devices will need to be designed to have core functional components making them applicable to a broad range of users. In addition, they must also have co-adaptive components that help customize the device at an individual user level. The device must adapt to the user based on patterns gathered from user interaction with the device. The learning models for co-adaptation will be based on unsupervised domain adaptation, which would involve gleaning patterns from unlabeled user interaction data. There has been no work so far in domain adaptation literature for person-centered device adaptation and these person-centered adaptive models would make technology more accessible, especially to individuals with disabilities.

Chapter 9

SUMMARY

The dissertation approaches the problem of domain adaptation from the concept of feature spaces. The literature survey was presented along these lines and it provides a new perspective on domain adaptation approaches. The dissertation presented three models for domain adaptation based on linear, nonlinear and hierarchical feature spaces.

The linear model was a max-margin solution with a pair of linear decision boundaries for the source and the target that was learned simultaneously. The nonlinear model was a kernel-PCA solution based on domain alignment using maximum mean discrepancy. The method also embedded data onto a manifold to ensure enhanced classification. A highlight of this work was a validation procedure that can be used to learn model parameters for unsupervised domain adaptation. The hierarchical method was a deep learning model based on estimating domain aligned hash values for the source and target data. This work introduced an unsupervised hash loss for the unlabeled target. The dissertation introduced the *Office-Home* dataset for domain adaptation consisting of 4 domains and around $15,000$ images. It also proposed a nonlinear method for domain adaptation based on feature selection.

The solutions for domain adaptation developed over the years can be viewed as dataset or feature specific. Although the proposed models have been tested on different datasets, the range of problems being addressed has been narrow. The gamut of applications that can tackled by domain adaptation, have not been fully explored. The lack of variety in datasets has restricted the range of domain adaptation problems that have been explored. The chapter on future directions provides some

insights into the promising areas of research in domain adaptation. The advent of deep learning and its unprecedented success in tackling domain adaptation, will eventually lead to the introduction of new datasets and generic problem statements in domain adaptation.

BIBLIOGRAPHY

Ajakan, H., P. Germain, H. Larochelle, F. Laviolette and M. Marchand, "Domain-adversarial neural networks", arXiv preprint arXiv:1412.4446 (2014).

Aytar, Y. and A. Zisserman, "Tabula rasa: Model transfer for object category detection", in "Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV)", pp. 2252–2259 (2011).

Bakker, B. and T. Heskes, "Task clustering and gating for bayesian multitask learning", J. Mach. Learn. Res. **4**, 83–99 (2003).

Beijbom, O., "Domain adaptations for computer vision applications", arXiv preprint arXiv:1211.4860 (2012).

Belkin, M. and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation", Neural computation **15**, 6, 1373–1396 (2003).

Ben-David, S., J. Blitzer, K. Crammer, A. Kulesza, F. Pereira and J. W. Vaughan, "A theory of learning from different domains", Machine learning **79**, 1-2, 151–175 (2010).

Bengio, Y., "Learning deep architectures for ai", Foundations and trends® in Machine Learning **2**, 1, 1–127 (2009).

Bengio, Y., A. Courville and P. Vincent, "Representation learning: A review and new perspectives", IEEE Trans. on Pattern Analysis and Machine Intelligence **35**, 8, 1798–1828 (2013).

Bengio, Y. *et al.*, "Deep learning of representations for unsupervised and transfer learning.", Workshops, Proceedings of the ACM Intl. Conf. on Machine Learning (ICML) **27**, 17–36 (2012).

Bickel, S., M. Brückner and T. Scheffer, "Discriminative learning under covariate shift", J. Mach. Learn. Res. **10**, 2137–2155 (2009).

Blitzer, J., K. Crammer, A. Kulesza, F. Pereira and J. Wortman, "Learning bounds for domain adaptation", in "Advances in Neural Information Processing Systems (NIPS)", pp. 129–136 (2008).

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

Bousmalis, K., N. Silberman, D. Dohan, D. Erhan and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks", in "accepted to the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2017).

Bousmalis, K., G. Trigeorgis, N. Silberman, D. Krishnan and D. Erhan, "Domain separation networks", in "Advances in Neural Information Processing Systems (NIPS)", pp. 343–351 (2016).

Brown, G., A. Pocock, M.-J. Zhao and M. Luján, "Conditional likelihood maximisation: a unifying framework for information theoretic feature selection", J. Mach. Learn. Res. **13**, 1, 27–66 (2012).

Bruzzone, L. and M. Marconcini, "Domain adaptation problems: A DASVM classification technique and a circular validation strategy", IEEE Trans. on Pattern Analysis and Machine Intelligence **32**, 5, 770–787 (2010).

Carreira-Perpinán, M. A. and R. Raziperchikolaei, "Hashing with binary autoencoders", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 557–566 (2015).

Caruana, R., "Multitask learning", Machine learning **28**, 1, 41–75 (1997).

Castrejon, L., Y. Aytar, C. Vondrick, H. Pirsiavash and A. Torralba, "Learning aligned cross-modal representations from weakly aligned data", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2016).

Chapelle, O., B. Scholkopf and A. Zien, eds., *Semi-Supervised learning* (MIT Press, Cambridge, 2006).

Chatfield, K., V. Lempitsky, A. Vedaldi and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods", in "Proceedings of the British Machine Vision Conference (BMVC)", (2011a).

Chatfield, K., V. S. Lempitsky, A. Vedaldi and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods.", in "British Machine Vision Conference (BMVC)", p. 8 (2011b).

Chatfield, K., K. Simonyan, A. Vedaldi and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets", in "British Machine Vision Conference (BMVC)", (2014).

Chattopadhyay, R., W. Fan, I. Davidson, S. Panchanathan and J. Ye, "Joint transfer and batch-mode active learning", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 253–261 (2013).

Chattopadhyay, R., Q. Sun, W. Fan, I. Davidson, S. Panchanathan and J. Ye, "Multisource domain adaptation and its application to early detection of fatigue", ACM Transactions on Knowledge Discovery from Data (TKDD) **6**, 4, 18 (2012).

Chen, X., Y. Duan, R. Houthooft, J. Schulman, I. Sutskever and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets", in "Advances in Neural Information Processing Systems (NIPS)", pp. 2172–2180 (2016).

Chopra, S., S. Balakrishnan and R. Gopalan, "DLID: Deep learning for domain adaptation by interpolating between domains", in "ICML Workshop on Challenges in Representation Learning", (2013).

Chu, W.-S., F. De la Torre and J. F. Cohn, "Selective transfer machine for personalized facial action unit detection", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 3515–3522 (2013).

Chung, F. R., *Spectral graph theory*, vol. 92 (American Mathematical Soc., 1997).

Collobert, R. and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 160–167 (2008).

Csurka, G., "Domain adaptation for visual applications: A comprehensive survey", arXiv preprint arXiv:1702.05374 (2017).

Daumé III, H., "Frustratingly easy domain adaptation", in "Conference of the Association for Computational Linguistics (ACL)", (2007).

Daumé III, H., A. Kumar and A. Saha, "Frustratingly easy semi-supervised domain adaptation", in "Workshop on Domain Adaptation for NLP", (2010), URL `http://hal3.name/docs/#daume10easyss`.

Davis, J. V., B. Kulis, P. Jain, S. Sra and I. S. Dhillon, "Information-theoretic metric learning", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 209–216 (2007).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2009).

Ding, C. and H. Peng, "Minimum redundancy feature selection from microarray gene expression data", Journal of bioinformatics and computational biology **3**, 02, 185–205 (2005).

Do, T.-T., A.-D. Doan and N.-M. Cheung, "Learning to hash with binary deep neural network", in "European Conference on Computer Vision", pp. 219–234 (Springer, 2016).

