Domain adaptation : methods and applications

Wei, Pengfei

2018

Wei, P. (2018). Domain adaptation : methods and applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/82943>

<https://doi.org/10.32657/10220/47539>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**DOMAIN ADAPTATION: METHODS AND
APPLICATIONS**

PENGFEI WEI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2019

DOMAIN ADAPTATION: METHODS AND APPLICATIONS

PENGFEI WEI

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirements for the degree of
Doctor of Philosophy

2019

Acknowledgements

At first, I would like to express my sincere appreciation to my supervisor Dr. Yiping Ke and my co-supervisor Dr. Chi Keong Goh for their constructive instructions and consecutive support through my Ph.D research. I have a very impressive experience working with them in the past four years, and I have benefited a lot from their insightful comments on my research works. I am also very thankful for Dr. Ke's help in building rigorous research attitude and improving writing and presentation skills through the four years.

I sincerely appreciate Dr. Yew Soon Ong and Dr. Ramon Sagarna. They are excellent collaborators who give me many helpful comments and feedback on my research works. With their help, my work becomes more comprehensive and solid.

I would also like to thank all my friends and staffs in RR@NTU corporate Lab for their technical supports and enthusiastic help. They give me a comfortable working and living environment so that I can have a joyful and relax mood in my research studies.

Finally, I would like to give special thanks to my parents for their consecutive support, understanding and love to me.

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

Date: 2019/01/21

Name: Wei Pengfei

Date: 2019/01/21

Signature:

A handwritten signature in black ink, appearing to read '魏鹏飞' (Wei Pengfei), written in a cursive style.

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

Date: 2019/01/21

Name: Ke Yiping

Date: 2019/01/21

Signature:

Authorship Attribution Statement

This thesis contains material from 4 paper(s) published in the following peer-reviewed conferences and journals where I was the first author.

Chapter 3 is published as Pengfei Wei, Yiping Ke, Chi Keong Goh. “Feature Analysis of Marginalized Stacked Denoising Autoencoder for Unsupervised Domain Adaptation”, IEEE Transaction on Neural Networks and Learning Systems. 1-14(2018). DOI: 10.1109/TNNLS.2018.2868709.

The contributions of the co-authors are as follows:

- Prof Ke provided instructive comments on the idea of the work.
- I co-designed the study with Prof Ke and I performed all the laboratory work at the RR@NTU corporate Lab. I also analyzed the data and the results.
- I prepared the manuscript drafts. The manuscript was revised by Prof Ke and Dr. Goh.

Chapter 4 is published as Pengfei Wei, Yiping Ke, Chi Keong Goh. “Deep Non-linear Feature Coding for Unsupervised Domain Adaptation”, International Joint Conference on Artificial Intelligence, 2016, 2189-2195.

The contributions of the co-authors are as follows:

- Prof Ke provided instructive comments on the idea of the work.
- I co-designed the experimental study with Prof Ke and I performed all the laboratory work at the RR@NTU corporate Lab. I also analyzed the data and the results.
- I prepared the manuscript drafts. The manuscript was revised by Prof Ke and Dr. Goh.

Chapter 6 is published as Pengfei Wei, Yiping Ke, Chi Keong Goh. “A General Domain Specific Feature Transfer Framework for Hybrid Domain Adaptation”, IEEE Transaction on Knowledge and Data Engineering. 1-12(2018). DOI: 10.1109/TKDE.2018.2864732

The contributions of the co-authors are as follows:

- Prof Ke provided instructive comments on the idea of the work.

- I co-designed the experimental study with Prof Ke and I performed all the laboratory work at the RR@NTU corporate Lab. I also analyzed the data and the results.
- I prepared the manuscript drafts. The manuscript was revised by Prof Ke and Dr. Goh.

Chapter 7 is published as Pengfei Wei, Ramon Sagarna, Yiping Ke, Yew Soon Ong, Chi Keong Goh. “Source-Target Similarity Modelings for Multi-Source Transfer Gaussian Process Regression”, International Conference on Machine Learning, 2017, 3722-3731.

The contributions of the co-authors are as follows:

- Prof Ke, Prof Ong, and Dr. Goh provided the initial project direction.
- Dr. Ramon, Prof Ke and Prof Ong provided comments on the idea of the work.
- I co-designed the experimental study with Dr. Ramon. Dr. Ramon assisted in the implementation of the experimental studies.
- I prepared the manuscript drafts. The manuscript was revised by Dr. Ramon.