Donahue, J., Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition.", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", vol. 32, pp. 647–655 (2014).

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

Duan, L., I. W. Tsang, D. Xu and S. J. Maybank, "Domain transfer SVM for video concept detection", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 1375–1381 (2009).

Dudík, M., S. J. Phillips and R. E. Schapire, "Correcting sample selection bias in maximum entropy density estimation", in "Advances in Neural Information Processing Systems (NIPS)", pp. 323–330 (2005).

Editorial, "Anticipating artificial intelligence", Nature **532**, 7600, URL http://www.nature.com/news/anticipating-artificial-intelligence-1.19825 (2016).

Everingham, M., S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The pascal visual object classes challenge: A retrospective", International Journal of Computer Vision (IJCV) **111**, 1, 98–136 (2015).

Evgeniou, A. and M. Pontil, "Multi-task feature learning", Advances in Neural Information Processing Systems (NIPS) **19**, 41 (2007).

Evgeniou, T. and M. Pontil, "Regularized multi-task learning", in "Proceedings of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining", pp. 109–117 (2004).

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "LIBLINEAR: A library for large linear classification", J. Mach. Learn. Res. **9**, 1871–1874 (2008).

Fei, G., S. Wang and B. Liu, "Learning cumulatively to become more knowledgeable", in "Proceedings of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining", pp. 1565–1574 (ACM, 2016).

Fei-Fei, L., R. Fergus and P. Perona, "One-shot learning of object categories", IEEE Trans. on Pattern Analysis and Machine Intelligence **28**, 4, 594–611 (2006).

Fernando, B., A. Habrard, M. Sebban and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 2960–2967 (2013).

Ganin, Y., E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand and V. Lempitsky, "Domain-adversarial training of neural networks", J. Mach. Learn. Res. **17**, 59, 1–35 (2016).

Garey, M. R. and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York, NY, USA, 1990).

Ghifary, M., W. B. Kleijn, M. Zhang, D. Balduzzi and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation", in "Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 597–613 (2016).

Glorot, X., A. Bordes and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 513–520 (2011).

Goemans, M. X. and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming", Journal of the ACM (JACM) **42**, 6, 1115–1145 (1995).

Gong, B., K. Grauman and F. Sha, "Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation.", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 222–230 (2013a).

Gong, B., K. Grauman and F. Sha, "Reshaping visual datasets for domain adaptation", in "Advances in Neural Information Processing Systems (NIPS)", pp. 1286–1294 (2013b).

Gong, B., Y. Shi, F. Sha and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2012a).

Gong, P., J. Ye and C. Zhang, "Robust multi-task feature learning", in "Proceedings of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining", pp. 895–903 (2012b).

Gong, Y. and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 817–824 (2011).

Gong, Y., S. Lazebnik, A. Gordo and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval", IEEE Trans. on Pattern Analysis and Machine Intelligence **35**, 12, 2916–2929 (2013c).

Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning* (MIT Press, 2016), `http://www.deeplearningbook.org`.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets", in "Advances in Neural Information Processing Systems (NIPS)", pp. 2672–2680 (2014).

Gopalan, R., R. Li and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach", in "Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV)", pp. 999–1006 (IEEE, 2011).

Gopalan, R., R. Li and R. Chellappa, "Unsupervised adaptation across domain shifts by generating intermediate data representations", IEEE Trans. on Pattern Analysis and Machine Intelligence **36**, 11, 2288–2302 (2014).

Gorski, J., F. Pfeuffer and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions", Mathematical Methods of Operations Research **66**, 3, 373–407 (2007).

Grant, M. and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1", `http://cvxr.com/cvx` (2014).

Gretton, A., K. M. Borgwardt, M. Rasch, B. Schölkopf, A. J. Smola *et al.*, "A kernel method for the two-sample-problem", Advances in Neural Information Processing Systems (NIPS) **19**, 513 (2007).

Gretton, A., D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu and B. K. Sriperumbudur, "Optimal kernel choice for large-scale two-sample tests", in "Advances in Neural Information Processing Systems (NIPS)", pp. 1205–1213 (2012).

Gretton, A., A. Smola, J. Huang, M. Schmittfull, K. Borgwardt and B. Schölkopf, "Covariate shift by kernel mean matching", Dataset shift in machine learning **3**, 4, 5 (2009).

Griffin, G., A. Holub and P. Perona, "Caltech-256 object category dataset", Tech. Rep. 7694, California Institute of Technology, URL `http://authors.library.caltech.edu/7694` (2007).

He, K., F. Wen and J. Sun, "K-Means hashing: An affinity-preserving quantization method for learning binary compact codes", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 2938–2945 (2013).

He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 770–778 (2016).

Heckman, J. J., "Sample selection bias as a specification error", Econometrica: Journal of the econometric society pp. 153–161 (1979).

Helleputte, T. and P. Dupont, "Feature selection by transfer learning with linear regularized models", in "Machine Learning and Knowledge Discovery in Databases", pp. 533–547 (Springer, 2009).

Herman, G., B. Zhang, Y. Wang, G. Ye and F. Chen, "Mutual information-based method for selecting informative feature sets", Pattern Recognition **46**, 12, 3315–3327 (2013).

Hochreiter, S., Y. Bengio, P. Frasconi and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies", (2001).

Hoffman, J., B. Kulis, T. Darrell and K. Saenko, "Discovering latent domains for multisource domain adaptation", in "Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 702–715 (2012).

Hoffman, J., E. Rodner, J. Donahue, K. Saenko and T. Darrell, "Efficient learning of domain-invariant image representations", in "Intl. Conf. on Learning Representations (ICLR)", (2013).

Hu, J., J. Lu and Y.-P. Tan, "Deep transfer metric learning", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 325–333 (2015).

Huang, J., 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 (NIPS)", pp. 601–608 (2006).

IBM-ILOG-CPLEX, "High-performance mathematical programming engine", IBM Corp (2013).

ImageNet, "ImageNet large scale visual recognition challenge 2013 (ilsvrc2013)", http://www.image-net.org/challenges/LSVRC/2013/, accessed: 2017-02-12 (2013).

Isola, P., J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-image translation with conditional adversarial networks", arXiv preprint arXiv:1611.07004 (2016).

Jarrett, K., K. Kavukcuoglu, Y. Lecun *et al.*, "What is the best multi-stage architecture for object recognition?", in "Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV)", pp. 2146–2153 (2009).

Jhuo, I.-H., D. Liu, D. Lee and S.-F. Chang, "Robust visual domain adaptation with low-rank reconstruction", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 2168–2175 (2012).

Jiang, W., F. Nie, F.-l. K. Chung and H. Huang, "Algorithm and theoretical analysis for domain adaptation feature learning with linear classifiers", arXiv preprint arXiv:1509.01710 (2015).

Joachims, T., "Transductive inference for text classification using support vector machines", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", vol. 99, pp. 200–209 (1999).

Kamnitsas, K., C. Baumgartner, C. Ledig, V. F. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, A. Nori, A. Criminisi, D. Rueckert *et al.*, "Unsupervised domain adaptation in brain lesion segmentation with adversarial networks", in "International Conference on Information Processing in Medical Imaging (IPMI) 2017", (2016).

Kang, Z., K. Grauman and F. Sha, "Learning with whom to share in multi-task feature learning", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 521–528 (2011).

Kantorov, V. and I. Laptev, "Efficient feature extraction, encoding, and classification for action recognition", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2014).

Kifer, D., S. Ben-David and J. Gehrke, "Detecting change in data streams", in "Proceedings of the Thirtieth international conference on Very large data bases-Volume 30", pp. 180–191 (VLDB Endowment, 2004).

Kimeldorf, G. S. and G. Wahba, "A correspondence between Bayesian estimation on stochastic processes and smoothing by splines", The Annals of Mathematical Statistics **41**, 2, 495–502 (1970).

Kingma, D. P. and M. Welling, "Auto-encoding variational bayes", arXiv preprint arXiv:1312.6114 (2013).

Koniusz, P., Y. Tas and F. Porikli, "Domain adaptation by mixture of alignments of second-or higher-order scatter tensors", in "accepted to the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2017).

Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in "Advances in Neural Information Processing Systems (NIPS)", pp. 1097–1105 (2012).

Kuehne, H., H. Jhuang, E. Garrote, T. Poggio and T. Serre, "HMDB: a large video database for human motion recognition", in "Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV)", (2011).

Kulis, B., K. Saenko and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 1785–1792 (2011).

Kumar, A. and H. Daumé III, "Learning task grouping and overlap in multi-task learning", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", (2012), URL `http://hal3.name/docs/#daume12gomtl`.

Kurgan, L. A. and K. J. Cios, "Caim discretization algorithm", IEEE Transactions on Knowledge and Data Engineering **16**, 2, 145–153 (2004).

Larochelle, H., D. Erhan and Y. Bengio, "Zero-data learning of new tasks", in "Proceedings of the AAAI Conf. on Artificial Intelligence", vol. 1, p. 3 (2008).

LeCun, Y., C. Cortes and C. J. Burges, "The mnist database of handwritten digits", (1998).

Ledig, C., L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network", arXiv preprint arXiv:1609.04802 (2016).

Lee, H., R. Grosse, R. Ranganath and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 609–616 (2009).

Lewis, D. D., "Feature selection and feature extraction for text categorization", in "Proceedings of the workshop on Speech and Natural Language", pp. 212–217 (Association for Computational Linguistics, 1992).

Li, W., L. Duan, D. Xu and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation", IEEE Trans. on Pattern Analysis and Machine Intelligence **36**, 6, 1134–1148 (2014).

Li, W.-J., S. Wang and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels", in "Proceedings of the Intl. Joint Conf. on Artificial Intelligence", (2016).

Li, Z. and D. Hoiem, "Learning without forgetting", in "Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 614–629 (Springer, 2016).

Lichman, M., "UCI machine learning repository", URL `http://archive.ics.uci.edu/ml` (2013).

Liu, J., S. Ji and J. Ye, "Multi-task feature learning via efficient l2, 1-norm minimization", in "Proceedings of the Twenty-Fifth conference on Uncertainty in Artificial Intelligence", pp. 339–348 (AUAI Press, 2009).

Liu, M.-Y. and O. Tuzel, "Coupled generative adversarial networks", in "Advances in Neural Information Processing Systems (NIPS)", pp. 469–477 (2016).

Long, M., Y. Cao, J. Wang and M. Jordan, "Learning transferable features with deep adaptation networks", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 97–105 (2015).

Long, M., J. Wang, G. Ding, J. Sun and P. S. Yu, "Transfer feature learning with joint distribution adaptation", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 2200–2207 (2013).

Long, M., J. Wang, G. Ding, J. Sun and P. S. Yu, "Transfer joint matching for unsupervised domain adaptation", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2014).

Long, M., J. Wang and M. I. Jordan, "Deep transfer learning with joint adaptation networks", arXiv preprint arXiv:1605.06636 (2016a).

Long, M., H. Zhu, J. Wang and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks", in "Advances in Neural Information Processing Systems (NIPS)", (2016b).

Lucey, P., J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, "The extended cohn-kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression", in "Workshops, Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPRW)", (2010).

Mairal, J., F. Bach, J. Ponce and G. Sapiro, "Online learning for matrix factorization and sparse coding", J. Mach. Learn. Res. **11**, 19–60 (2010).

Mansour, Y., M. Mohri and A. Rostamizadeh, "Domain adaptation with multiple sources", in "Advances in Neural Information Processing Systems (NIPS)", pp. 1041–1048 (2009).

Mensink, T., J. Verbeek, F. Perronnin and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost", IEEE Trans. on Pattern Analysis and Machine Intelligence **35**, 11, 2624–2637 (2013).

Meyer, P. E., C. Schretter and G. Bontempi, "Information-theoretic feature selection in microarray data using variable complementarity", IEEE Journal of Selected Topics in Signal Processing **2**, 3, 261–274 (2008).

Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning", in "Workshops - Advances in Neural Information Processing Systems (NIPS)", (2011), URL `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`.

Ng, A., "Hiring your first chief AI officer", Harvard Business Review URL https://hbr.org/2016/11/hiring-your-first-chief-ai-officer (2016).

Nguyen, A., J. Yosinski, Y. Bengio, A. Dosovitskiy and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space", arXiv preprint arXiv:1612.00005 (2016).

Nguyen, X. V., J. Chan, S. Romano and J. Bailey, "Effective global approaches for mutual information based feature selection", in "Proceedings of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining", pp. 512–521 (2014).

Ni, J., Q. Qiu and R. Chellappa, "Subspace interpolation via dictionary learning for unsupervised domain adaptation", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 692–699 (2013).

Nowozin, S. and C. H. Lampert, "Structured learning and prediction in computer vision", Foundations and Trends in Computer Graphics and Vision **6**, 3-4, 185–365, URL http://dx.doi.org/10.1561/0600000033 (2011).

Obozinski, G., B. Taskar and M. I. Jordan, "Joint covariate selection and joint subspace selection for multiple classification problems", Statistics and Computing **20**, 2, 231–252 (2010).

Oord, A. v. d., N. Kalchbrenner and K. Kavukcuoglu, "Pixel recurrent neural networks", arXiv preprint arXiv:1601.06759 (2016).

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

Palatucci, M., D. Pomerleau, G. E. Hinton and T. M. Mitchell, "Zero-shot learning with semantic output codes", in "Advances in Neural Information Processing Systems (NIPS)", pp. 1410–1418 (2009).

Pan, S. J., J. T. Kwok and Q. Yang, "Transfer learning via dimensionality reduction.", in "Proceedings of the AAAI Conf. on Artificial Intelligence", vol. 8, pp. 677–682 (2008).

Pan, S. J., I. W. Tsang, J. T. Kwok and Q. Yang, "Domain adaptation via transfer component analysis", IEEE Trans. on Neural Networks **22**, 2, 199–210 (2011).

Pan, S. J. and Q. Yang, "A survey on transfer learning", IEEE Trans. on Knowledge and Data Engineering **22**, 10, 1345–1359 (2010).

Panchanathan, S., T. McDaniel and V. Balasubramanian, "Person-centered accessible technologies: Improved usability and adaptation through inspirations from disability research", in "Proceedings of the 2012 ACM workshop on User experience in e-learning and augmented technologies in education", pp. 1–6 (ACM, 2012).

Pantic, M., M. Valstar, R. Rademaker and L. Maat, "Web-based database for facial expression analysis", in "Proceedings of the IEEE Conf. on Multimedia and Expo (ICME)", (2005).

Papailiopoulos, D., I. Mitliagkas, A. Dimakis and C. Caramanis, "Finding dense subgraphs via low-rank bilinear optimization", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 1890–1898 (2014).

Papailiopoulos, D. S., A. G. Dimakis and S. Korokythakis, "Sparse PCA through low-rank approximations.", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 747–755 (2013).

Patel, V. M., R. Gopalan, R. Li and R. Chellappa, "Visual domain adaptation: A survey of recent advances", IEEE signal processing magazine **32**, 3, 53–69 (2015).