Date: 2019/01/21

Name: Wei Pengfei

Date: 2019/01/21

Signature:

Contents

Acknowledgements	3
Statement of Originality	5
Supervisor Declaration Statement	7
Authorship Attribution Statement	9
List of Tables	16
List of Figures	17
Abstract	18
1 Introduction	21
1.1 Domain Adaptation	22
1.1.1 Motivation	22
1.1.2 Definition	23
1.1.3 Domain Adaptation and Transfer Learning	24
1.1.4 Domain Adaptation Applications	25
1.1.5 Domain Adaptation Settings and Challenges	26
1.2 Main Contributions	28
1.3 Thesis Outline	30
2 Background and Related Works	33
2.1 Instance-based Domain Adaptation	34
2.1.1 Problem Definition	34
2.1.2 Weights Estimation Methods	35

2.1.3	Other Methods	37
2.2	Parameter-based Domain Adaptation	37
2.2.1	Maximum Entropy Model	38
2.2.2	Support Vector Machine	39
2.2.3	Gaussian Process Model	40
2.3	Feature-based Domain Adaptation	41
2.3.1	Homogeneous Domain Adaptation	41
2.3.1.1	Theoretical Foundation	41
2.3.1.2	Domain Divergence Metric	42
2.3.1.3	Subspace-based Methods	43
2.3.1.4	Deep Feature Learning Methods	46
2.3.2	Heterogeneous Domain Adaptation	48
3	Feature Analysis of mSDA for Domain Adaptation	51
3.1	Problem Setting	54
3.1.1	Problem Statement	54
3.1.2	The Original mDA	54
3.2	Rationale of mSDA to Domain Adaptation	55
3.2.1	Denoising in mDA	56
3.2.2	Frobenius Regularization in mDA_g	57
3.2.3	Data-dependent Low-rank Regularization in mDA_{bd}	58
3.2.4	Discussions	59
3.3	mSDA_{md}	60
3.3.1	Motivation	60
3.3.2	A Data-dependent Multinomial Dropout Noise	60
3.3.3	Data-dependent Multinomial Dropout Noise mSDA	62
3.3.4	Data-dependent Low-rank Regularization in mSDA_{md}	64
3.3.5	Discussion	64
3.4	Experimental Results	65
3.4.1	Datasets & Experimental Setting	65
3.4.2	Comparison of mSDA with Different Noises	66
3.5	Summary	68

4	Deep Nonlinear Feature Coding	70
4.1	A Deep Nonlinear Coding Approach	71
4.1.1	Problem Statement	71
4.1.2	Domain Divergence Minimization by MMD	71
4.1.2.1	Bernoulli Dropout Noise Case	72
4.1.2.2	Multinomial Dropout Noise Case	73
4.1.3	Nonlinear Coding by Kernelization	74
4.1.3.1	Bernoulli Dropout Noise Case	76
4.1.3.2	Multinomial Dropout Noise Case	76
4.1.4	Deep Nonlinear Feature Coding (DNFC)	77
4.1.5	Complexity Analysis on DNFC	77
4.2	Experimental Results	78
4.2.1	Datasets & Experimental Setting	78
4.2.2	Comparison with Baselines	79
4.2.3	Property Study on DNFC	79
4.2.4	Effect of New Elements in DNFC	82
4.3	Summary	83
5	Local Manifolds Information Transfer	84
5.1	Related Works	87
5.2	A Local Manifolds Information Transfer Approach	88
5.2.1	Manifolds Neighborhood Preservation	89
5.2.2	Local Manifolds Information Transfer Framework	91
5.2.3	Discussion	93
5.3	Experiments	94
5.3.1	Experimental Results	96
5.3.2	Multiple Manifolds Neighbors	96
5.3.3	Geometric Information Preservation	98
5.3.4	Sensitivity Analysis	99
5.4	Summary	100
6	Domain Specific Feature Transfer	102
6.1	Related Works	104
6.2	A General Domain Specific Feature Transfer Framework	105

6.2.1	Problem Statement	105
6.2.2	Domain Specific Feature Transfer	106
6.2.3	Distribution Discrepancy Minimization	107
6.2.4	Final DSFT framework	107
6.3	Instantiations of the Domain Specific Feature Transfer framework . .	108
6.3.1	A Distribution Discrepancy Metric MMD	108
6.3.2	A Linear Case DSFT _{<i>l</i>}	109
6.3.3	A Nonlinear Case DSFT _{<i>nl</i>}	111
6.4	Experimental Study	112
6.4.1	Datasets & Experimental Setting	112
6.4.2	DSFT v.s. ‘ <i>cut-off</i> ’	114
6.4.3	DSFT v.s. ‘ <i>fill-up</i> ’	116
6.4.4	Adaptation Analysis of DSFT	119
6.4.4.1	Feature Reweighting	119
6.4.4.2	New Feature Representation	120
6.4.5	Property Analysis of DSFT	121
6.4.5.1	Sensitivity Analysis	122
6.4.5.2	Term Analysis	122
6.4.5.3	Domain Specific Feature Analysis	123
6.5	Summary	124
7	Transfer Covariance Function Stacking	125
7.1	Problem Statement	128
7.2	<i>GP</i> with Transfer Covariance Function	129
7.2.1	Transfer Covariance Function for Multi-Source	129
7.2.2	Generalization Bounds of <i>GP-TC_{MS}</i>	130
7.3	Transfer Covariance Function Stacking	133
7.3.1	Conventional Transfer Stacking <i>TC_{SS}Stack</i>	133
7.3.2	Improved Transfer Stacking <i>TC_{MS}Stack</i>	134
7.3.3	Complexity Analysis	135
7.4	Experimental Study	135
7.4.1	Experiment Setting	135
7.4.2	Domain Similarity Capture	136
7.4.3	Performance on Real-World Datasets	139