Peng, H., F. Long and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy", IEEE Trans. on Pattern Analysis and Machine Intelligence **27**, 8, 1226–1238 (2005).

Peng, X. and K. Saenko, "Synthetic to real adaptation with deep generative correlation alignment networks", arXiv preprint arXiv:1701.05524 (2017).

Qiu, Q., V. M. Patel, P. Turaga and R. Chellappa, "Domain adaptive dictionary learning", in "Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 631–645 (2012).

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

Raina, R., A. Battle, H. Lee, B. Packer and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 759–766 (2007).

Razavian, A. S., H. Azizpour, J. Sullivan and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition", in "Workshops, Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPRW)", pp. 512–519 (2014).

Rebuffi, S.-A., A. Kolesnikov and C. H. Lampert, "iCaRL: Incremental classifier and representation learning", in "accepted to the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2017).

Reddy, K. K. and M. Shah, "Recognizing 50 human action categories of web videos", Machine Vision and Applications **24**, 5, 971–981 (2013).

Reed, S., Z. Akata, X. Yan, L. Logeswaran, B. Schiele and H. Lee, "Generative adversarial text to image synthesis", in "Proceedings of The 33rd International Conference on Machine Learning", vol. 3 (2016).

Rodriguez-Lujan, I., R. Huerta, C. Elkan and C. S. Cruz, "Quadratic programming feature selection", J. Mach. Learn. Res. **11**, 1491–1516 (2010).

Roweis, S., "The usps database of handwritten digits", `http://www.cs.nyu.edu/~roweis/data.html` (2000).

Rozantsev, A., M. Salzmann and P. Fua, "Beyond sharing weights for deep domain adaptation", arXiv preprint arXiv:1603.06432 (2016).

Saenko, K., B. Kulis, M. Fritz and T. Darrell, "Adapting visual category models to new domains", in "Proceedings of the European Conf. on Computer Vision (ECCV)", (2010).

Sankaranarayanan, S., Y. Balaji, C. D. Castillo and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks", arXiv preprint arXiv:1704.01705 (2017).

Sener, O., H. O. Song, A. Saxena and S. Savarese, "Unsupervised transductive domain adaptation", arXiv preprint arXiv:1602.03534 (2016).

Shekhar, S., V. M. Patel, H. V. Nguyen and R. Chellappa, "Generalized domain-adaptive dictionaries", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 361–368 (2013).

Shimodaira, H., "Improving predictive inference under covariate shift by weighting the log-likelihood function", Journal of statistical planning and inference **90**, 2, 227–244 (2000).

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", CoRR **abs/1409.1556** (2014).

Smola, A., A. Gretton, L. Song and B. Schölkopf, "A Hilbert space embedding for distributions", in "International Conference on Algorithmic Learning Theory", pp. 13–31 (Springer, 2007).

Socher, R., M. Ganjoo, C. D. Manning and A. Ng, "Zero-shot learning through cross-modal transfer", in "Advances in Neural Information Processing Systems (NIPS)", pp. 935–943 (2013).

Srivastava, N. and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines", in "Advances in Neural Information Processing Systems (NIPS)", pp. 2222–2230 (2012).

Sugiyama, M., S. Nakajima, H. Kashima, P. V. Buenau and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation", in "Advances in Neural Information Processing Systems (NIPS)", pp. 1433–1440 (2008).

Sun, B., J. Feng and K. Saenko, "Return of frustratingly easy domain adaptation", in "Workshops, Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV TASK-CV)", (2015a).

Sun, B. and K. Saenko, "Deep Coral: Correlation alignment for deep domain adaptation", in "Workshops, Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 443–450 (2016).

Sun, C., S. Shetty, R. Sukthankar and R. Nevatia, "Temporal localization of fine-grained actions in videos by domain transfer from web images", in "Proceedings of the ACM Intl. Conf. on Multimedia (ACM-MM)", pp. 371–380 (2015b).

Sun, Q., R. Chattopadhyay, S. Panchanathan and J. Ye, "A two-stage weighting framework for multi-source domain adaptation", in "Advances in Neural Information Processing Systems (NIPS)", pp. 505–513 (2011).

Thrun, S., "Is learning the n-th thing any easier than learning the first?", in "Advances in neural information processing systems", pp. 640–646 (MORGAN KAUFMANN PUBLISHERS, 1996).

Thrun, S. and L. Pratt, eds., *Learning to learn* (Kluwer Academic Publishers, Norwell, MA, USA, 1998).

Thrun, S. and L. Pratt, *Learning to learn* (Springer Science & Business Media, 2012).

Toh, K.-C., M. J. Todd and R. H. Tütüncü, "Sdpt3 matlab software package for semidefinite programming, version 1.3", Optimization methods and software **11**, 1-4, 545–581 (1999).

Tommasi, T., M. Lanzi, P. Russo and B. Caputo, "Learning the roots of visual domain shift", in "Workshops, Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV TASK-CV)", pp. 475–482 (Springer, 2016).

Torralba, A. and A. A. Efros, "Unbiased look at dataset bias", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 1521–1528 (2011).

Tzeng, E., J. Hoffman, T. Darrell and K. Saenko, "Simultaneous deep transfer across domains and tasks", in "Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV)", pp. 4068–4076 (2015a).

Tzeng, E., J. Hoffman, T. Darrell and K. Saenko, "Simultaneous deep transfer across domains and tasks", in "Proceedings of the IEEE Intl. Conf. on Computer Vision (ICCV)", (2015b).

Tzeng, E., J. Hoffman, K. Saenko and T. Darrell, "Adversarial discriminative domain adaptation", Technical Report (2017).

Tzeng, E., J. Hoffman, N. Zhang, K. Saenko and T. Darrell, "Deep domain confusion: Maximizing for domain invariance", arXiv preprint arXiv:1412.3474 (2014).

Uguroglu, S. and J. Carbonell, "Feature selection for transfer learning", in "Machine Learning and Knowledge Discovery in Databases", pp. 430–442 (Springer, 2011).

Van der Maaten, L. and G. Hinton, "Visualizing data using t-sne", Journal of Machine Learning Research **9**, 2579-2605, 85 (2008).

Vapnik, V., *The nature of statistical learning theory* (Springer science & business media, 2013).

Vedaldi, A. and K. Lenc, "Matconvnet – convolutional neural networks for MATLAB", in "Proceedings of the ACM Intl. Conf. on Multimedia (ACM-MM)", (2015).

Venkateswara, H., V. N. Balasubramanian, P. Lade and S. Panchanathan, "Multiresolution match kernels for gesture video classification", in "Proceedings of the IEEE Intl. Conf. on Multimedia and Expo Workshops (ICME)", pp. 1–4 (2013).

Venkateswara, H., S. Chakraborty, T. McDaniel and S. Panchanathan, "Model selection with nonlinear embedding for unsupervised domain adaptation", in "KnowPros Workshop - Proceedings of the AAAI Conf. on Artificial Intelligence", (2017a).

Venkateswara, H., S. Chakraborty and S. Panchanathan, "Nonlinear embedding transform for unsupervised domain adaptation", in "Workshops, Proceedings of the European Conf. on Computer Vision (ECCV)", pp. 451–457 (Springer, 2016).

Venkateswara, H., J. Eusebio, S. Chakraborty and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation", in "accepted to the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2017b).

Venkateswara, H., P. Lade, B. Lin, J. Ye and S. Panchanathan, "Efficient approximate solutions to mutual information based global feature selection", in "Proceedings of the (IEEE) Intl. conf. on Data Mining (ICDM)", pp. 1009–1014 (2015a).

Venkateswara, H., P. Lade, J. Ye and S. Panchanathan, "Coupled support vector machines for supervised domain adaptation", in "Proceedings of the ACM Intl. Conf. on Multimedia (ACM-MM)", pp. 1295–1298 (2015b).

Wang, J., H. T. Shen, J. Song and J. Ji, "Hashing for similarity search: A survey", arXiv preprint arXiv:1408.2927 (2014).

Weinberger, K., A. Dasgupta, J. Langford, A. Smola and J. Attenberg, "Feature hashing for large scale multitask learning", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 1113–1120 (2009).

Widmer, C., M. Kloft, N. Görnitz, G. Rätsch, P. Flach, T. De Bie and N. Cristianini, "Efficient training of graph-regularized multitask SVMs", in "Proceedings of the European Conference on Machine Learning, (ECML)", (2012).

Wu, C., W. Wen, T. Afzal, Y. Zhang, Y. Chen and H. Li, "A compact dnn: Approaching googlenet-level accuracy of classification and domain adaptation", in "accepted to the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", (2017).

Yang, H. H. and J. Moody, "Data visualization and feature selection: New algorithms for non-gaussian data", Advances in Neural Information Processing Systems (NIPS) **12** (1999).

Yang, J., R. Yan and A. G. Hauptmann, "Adapting SVM classifiers to data with shifted distributions", in "Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on", pp. 69–76 (2007a).

Yang, J., R. Yan and A. G. Hauptmann, "Cross-domain video concept detection using adaptive SVMs", in "Proceedings of the ACM Intl. Conf. on Multimedia (ACM-MM)", pp. 188–197 (2007b).

Yang, J., K. Yu, Y. Gong and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 1794–1801 (2009).

Yao, T., Y. Pan, C.-W. Ngo, H. Li and T. Mei, "Semi-supervised domain adaptation with subspace learning for visual recognition", in "Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)", pp. 2142–2150 (2015).

Yosinski, J., J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?", in "Advances in Neural Information Processing Systems (NIPS)", pp. 3320–3328 (2014).

Yuan, X.-T. and T. Zhang, "Truncated power method for sparse eigenvalue problems", J. Mach. Learn. Res. **14**, 1, 899–925 (2013).

Zadrozny, B., "Learning and evaluating classifiers under sample selection bias", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", p. 114 (2004).

Zhang, C., L. Zhang and J. Ye, "Generalization bounds for domain adaptation", in "Advances in Neural Information Processing Systems (NIPS)", pp. 3320–3328 (2012).

Zhang, H., T. Xu, H. Li, S. Zhang, X. Huang, X. Wang and D. Metaxas, "Stack-GAN: Text to photo-realistic image synthesis with stacked generative adversarial networks", arXiv preprint arXiv:1612.03242 (2016).

Zhang, K., K. Muandet, Z. Wang *et al.*, "Domain adaptation under target and conditional shift", in "Proceedings of the ACM Intl. Conf. on Machine Learning (ICML)", pp. 819–827 (2013).

Zhu, H., M. Long, J. Wang and Y. Cao, "Deep hashing network for efficient similarity retrieval", in "Proceedings of the Thirtieth Conference on the Association for the Advancement of Artificial Intelligence", (2016).

APPENDIX A

LOWER BOUND FOR BQP

This section estimates the goodness of approximation to BQP provided by the solution to LP2. An equivalent problem to BQP is defined as,

Proposition A.0.1.

$$q : \{0,1\}^n \to (-\infty, 0], \ \textit{where}$$
$$q(\mathbf{x}) = BQP - ||\mathbf{Q}||_1$$

is equivalent to BQP.

Proof. $||\mathbf{Q}||_1$ is the sum of all elements in $\mathbf{Q}$. Since $\mathbf{Q}_{ij} \geq 0$, BQP $\leq ||\mathbf{Q}||_1 \ \forall \mathbf{x}$. Therefore, $q(\mathbf{x}) \leq 0, \ \forall \mathbf{x}$. Since $||\mathbf{Q}||_1$ is a constant for a matrix, under the same set of constraints,

$$\underset{\mathbf{x}}{\operatorname{argmax}} \, \text{BQP} \equiv \underset{\mathbf{x}}{\operatorname{argmax}} \, q(\mathbf{x})$$

■

A few new terms are defined before the derivation of the bound. Let $\mathbf{x}^*$ be the solution of BQP. Let $\bar{\mathbf{x}}$ be the solution of LP2. Since $||\mathbf{Q}||_1 = \sum_{ij} \mathbf{Q}_{ij}$, $||\mathbf{Q}||_1$ can be expanded in terms of any binary vector $\mathbf{x}$. Specifically $||\mathbf{Q}||_1$, is defined in terms of $\bar{\mathbf{x}}$,

Definition A.0.1.

$$||\mathbf{Q}||_1 = Q^0 + Q^1 + Q^2 \ \textit{where}, \tag{A.1}$$
$$Q^0 \leftarrow \sum_{i,j|\bar{x}_i + \bar{x}_j = 0} \mathbf{Q}_{ij} \tag{A.2}$$
$$Q^1 \leftarrow \sum_{i,j|\bar{x}_i + \bar{x}_j = 1} \mathbf{Q}_{ij} \tag{A.3}$$
$$Q^2 \leftarrow \sum_{i,j|\bar{x}_i + \bar{x}_j = 2} \mathbf{Q}_{ij} \equiv \bar{\mathbf{x}}^\top \mathbf{Q} \bar{\mathbf{x}} \tag{A.4}$$

Lemma A.0.1. $||\mathbf{Q}||_1 - \bar{\mathbf{x}}^\top \mathbf{Q} \bar{\mathbf{x}} \geq Q^1$

Proof. From (A.1),

$$||\mathbf{Q}||_1 = Q^0 + Q^1 + Q^2$$
$$||\mathbf{Q}||_1 \geq Q^1 + Q^2$$
$$||\mathbf{Q}||_1 - \bar{\mathbf{x}}^\top \mathbf{Q} \bar{\mathbf{x}} \geq Q^1 \text{ using (A.4)}$$

■

Let Q^* denote the maximum value of BQP and let Q^*_{LP2} denote maximum value of LP2. If $\mathbf{x}^*$ is the solution of BQP and $\bar{\mathbf{x}}$ is the solution of LP2. The following result is obtained:

Lemma A.0.2. $Q^*_{LP2} \geq Q^*$

Proof.

$$\begin{aligned} Q^*_{LP2} &= \max ||\mathbf{Q}\mathbf{x}||_1 \\ &= ||\mathbf{Q}\bar{\mathbf{x}}||_1 \\ &\geq ||\mathbf{Q}\mathbf{x}^*||_1 \\ &\geq \mathbf{x}^{*\top}\mathbf{Q}\mathbf{x}^* \\ &= Q^* \end{aligned}$$

■

The bound for LP2 can now be stated.