7.5	Summary	140
8	Conclusions and Future Work	142
8.1	Conclusions	142
8.2	Future Works	145
8.2.1	Theoretical Foundation	145
8.2.2	Multi-source Adaptive Transfer	146
	Bibliography	147
A	Proofs	159
B	Publication List	162

List of Tables

3.1	Notations and descriptions	53
4.1	Accuracy (%) on two benchmark datasets	80
4.2	Accuracy (%) on 20-Newsgroups for assessing the effect of new elements in $DNFC_{bd}$	82
5.1	Results (%) on three benchmark datasets	95
5.2	Neighborhood statistics	98
5.3	Comparison results on the neighborhood structure preservation	99
6.1	<i>Cut-off</i> comparison results (%)	115
6.2	Discriminative power analysis	117
6.3	<i>Fill-up</i> comparison results (%)	118
6.4	Source training error (e-2) and A-distance	121
6.5	Average results (%) for term analysis	123
6.6	Domain specific feature analysis	123
7.1	Amazon review products dataset 15.	135

List of Figures

1.1	Learning process of domain adaptation	23
1.2	The Organization of Thesis	31
2.1	Basic idea of subspace-based methods	44
2.2	Heterogeneous domain adaptation using labels	49
3.1	The framework of mSDA	52
3.2	Results of mSDA with different noises on the Amazon dataset	67
3.3	Results of mSDA with different noises on the 20-Newsgroup dataset .	67
3.4	Results of mSDA with different noises on the COIL1-COIL2 dataset .	68
4.1	Deep nonlinear feature coding framework	71
4.2	Property study on DNFC _{bd}	81
5.1	Multiple manifolds assumption example	86
5.2	Neighbor-class structure in three spaces.	97
5.3	Sensitivity analysis	100
6.1	Framework of DSFT	106
6.2	NDCG for COM and DSFT	119
6.3	Sensitivity analysis on $RT_{1 \rightarrow 2}$ in DSFT _{<i>l</i>}	122
7.1	Results on 6 problem settings for each scenario of 2, 5, 10 and 15 sources.	138
7.2	Comparison results on the two datasets.	140
8.1	The Conclusive Remarks	143

Abstract

Conventional machine learning needs sufficient labeled data to achieve satisfactory generalization performance. Nonetheless, the acquiring of labeled data is expensive and time-consuming. Domain adaptation provides an efficient way to manage the data label scarcity. The objective of domain adaptation is to learn a model that generalizes good performance in a target domain where none or scarce labeled data is available, by leveraging upon the knowledge transferred from a different but related source domain with plenty of labeled data. Depending on the homogeneity of the source and target feature spaces, domain adaptation is divided into two classes: homogeneous one and heterogeneous one. In homogeneous domain adaptation, the crucial point is to align the source and target feature distributions, while in heterogeneous case, eliminating the heterogeneity of feature spaces should be done before the distributions match.

In this thesis, we develop new algorithms for homogeneous and heterogeneous domain adaptation problems, respectively. For the homogeneous case, we propose novel frameworks that aim to improve the adaptation performance by solving limitations of a state-of-the-art domain adaptation method. Regarding the heterogeneous case, we are dedicated to eliminating the heterogeneity of feature spaces without using any supplementary information that is indispensable in the existing heterogeneous adaptation studies. Except for these two conventional domain adaptation problem settings, we are also dedicated to a more general and challenging problem, which aims at fusing knowledge from multiple source domains for regression tasks. Such a problem setting is popular in various real-world applications, but has been less studied in the literatures. Specifically, three novel works are conducted.

First of all, we develop methods, specifically, one deep feature learning method and one subspace-based method, for homogeneous problem settings. Regarding the deep feature learning method, we conduct investigation and development based on marginalized stacked denoising autoencoder (mSDA). We first investigate the rationale for why mSDA can benefit adaptation tasks from the perspective of adaptive regularization. Then, we propose a new mSDA with data-dependent *Multinomial dropout* noise (mSDA_{md}), which overcomes limitations of the conventional mSDA and further improves the adaptation performance. Finally, we develop a deep nonlinear feature coding (DNFC) framework that introduces two new elements to

mSDA: domain divergence minimization by Maximum Mean Discrepancy, and non-linear coding by kernelization. Generalization of both the conventional mSDA and our proposed mSDA_{md} to DNFC framework is also investigated. Regarding the subspace-based method, we introduce a multiple manifolds assumption to domain adaptation, and develop a local manifolds information transfer framework under such an assumption. Specifically, we first propose a manifold neighborhood preservation embedding algorithm to preserve the neighborhood structure of each low-dimensional manifold in subspace learning. We then incorporate it with the global distribution discrepancy minimization into one unified framework.

Next, we propose a new problem setting which is a special case of heterogeneous domain adaptation, called hybrid domain adaptation, so that it can be solved efficiently without the help of any supplementary information. We propose a general domain specific feature transfer (DSFT) framework, which can link up two domains using common features and simultaneously reduce domain divergences. Specifically, we learn the translations between common features and domain specific features, and cross-use the learned translations to transfer domain specific features of one domain to another domain. A homogeneous space, where the domain divergences are minimized, is then composed. A linear instantiation of the DSFT framework and also a nonlinear one are presented, respectively. Extensive experiments verify the effectiveness of our proposed DSFT.

Finally, we focus on a more general problem, which aims to transfer knowledge across multiple source domains. Different from the conventional domain adaptation that assumes the source and target tasks are the same, we focus on a more challenging case where the tasks of different domains are also heterogeneous. Specifically, we are dedicated to modeling diverse task similarities through a transfer covariance function. We first investigate the feasibility and performance of a family of transfer covariance functions that represent the pairwise similarity of each source and the target domain. We theoretically prove that using such a transfer covariance function for general Gaussian process modelling can only capture the same similarity coefficient for all the sources, and thus may result in unsatisfactory adaptation performance. This leads us to propose *TC_{MS}Stack*, an integrated strategy incorporating the benefits of the transfer covariance function and stacking. Experimental studies on both synthetic and real-world datasets verify *TC_{MS}Stack*.

Keywords: Domain Adaptation, Homogeneous, Hybrid, Transfer Covariance Function

Chapter 1

Introduction

In the last decade, domain adaptation has attracted more and more attention (Pan and Yang, 2010; Margolis, 2011). It avoids the expensive and time-consuming process of collecting labeled training data, which is indispensable in conventional machine learning to obtain a satisfactory test generalization performance. Different with the assumption made in conventional machine learning, that is the train and test data are homologous, domain adaptation allows that the train and test sets are from different domains. However, domain divergence emerges since data from different domains are in different modalities, e.g., representation, distribution, dimensionality, and scale. For instance, domain 1 includes text data represented by the count of words, whereas domain 2 are image data with pixel features. These two domains have completely different data representations. Even if the data representation of domains are the same, data distributions may differ dramatically. For instance, both domain 1 and domain 2 contain images. However, the images of domain 1 and domain 2 are obtained from phone attached cameras and professional digital cameras, respectively. The data distributions are different as the images from these two devices may be taken from different directions and angles. Domain adaptation is proposed to deal with the domain divergence across different domains so that the shared knowledge can be transferred from one domain to help prediction tasks in another domain. The main objective of domain adaptation is to build a robust model that generalizes well in a target domain where none or scarce labeled data is available, by leveraging upon the knowledge from another different but related source domain with plenty of labeled data. Domain adaptation has achieved a great success in various real-world applications, for instance, sentiment classifica-

tion (Chen et al., 2012), cross language text classification (Zhou et al., 2014a), visual object recognition (Ding et al., 2015), face recognition (Wang et al., 2015), WI-FI location (Gönen and Margolin, 2014), and part-of-speech tagging (Yang et al., 2017).

In this thesis, we conduct a review on state-of-the-art domain adaptation methods. Specifically, we investigate three different domain adaptation strategies based on what is transferred across domains. Related works within each of these three adaptation strategies are discussed. Then we propose novel frameworks to solve three different domain adaptation scenarios. Technical details on these frameworks are presented, and extensive experimental results are shown to verify their effectiveness subsequently. In the end, we make conclusions.

1.1 Domain Adaptation

1.1.1 Motivation

In conventional machine learning, a test set is assumed to be homologous (in the same feature representation and drawn from the same feature distribution) with a training set so that the model trained from the training set can generalize satisfactory performance on the test set (Bishop, 2006). However, this assumption does not always hold in reality. For instance, in review sentiment classification, sentiment model is trained from book reviews, but the test is done for electronic product reviews that equip different vocabularies with book reviews (Glorot et al., 2011). The inconsistency between the train and test sets leads to a dilemma. On the one hand, the labeled data of the test set is too scarce to train a good model and collecting new labeled data is an expensive and time-consuming process. On the other hand, the performance of applying the model learned from the training set to the test set leaves much to be desired.