Theorem A.0.1.

$$2q(\mathbf{x}^*) \leq q(\bar{\mathbf{x}}) \tag{A.5}$$

Proof. From Lemma (A.0.2):

$$\mathbf{x}^{*\top}\mathbf{Q}\mathbf{x}^* \leq \frac{1}{2}\sum_{ij}\mathbf{Q}_{ij}(\bar{x}_i + \bar{x}_i) \tag{A.6}$$

$$\mathbf{x}^{*\top}\mathbf{Q}\mathbf{x}^* \leq \frac{1}{2}Q^1 + \bar{\mathbf{x}}^\top\mathbf{Q}\bar{\mathbf{x}} \qquad \text{(A.3, A.4)} \tag{A.7}$$

$$2\mathbf{x}^{*\top}\mathbf{Q}\mathbf{x}^* \leq Q^1 + 2\bar{\mathbf{x}}^\top\mathbf{Q}\bar{\mathbf{x}} \tag{A.8}$$

$$2\mathbf{x}^{*\top}\mathbf{Q}\mathbf{x}^* \leq ||\mathbf{Q}||_1 + \bar{\mathbf{x}}^\top\mathbf{Q}\bar{\mathbf{x}} \qquad \text{Lemma (A.0.1)} \tag{A.9}$$

$$2q(\mathbf{x}^*) \leq q(\bar{\mathbf{x}}) \tag{A.10}$$

■

The last statement (A.10) is arrived at by adding $-2||\mathbf{Q}||_1$ on both sides. Since $q(\mathbf{x}) \leq 0$, $2q(\mathbf{x}^*) \leq q(\bar{\mathbf{x}})$ implies that $q(\bar{\mathbf{x}})$ is a lower bound for $q(\mathbf{x}^*)$. This guarantees a lower bound for BQP by solving LP2 and Equation (A.10) provides the tightness of the bound.

APPENDIX B

DERIVATIVES FOR THE DAH LOSS FUNCTION

In this section the partial derivative of Equation (6.8) for the backpropagation algorithm is outlined;

$$\min_{\mathcal{U}} \mathcal{J} = \mathcal{L}(\mathcal{U}_s) + \gamma \mathcal{M}(\mathcal{U}_s, \mathcal{U}_t) + \eta \mathcal{H}(\mathcal{U}_s, \mathcal{U}_t), \tag{6.8}$$

where, $\mathcal{U} \coloneqq \{\mathcal{U}_s \cup \mathcal{U}_t\}$ and (γ, η) control the importance of domain adaptation (6.1) and target entropy loss (6.7) respectively. In the following subsections, the partial derivative of the individual terms w.r.t. the input $\mathcal{U}$ is outlined.

B.1 Derivative for MK-MMD

$$\mathcal{M}(\mathcal{U}_s, \mathcal{U}_t) = \sum_{l \in \mathcal{F}} d_k^2(\mathcal{U}_s^l, \mathcal{U}_t^l), \tag{6.1}$$

$$d_k^2(\mathcal{U}_s^l, \mathcal{U}_t^l) = \Big|\Big| \mathbb{E}[\phi(\boldsymbol{u}^{s,l})] - \mathbb{E}[\phi(\boldsymbol{u}^{t,l})] \Big|\Big|_{\mathscr{H}_k}^2. \tag{6.2}$$

The linear MK-MMD loss is implemented according to Gretton *et al.* (2012). For this derivation, the loss at just one layer is considered. The derivative for the MK-MMD loss at every other layer can be derived in a similar manner. The output of i^{th} source data point at layer l is represented as $\boldsymbol{u}_i$ and the output of the i^{th} target data point is represented as $\boldsymbol{v}_i$. For ease of representation, the superscripts for the source (s), the target (t) and the layer (l) are dropped. Unlike the conventional MMD loss which is $\mathcal{O}(n^2)$, the MK-MMD loss outlined in Gretton *et al.* (2012) is $\mathcal{O}(n)$ and can be estimated online (does not require all the data). The loss is calculated over every batch of data points during the back-propagation. Let n be the number of source data points $\mathcal{U} \coloneqq \{\boldsymbol{u}_i\}_{i=1}^n$ and the number of target data points $\mathcal{V} \coloneqq \{\boldsymbol{v}_i\}_{i=1}^n$ in the batch. It is assumed there are equal number of source and target data points in a batch and that n is even. The MK-MMD is defined over a set of 4 data points $\boldsymbol{w}_i = [\boldsymbol{u}_{2i-1}, \boldsymbol{u}_{2i}, \boldsymbol{v}_{2i-1}, \boldsymbol{v}_{2i}], \forall i \in \{1, 2, \ldots, n/2\}$. The MK-MMD is given by,

$$\mathcal{M}(\mathcal{U}, \mathcal{V}) = \sum_{m=1}^{\kappa} \beta_m \frac{1}{n/2} \sum_{i=1}^{n/2} h_m(\boldsymbol{w}_i), \tag{B.1}$$

where, κ is the number of kernels and $\beta_m = 1/\kappa$ is the weight for each kernel and,

$$h_m(\boldsymbol{w}_i) = k_m(\boldsymbol{u}_{2i-1}, \boldsymbol{u}_{2i}) + k_m(\boldsymbol{v}_{2i-1}, \boldsymbol{v}_{2i}) - k_m(\boldsymbol{u}_{2i-1}, \boldsymbol{v}_{2i}) - k_m(\boldsymbol{u}_{2i}, \boldsymbol{v}_{2i-1}), \tag{B.2}$$

where, $k_m(\boldsymbol{x}, \boldsymbol{y}) = \exp\big(-\frac{||\boldsymbol{x}-\boldsymbol{y}||_2^2}{\sigma_m}\big)$. Re-writing the MK-MMD in terms of the kernels:

$$\begin{aligned}\mathcal{M}(\mathcal{U}, \mathcal{V}) = \frac{2}{n\kappa} \sum_{m=1}^{\kappa} \sum_{i=1}^{n/2} \big[&k_m(\boldsymbol{u}_{2i-1}, \boldsymbol{u}_{2i}) + k_m(\boldsymbol{v}_{2i-1}, \boldsymbol{v}_{2i}) \\ &- k_m(\boldsymbol{u}_{2i-1}, \boldsymbol{v}_{2i}) - k_m(\boldsymbol{u}_{2i}, \boldsymbol{v}_{2i-1})\big],\end{aligned} \tag{B.3}$$

The partial derivative of Equation (B.3) w.r.t. source output $\boldsymbol{u}_q$ and target output $\boldsymbol{v}_q$ is outlined as,

$$\frac{\partial \mathcal{M}}{\partial \boldsymbol{u}_q} = \frac{2}{n\kappa}\sum_{m=1}^{\kappa}\sum_{i=1}^{n/2}\Big[\frac{2}{\sigma_m}k_m(\boldsymbol{u}_{2i-1},\boldsymbol{u}_{2i}).(\boldsymbol{u}_{2i-1}-\boldsymbol{u}_{2i}).(\mathcal{I}\{q=2i\}-\mathcal{I}\{q=2i-1\}) + \frac{2}{\sigma_m}k_m(\boldsymbol{u}_{2i-1},\boldsymbol{v}_{2i}).(\boldsymbol{u}_{2i-1}-\boldsymbol{v}_{2i}).\mathcal{I}\{q=2i-1\} + \frac{2}{\sigma_m}k_m(\boldsymbol{u}_{2i},\boldsymbol{v}_{2i-1}).(\boldsymbol{u}_{2i}-\boldsymbol{v}_{2i-1}).\mathcal{I}\{q=2i\}\Big], \tag{B.4}$$

where, $\mathcal{I}\{.\}$ is the indicator function which is 1 if the condition is true, else it is false. The partial derivative w.r.t. the target data output $\boldsymbol{v}_q$ is,

$$\frac{\partial \mathcal{M}}{\partial \boldsymbol{v}_q} = \frac{2}{n\kappa}\sum_{m=1}^{\kappa}\sum_{i=1}^{n/2}\Big[\frac{2}{\sigma_m}k_m(\boldsymbol{v}_{2i-1},\boldsymbol{v}_{2i}).(\boldsymbol{v}_{2i-1}-\boldsymbol{v}_{2i}).(\mathcal{I}\{q=2i\}-\mathcal{I}\{q=2i-1\}) - \frac{2}{\sigma_m}k_m(\boldsymbol{u}_{2i-1},\boldsymbol{v}_{2i}).(\boldsymbol{u}_{2i-1}-\boldsymbol{v}_{2i}).\mathcal{I}\{q=2i\} - \frac{2}{\sigma_m}k_m(\boldsymbol{u}_{2i},\boldsymbol{v}_{2i-1}).(\boldsymbol{u}_{2i}-\boldsymbol{v}_{2i-1}).\mathcal{I}\{q=2i-1\}\Big], \tag{B.5}$$