Motivated by the human intelligence of learning and utilizing knowledge, domain adaptation (Jiang, 2008) is proposed to alleviate the data inconsistency issue. The key idea of domain adaptation is to learn a model that works well in a target domain (refers to the test set) where none or scarce labeled data is available, by leveraging upon the knowledge from a different but related source domain (refers to the training set) with plenty of labeled data. The underlying assumption is that the source and the target domains are related, and they share some similarities which can be

Figure 1.1: Learning process of domain adaptation

transferred from one to another. Figure 1.1 shows the idea of domain adaptation in the learning process. In the learning of the target model, source knowledge such as shared features, model parameters, or instances, are transferred to help improve the performance of the target model.

1.1.2 Definition

In this section, we first state some basic concepts related to domain adaptation, and then give a formal definition of domain adaptation. Note that the concepts of domain adaptation stated below are referring to (Pan and Yang, 2010).

In a domain adaptation problem, there are two crucial concepts, namely, domain $\mathcal{D}$ and task $\mathcal{T}$. A domain consists of two components: feature space $\mathcal{X}$ and marginal probability distribution $P(X)$, where $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{X}$. Likewise, a task also consists of two components: label space $\mathcal{Y}$ and objective predictive function $f(\cdot)$. We take sentiment analysis as an example. A domain is a collection of reviews from a specific product, e.g., electronic product. Each review is represented by a binary feature vector. Consequently, the feature space $\mathcal{X}$ is the word lexicon of all review feature vectors, and $P(X)$ corresponds to the joint distribution of all the

review feature vectors. Moreover, each review is labeled as positive or negative using 1 and -1. Then the label space $\mathcal{Y}$ is $\{1, -1\}$, and the objective function $f(\cdot)$ is closely related to the conditional probability distribution $P(y_i|x_i)$ where $y_i \in \mathcal{Y}$ is the sentiment label of x_i .

Two domains are regarded to be different in two cases. The first case is that the feature spaces are the same but the marginal distributions are different. We still use sentiment analysis as an example. Reviews of two products are in the same word lexicon, and thus the feature spaces are the same. However, the frequency of the words used in the reviews may differ dramatically between different products. For instance, customers prefer to use “is_good” to demonstrate their positive attitude to a book, but use more “work_well” to give a compliment to an electronic product. The second case refers to the scenario where the feature spaces are heterogeneous. In this case, reviews of two domains may be written by different languages, and thus they have completely different word lexicons. For instance, book reviews are written in English while electronic product reviews are in Chinese. Likewise, two tasks are different when (1) the conditional distributions are different or (2) the label spaces are different. In the sentiment analysis example, the former case happens when the sentiment of a word varies between different products. An example is the word “noisy”, which is a negative word in electronic product reviews, but a neutral word in book reviews. The latter case refers to the scenario where book reviews are labeled using sentiment polar but electronic products reviews are labeled using stars.

A domain adaptation problem is defined when two domains are different but the tasks related to the two domains are the same. A formal definition of domain adaptation is given by the survey paper (Pan and Yang, 2010) as follows.

Definition 1.1. *Domain Adaptation.* Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, domain adaptation aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ but $\mathcal{T}_S = \mathcal{T}_T$.

1.1.3 Domain Adaptation and Transfer Learning

Transfer learning is a widely used terminology, which is closely related to domain adaptation. In (Pan and Yang, 2010), a definition for transfer learning is also given.

Definition 1.2. *Transfer Learning.* Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

According to the definition of transfer learning, we can see that transfer learning is a more general concept. When any of the four elements in task and domain is different, it is a transfer learning problem. As categorized in (Pan and Yang, 2010), domain adaptation is one of the sub-problem settings in transfer learning, where domains are different but tasks are the same. However, the definition of transfer learning is not unified in academy. In many research works, “domain adaptation”, “concept drift”, “multi-task learning”, “transfer learning”, “robust learning”, and “sample selection bias” are all terms used to describe related scenarios in which the knowledge is transferred from one learning process to another learning process (Margolis, 2011). One point needs to be highlighted: multi-task learning focuses on the prediction performance on multi-tasks or multiple domains simultaneously, while others just focus on one domain (usually the target domain). In (Pan and Yang, 2010), transfer learning is the collective name of others, but it is not always the case in other literatures (Huang et al., 2006; Margolis, 2011; Schwaighofer et al., 2004). For example, in (Margolis, 2011), domain adaptation attempts to align the mismatch of joint distribution $p(\mathcal{X}, \mathcal{Y})$, but not that of marginal distribution $p(\mathcal{X})$ only. In this thesis, without special statements, we use the terminology “domain adaptation” in definition 1.1 as default.

1.1.4 Domain Adaptation Applications

Domain adaptation has achieved a great success in various real-world applications, for instance, sentiment classification (Chen et al., 2012), cross language text classification (Zhou et al., 2014a), visual object recognition (Ding et al., 2015), face recognition (Wang et al., 2015), and WI-FI location (Gönen and Margolin, 2014). In sentiment analysis, reviewers have different words preference when writing comments to different products. For instance, ‘comprehensive’ is widely used in book product, and ‘noisy’ appears more frequently in electronic products. The book products and electronic products are two domains, and thus have different words distribution. A more complicated case is cross language sentiment analysis where

two domains are reviews written by two different languages. In object or face recognition, different domains include images taken from different directions or angles. For instance, (Long et al., 2013b) contains images from two domains, COIL1 and COIL2. COIL1 are images taken in the directions of $[0^\circ, 85^\circ] \cup [180^\circ, 265^\circ]$. COIL2 are images taken in the directions of $[90^\circ, 175^\circ] \cup [270^\circ, 355^\circ]$. In WI-FI location, the WI-FI signal collected in different devices and environments can be regarded as different domains. All these real-world applications can be formulated into domain adaptation problem settings, and efficiently be solved by appropriate domain adaptation methods.

1.1.5 Domain Adaptation Settings and Challenges

There are different criteria to categorize the problem settings of domain adaptation. Herein, we introduce two popular criteria that are widely studied in the literature (Cao et al., 2010; Long et al., 2014a; Zhou et al., 2014a), namely, target label and feature space.

According to the availability of target labels, domain adaptation is categorized into the semi-supervised one and the unsupervised one. In semi-supervised domain adaptation, target domain has a limited number of labeled data. This is a common case in reality since it may not be expensive to collect a few labeled data given a specific task. However, in unsupervised domain adaptation, there is no label information available in the target domain. This scenario happens when we do not have any prior knowledge for a given task but we are free to get access to the unlabeled data related to the task. In general, unsupervised domain adaptation problems are more difficult to tackle as no target label information can be used to guide the target model learning.

Based on the homogeneity of feature spaces, domain adaptation can be categorized into the homogeneous one and the heterogeneous one. In homogeneous domain adaptation, the feature spaces of two domains are the same, but the feature distributions are different. An example on sentiment analysis can be found in Section 1.1.2. Compared to the homogeneous one, heterogeneous domain adaptation is more challenging due to the heterogeneity of the feature spaces. For instance, two sets of sentiment reviews are written by different languages (Reviews on Taobao China are in Chinese, but reviews on Taobao Overseas are in English). Since two languages

are using completely different word lexicons, supplementary information, such as a translator, is usually needed to establish the bridge between the two languages. If such information is unavailable, which is usually the case in reality, heterogeneous domain adaptation is intractable.

These two ways of categorization are not exclusive, but come out simultaneously in state-of-the-art domain adaptation studies. For instance, homogeneous domain adaptation problems are usually assumed to be unsupervised (Pan et al., 2011; Long et al., 2014a). Target label information can be taken as one of the supplementary information to link up two domains in heterogeneous domain adaptation problems (Wang and Mahadevan, 2011). The basic principle is to construct a domain adaptation problem setting, which is neither trivial nor intractable.

In this thesis, we are dedicated to dealing with two main problem settings, namely (1) unsupervised homogeneous domain adaptation and (2) heterogeneous domain adaptation. Problem (1) is also called as “sample selection bias” in the literature. A basic assumption made in problem (1) is that two domains have the same conditional distribution but different marginal distributions. This assumption is taken as one of the most widely used assumptions in domain adaptation, called “covariant shift”. To solve this problem, the main issue is to align the marginal distributions of two domains. Problem (2) is more difficult to tackle compared to problem (1) as the feature spaces of two domains are heterogeneous. Except for the distribution discrepancy issue, the heterogeneity of feature spaces makes adaptation tasks even harder. Before the distribution match of two domains, the alignment of the feature spaces must be done. This is usually achieved by exploiting some supplementary information, for example, translator, target label, or correspondence instance, in the existing works. However, in this thesis, we try to solve this problem without using any supplementary information. The key idea is to utilize some common grounds shared by two domains to establish the link between different feature spaces.

Note that what we have discussed above strictly follows the domain adaptation scenario where the tasks of two domains are the same. However, this is not always true in many real-world applications. As stated in (Margolis, 2011), domain adaptation problems also include the case where two domains may share the same label space, but have different conditional distributions. In this case, the conditional distribution discrepancy of different domains should also be aligned. In this thesis,

we take this harder problem setting into account. More specifically, we develop an adaptive adaptation method for a multiple source adaptation regression problem.

1.2 Main Contributions

In this thesis, we aim to develop effective algorithms for various domain adaptation scenarios including unsupervised homogeneous and heterogeneous domain adaptation problems, and a more difficult problem setting that assumes a conditional distribution mismatch between domains.