B.2 Derivative for Supervised Hash Loss

The supervised hash loss is given by,

$$\min_{\mathcal{U}_s}\mathcal{L}(\mathcal{U}_s) = -\sum_{s_{ij}\in\mathcal{S}}\Big(s_{ij}\boldsymbol{u}_i^\top\boldsymbol{u}_j - \log\big(1+\exp(\boldsymbol{u}_i^\top\boldsymbol{u}_j)\big)\Big) + \sum_{i=1}^{n_s}\big|\big|\boldsymbol{u}_i - \mathrm{sgn}(\boldsymbol{u}_i)\big|\big|_2^2. \tag{6.5}$$

The partial derivative of Equation (6.5) w.r.t. source data output $\boldsymbol{u}_p$ is given by,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_q} = \sum_{s_{ij}\in\mathcal{S}}\Big[I\{i=q\}\big(\sigma(\boldsymbol{u}_i^\top\boldsymbol{u}_j)-s_{ij}\big)\boldsymbol{u}_j + I\{j=q\}\big(\sigma(\boldsymbol{u}_i^\top\boldsymbol{u}_j)-s_{ij}\big)\boldsymbol{u}_i\Big] + 2(\boldsymbol{u}_q - \mathrm{sgn}(\boldsymbol{u}_q)) \tag{B.6}$$

where, $\sigma(x) = \frac{1}{1+exp(-x)}$. It is assumed sgn(.) is a constant in order to avoid the differentiability issues with sgn(.) at 0. Since $\mathcal{S}$ is symmetric, the partial derivative can be reduced to,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_q} = \sum_{j=1}^{n_s}\Big[2\big(\sigma(\boldsymbol{u}_q^\top\boldsymbol{u}_j)-s_{qj}\big)\boldsymbol{u}_j\Big] + 2\big(\boldsymbol{u}_q - \mathrm{sgn}(\boldsymbol{u}_q)\big). \tag{B.7}$$

B.3 Derivative for Unsupervised Entropy Loss

The partial derivative of $\frac{d\mathcal{H}}{d\mathcal{U}}$ is outlined in the following section, where $\mathcal{H}$ is defined as,

$$\mathcal{H}(\mathcal{U}_s, \mathcal{U}_t) = -\frac{1}{n_t}\sum_{i=1}^{n_t}\sum_{j=1}^{C} p_{ij}\log(p_{ij}) \tag{6.7}$$

and p_{ij} is the probability of target data output $\boldsymbol{u}_i^t$ belonging to category j, given by

$$p_{ij} = \frac{\sum_{k=1}^{K}\exp(\boldsymbol{u}_i^{t\top}\boldsymbol{u}_k^{s_j})}{\sum_{l=1}^{C}\sum_{k'=1}^{K}\exp(\boldsymbol{u}_i^{t\top}\boldsymbol{u}_{k'}^{s_l})} \tag{6.6}$$

For ease of representation, the target output $\boldsymbol{u}_i^t$ is denoted as $\boldsymbol{v}_i$ and superscript t is dropped. Similarly, the k^{th} source data point in the j^{th} category $\boldsymbol{u}_k^{s_j}$ is denoted as $\boldsymbol{u}_k^j$, after dropping the domain superscript. The probability p_{ij} with the new terms is now,

$$p_{ij} = \frac{\sum_{k=1}^{K}\exp(\boldsymbol{v}_i{}^{\top}\boldsymbol{u}_k^j)}{\sum_{l=1}^{C}\sum_{k'=1}^{K}\exp(\boldsymbol{v}_i{}^{\top}\boldsymbol{u}_{k'}^l)} \tag{B.8}$$

Further simplification is achieved by replacing $\exp(\boldsymbol{v}_i^{\top}\boldsymbol{u}_k^j)$ with $\exp(i,jk)$. Equation (B.8) can now be represented as,

$$p_{ij} = \frac{\sum_{k=1}^{K}\exp(i,jk)}{\sum_{l=1}^{C}\sum_{k'=1}^{K}\exp(i,lk')} \tag{B.9}$$

The outer summations are dropped (along with the -ve sign) and will be reintroduced at a later time. The entropy loss can be re-phrased using $\log(\frac{a}{b}) = \log(a)$ - $\log(b)$ as,

$$\mathcal{H}_{ij} = \frac{\sum_{k=1}^{K}\exp(i,jk)}{\sum_{l=1}^{C}\sum_{k'=1}^{K}\exp(i,lk')}\log\big(\textstyle\sum_{k=1}^{K}\exp(i,jk)\big) \tag{B.10}$$

$$- \frac{\sum_{k=1}^{K}\exp(i,jk)}{\sum_{l=1}^{C}\sum_{k'=1}^{K}\exp(i,lk')}\log\big(\textstyle\sum_{l=1}^{C}\sum_{k'=1}^{K}\exp(i,lk')\big) \tag{B.11}$$

Both, $\frac{\partial\mathcal{H}_{ij}}{\partial\boldsymbol{v}_i}$ for the target and $\frac{\partial\mathcal{H}_{ij}}{\partial\boldsymbol{u}_q^p}$ for the source need to be estimated. $\partial\boldsymbol{u}_q^p$ is used to refer to source data. The partial derivative $\frac{\partial\mathcal{H}_{ij}}{\partial\boldsymbol{u}_q^p}$ for Equation (B.10) is,

$$\left[\frac{\partial\mathcal{H}_{ij}}{\partial\boldsymbol{u}_q^p}\right]_{B.10} = \frac{\boldsymbol{v}_i}{\sum_{l,k'}\exp(i,lk')}\Big[\textstyle\sum_k I\{_{k=q}^{j=p,}\}\exp(i,jk).\log\big(\sum_k\exp(i,jk)\big)$$
$$+ \textstyle\sum_k I\{_{k=q}^{j=p,}\}\exp(i,jk) - p_{ij}\exp(i,pq)\log\big(\sum_k\exp(i,jk)\big)\Big], \tag{B.12}$$

where, $I\{.\}$ is an indicator function, which is 1 only when both the conditions within are true, else it is 0. The partial derivative $\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{u}_q^p}$ for Equation (B.11) is,

$$\begin{aligned}\left[\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{u}_q^p}\right]_{B.11} =& -\frac{\boldsymbol{v}_i}{\sum_{l,k'}\exp(i,lk')}\Big[\sum_k I\{_{k=q}^{j=p,}\}\exp(i,jk).\log\big(\sum_{l,k'}\exp(i,lk')\big) \\ &+ p_{ij}\exp(i,pq) - p_{ij}\exp(i,pq)\log\big(\sum_{l,k'}\exp(i,lk')\big)\Big] \end{aligned} \tag{B.13}$$