- **Studies on Marginal Stacked Denoising Autoencoder for Unsupervised Domain Adaptation**

In this pillar, we conduct comprehensive studies on a well-known deep feature learning method, marginalized stacked denoising autoencoder (mSDA), for unsupervised homogeneous domain adaptation problems. mSDA has been empirically shown a great success in some real-world adaptation tasks. In this work, we investigate the rationale for why mSDA can benefit adaptation tasks from the perspective of adaptive regularization, specifically on two types of feature corruption noise: *Gaussian* noise (mSDA_g) and *Bernoulli dropout* noise (mSDA_{bd}). Then, we propose a new mSDA with data-dependent *Multinomial dropout* noise (mSDA_{md}), which overcomes limitations of mSDA_{bd} and further improves adaptation performance. Our mSDA_{md} is based on a more realistic assumption: different features are correlated and should be corrupted with different probabilities. Finally, we propose a deep nonlinear feature coding (DNFC) framework to overcome limitations on the objective of mSDA. On the one hand, we employ maximum mean discrepancy (MMD) to minimize the feature mismatch in each layer. On the other hand, we introduce a nonlinear feature mapping induced by kernel methods to dig out the more complex relationships inside the data across domains. By utilizing a deep structure framework to learn deep feature codings jointly reducing feature mismatch and exploring nonlinearity, we manage to learn a set of discriminative and comprehensive cross-domain features from coarse to fine. The generalizations of both mSDA_{bd} and mSDA_{md} to DNFC are investigated.

- **Transfer Local Multiple Manifolds Information for Unsupervised Domain Adaptation**

In this pillar, we develop a subspace-based method for unsupervised domain adaptation. Different from the deep feature learning method that aims at discovering shared knowledge by exploring deep underlying data structures, subspace-based method pays more attention to search a latent subspace in which the source and target domains have similar distributions. Specifically, we introduce a multiple manifolds assumption to unsupervised domain adaptation problems. Under this assumption, we propose to preserve local multiple manifolds information in adaptation. To do so, we develop a manifold neighborhood preservation embedding algorithm to preserve the neighborhood structure in each low-dimensional manifold. To further reduce the distribution discrepancies of different domains, we use MMD as the metric to measure the distribution discrepancy and minimize it in adaptation. A unified framework combining the local manifolds information preservation and the global distribution alignment is then proposed.

- **A General Domain Specific Feature Transfer Framework for Hybrid Domain Adaptation**

In this pillar, a new problem setting called hybrid domain adaptation is investigated. It is a special case of heterogeneous domain adaptation in which different domains share some common features, but also possess their own domain specific features. In this case, it can be efficiently solved without any supplementary information by taking advantage of common features to link up domain specific features in adaptation. We leverage upon common features instead of supplementary information to achieve effective adaptation. We propose a general domain specific feature transfer (DSFT) framework, which can link up different domains using common features and simultaneously reduce domain divergences. Specifically, we learn translations between common features and domain specific features. Then we cross-use the learned translations to transfer the domain specific features of one domain to another domain. Finally, we compose a homogeneous space in which the domain divergence of different domains are minimized. We instantiate the general framework DSFT

to a linear case and a nonlinear case.

- **Source-Target Similarity Modelings for Multi-Source Adaptation Problems**

In this pillar, we are dealing with a difficult problem setting, where the source and target tasks are different. More generally, we are handling a multi-source adaptation regression problem. Target domain is assumed to have few labeled data, but plentiful labeled data from multiple source domains are available. The key challenge in multi-source domain adaptation is to capture the diverse inter-domain similarities. In this work, we study different approaches based on Gaussian process models to solve the multi-source adaptation regression problem. Precisely, we investigate the feasibility and performance of a family of transfer covariance functions that represent the pairwise similarity of each source and the target domain. We theoretically show that using such a transfer covariance function for general Gaussian process modelling can only capture the same similarity coefficient for all the sources, and thus may result in unsatisfactory adaptation performance. This leads us to propose $TC_{MS}Stack$, an integrated strategy incorporating the benefits of transfer covariance function and stacking.

1.3 Thesis Outline

The thesis aims to summarize related works on domain adaptation, present novel methods for homogeneous and heterogeneous domain adaptation problems, and discuss future works. It is organized as below:

- **Chapter 2 - Background and Related Works**

In this chapter, we discuss state-of-the-art domain adaptation methods from three perspectives: instance-based domain adaptation, parameter-based domain adaptation, and feature-based domain adaptation. We also introduce the metrics that are widely used in domain adaptation to measure domain divergence.

- **Chapter 3 - Feature Analysis of mSDA for Domain Adaptation**

Figure 1.2: The Organization of Thesis

This chapter analyzes the underlying rationale of mSDA on why it is effective for domain adaptation tasks. A new variant of mSDA is then introduced to further improve adaptation performance. Extensive experiments are reported after technical details.