Expressing $\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{u}_q^p} = \left[\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{u}_q^p}\right]_{B.10} + \left[\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{u}_q^p}\right]_{B.11}$, and defining $\bar{p}_{ijk} = \frac{\exp(i,jk)}{\sum_{l,k'}\exp(i,lk')}$ the partial derivative w.r.t. the source is,

$$\begin{aligned}\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{u}_q^p} =& \boldsymbol{v}_i\Big[\sum_k I\{_{k=q}^{j=p,}\}\bar{p}_{ijk}.\log\big(\sum_k\exp(i,jk)\big) + \sum_k I\{_{k=q}^{j=p,}\}\bar{p}_{ijk} \\ &- p_{ij}\bar{p}_{ipq}\log\big(\sum_k\exp(i,jk)\big) - \sum_k I\{_{k=q}^{j=p,}\}\bar{p}_{ijk}.\log\big(\sum_{l,k'}\exp(i,lk')\big) \\ &- p_{ij}\bar{p}_{ipq} + p_{ij}\bar{p}_{ipq}\log\big(\sum_{l,k'}\exp(i,lk')\big)\Big] \end{aligned} \tag{B.14}$$

$$=\boldsymbol{v}_i\Big[\sum_k I\{_{k=q}^{j=p,}\}\bar{p}_{ijk}\log(p_{ij}) - p_{ij}\bar{p}_{ipq}\log(p_{ij}) + \sum_k I\{_{k=q}^{j=p,}\}\bar{p}_{ijk} - p_{ij}\bar{p}_{ipq}\Big]$$

$$=\boldsymbol{v}_i\big(\log(p_{ij})+1\big)\Big[\sum_k I\{_{k=q}^{j=p,}\}\bar{p}_{ijk} - p_{ij}\bar{p}_{ipq}\Big] \tag{B.15}$$

The partial derivative of $\mathcal{H}$ w.r.t the **source** output $\boldsymbol{u}_q^p$ is given by,

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}_q^p} = -\frac{1}{n_t}\sum_{i=1}^{n_t}\sum_{j=1}^{C}\boldsymbol{v}_i\big(\log(p_{ij})+1\big)\Big[\sum_k I\{_{k=q}^{j=p,}\}\bar{p}_{ijk} - p_{ij}\bar{p}_{ipq}\Big] \tag{B.16}$$

The partial derivative $\frac{\partial \mathcal{H}}{\partial \boldsymbol{v}_i}$ for Equation (B.10) is outlined as,

$$\begin{aligned}\left[\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{v}_i}\right]_{B.10} =& \frac{1}{\sum_{l,k'}\exp(i,lk')}\Big[\log\big(\sum_k\exp(i,jk)\big)\sum_k\exp(i,jk)\boldsymbol{u}_k^j + \sum_k\exp(i,jk)\boldsymbol{u}_k^j \\ &- \frac{1}{\sum_{l,k'}\exp(i,lk')}\sum_k\exp(i,jk)\log\big(\sum_k\exp(i,jk)\big)\sum_{l,k'}\exp(i,lk')\boldsymbol{u}_{k'}^l\Big], \end{aligned} \tag{B.17}$$

and the partial derivative $\frac{\partial \mathcal{H}}{\partial \boldsymbol{v}_i}$ for Equation (B.11) as,

$$\begin{aligned}\left[\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{v}_i}\right]_{B.11} =& -\frac{1}{\sum_{l,k'}\exp(i,lk')}\Big[\log\big(\sum_{l,k'}\exp(i,lk')\big)\sum_k\exp(i,jk)\boldsymbol{u}_k^j \\ &+ \frac{\sum_k\exp(i,jk)}{\sum_{l,k'}\exp(i,lk')}\sum_{l,k'}\exp(i,lk')\boldsymbol{u}_{k'}^l \\ &- \frac{1}{\sum_{l,k'}\exp(i,lk')}\sum_k\exp(i,jk)\log\big(\sum_{l,k'}\exp(i,lk')\big)\sum_{l,k'}\exp(i,lk')\boldsymbol{u}_{k'}^l\Big], \end{aligned} \tag{B.18}$$

Expressing $\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{v}_i} = \Big[\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{v}_i}\Big]_{B.10} + \Big[\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{v}_i}\Big]_{B.11}$,

$$
\begin{aligned}
\frac{\partial \mathcal{H}_{ij}}{\partial \boldsymbol{v}_i} = \frac{1}{\sum_{l,k'} \exp(i, lk')} \Big[& \log\big(\textstyle\sum_k \exp(i,jk)\big) \textstyle\sum_k \exp(i,jk) \boldsymbol{u}_k^j \\
& - \log\big(\textstyle\sum_{l,k'} \exp(i,lk')\big) \textstyle\sum_k \exp(i,jk) \boldsymbol{u}_k^j \\
& + \textstyle\sum_k \exp(i,jk) \boldsymbol{u}_k^j - p_{ij} \textstyle\sum_{l,k'} \exp(i,lk') \boldsymbol{u}_{k'}^l \\
& - p_{ij} \log\big(\textstyle\sum_k \exp(i,jk)\big) \textstyle\sum_{l,k'} \exp(i,lk') \boldsymbol{u}_{k'}^l \\
& + p_{ij} \log\big(\textstyle\sum_{l,k'} \exp(i,lk')\big) \textstyle\sum_{l,k'} \exp(i,lk') \boldsymbol{u}_{k'}^l \Big] \qquad \text{(B.19)} \\
= \Big[& \log\big(\textstyle\sum_k \exp(i,jk)\big) \textstyle\sum_k \bar{p}_{ijk} \boldsymbol{u}_k^j - \log\big(\textstyle\sum_{l,k'} \exp(i,lk')\big) \textstyle\sum_k \bar{p}_{ijk} \boldsymbol{u}_k^j \\
& + \textstyle\sum_k \bar{p}_{ijk} \boldsymbol{u}_k^j - p_{ij} \textstyle\sum_{l,k'} \bar{p}_{ijk'} \boldsymbol{u}_{k'}^l \\
& - p_{ij} \log\big(\textstyle\sum_k \exp(i,jk)\big) \textstyle\sum_{l,k'} \bar{p}_{ijk'} \boldsymbol{u}_{k'}^l + p_{ij} \log\big(\textstyle\sum_{l,k'} \exp(i,lk')\big) \textstyle\sum_{l,k'} \bar{p}_{ijk'} \boldsymbol{u}_{k'}^l \Big] \\
= & \big(\log(p_{ij}) + 1\big) \textstyle\sum_k \bar{p}_{ijk} \boldsymbol{u}_k^j - \big(\log(p_{ij}) + 1\big) p_{ij} \textstyle\sum_{l,k'} \bar{p}_{ijk'} \boldsymbol{u}_{k'}^l \qquad \text{(B.20)} \\
= & \big(\log(p_{ij}) + 1\big) \big(\textstyle\sum_k \bar{p}_{ijk} \boldsymbol{u}_k^j - p_{ij} \textstyle\sum_{l,k'} \bar{p}_{ijk'} \boldsymbol{u}_{k'}^l \big) \qquad \text{(B.21)}
\end{aligned}
$$

The partial derivative of $\mathcal{H}$ w.r.t. **target** output $\boldsymbol{v}_q$ is given by,

$$
\frac{\partial \mathcal{H}}{\partial \boldsymbol{v}_q} = -\frac{1}{n_t} \sum_{j=1}^{C} \big(\log(p_{qj}) + 1\big) \big(\textstyle\sum_k \bar{p}_{qjk} \boldsymbol{u}_k^j - p_{qj} \textstyle\sum_{l,k'} \bar{p}_{qjk'} \boldsymbol{u}_{k'}^l \big) \qquad \text{(B.22)}
$$

The partial derivative of $\mathcal{H}$ w.r.t. the source outputs is given by Equation (B.16) and w.r.t. the target outputs is given by Equation (B.22).

APPENDIX C

PERMISSION STATEMENTS FROM CO-AUTHORS

Permission for including co-authored material in this dissertation was obtained from co-authors, Prof. Sethuraman Panchanathan, Prof. Jieping Ye, Dr. Vineeth Balasubramanian, Dr. Troy McDaniel, Dr. Shayok Chakraborty, Dr. Binbin Lin, Dr. Prasanth Lade and Jose Eusebio.