- **Chapter 4 - Deep Nonlinear Feature Coding (DNFC)**

This chapter presents details of a proposed framework: deep nonlinear feature coding (DNFC). We identify some gaps on the objective of mSDA, and then state technical details of DNFC that fills up the gaps. Extensive experiments are reported after technical details.

- **Chapter 5 - Local Manifolds Information Transfer (LMIT)**

This chapter proposes a new subspace-based method not only aligning global distributions of domains, but also preserving local data multiple manifolds information. A multiple manifolds assumption is introduced to unsupervised domain adaptation, and a manifold neighborhood preservation embedding (M-NPE) algorithm is developed. An LMIT framework is also proposed. Extensive experiments are reported after technical details.

- **Chapter 6 - Domain Specific Feature Transfer (DSFT)**

This chapter presents details of the proposed framework: domain specific feature transfer for hybrid domain adaptation. We first discuss the gaps of existing heterogeneous domain adaptation methods. Then, we propose a new problem setting, called hybrid domain adaptation. A novel framework, DSFT, is presented afterwards for the proposed new problem setting. Extensive experiments are done to show the effectiveness of the proposed DSFT.

- **Chapter 7 - Transfer Covariance Function Stacking**

This chapter describes our proposed method $TC_{MS}Stack$. We demonstrate the pathology of a family of transfer covariance functions for multi-source adaptation regression problem, and then present our $TC_{MS}Stack$ which integrates transfer covariance function with stacking. Extensive experiments are reported after technical details.

- **Chapter 8 - Conclusions and Future Work**

This chapter concludes this thesis and states future research topics.

Chapter 2

Background and Related Works

Domain adaptation has been widely studied in the last decade. In this chapter, we conduct comprehensive reviews on existing state-of-the-art domain adaptation methods. We follow the categorization in terms of what to transfer in the survey paper (Pan and Yang, 2010). Specifically, we discuss instance-based domain adaptation methods, parameter-based domain adaptation methods, and feature-based domain adaptation methods. Instance-based domain adaptation methods assume that only certain part of source data is good for the model learning in target domain, and the rest source data may not be helpful. Thus, to achieve effective adaptation, efforts are paid to identify those helpful source instances and enhance their effects in adaptation. This is usually done by reweighting source instances. Parameter-based domain adaptation methods assume that the source and target domains possess similar model parameters or share the same priors. Knowledge is transferred across domains through the shared parameters or priors. Feature-based domain adaptation methods assume that there is a latent shared feature space in which the source and target domains have similar feature distributions. In the latent shared feature space, the model learned using source mapped data can achieve a satisfactory performance in target tasks. Compared to the instance-based and parameter-based domain adaptation methods, feature-based domain adaptation methods work on the alignment of feature distributions directly. A distribution discrepancy metric that is used to measure the domain divergence between different domains is usually adopted. The minimization of the distribution discrepancy metric is an indispensable objective in loss functions of various feature-based domain adaptation methods. More specifically, feature-based domain adaptation can be categorized into homogeneous domain

adaptation and heterogeneous domain adaptation (Yang et al., 2009) based on the homogeneity of the source and target feature spaces. In homogeneous domain adaptation, source and target feature spaces are the same, but feature distributions are different. The main objective of homogeneous domain adaptation methods is to align the feature distributions of two domains. Compared to the homogeneous one, heterogeneous domain adaptation is more difficult to tackle due to the heterogeneity of the source and target feature spaces. Since there is no correspondence between the feature spaces of two domains, supplementary information, like label, translator, or correspondence instances, is exploited to align the feature spaces before distributions match.

In this chapter, we discuss existing methods on these three kinds of domain adaptation strategy, and pay more attention to the feature-based one as it is becoming more and more popular in recent years.

2.1 Instance-based Domain Adaptation

A key assumption in instance-based domain adaptation is that there are both similarities and dissimilarities between source and target instances. An effective instance-based method should successfully enhance the effect of those source instances that are similar to target instances in adaptation, and simultaneously discard those dissimilar ones.

2.1.1 Problem Definition

In principle, instance-based domain adaptation methods attempt to reweight source instances so that the effect of those less target-related source instances is reduced and the effect of those more target-related source instances is boosted. Most of instance-based domain adaptation methods follow an assumption, called covariate shift (Adams, 2010). It assumes that marginal distributions of source and target domains are different, but the conditional distributions are the same, that is, $p_s(x) \neq p_t(x)$ but $p_s(y|x) = p_t(y|x)$. Under such an assumption, the aim is to find the optimal θ^* that can maximize the expected log likelihood on the target domain given a model family:

$$\theta^* = \arg \min_{\theta \in \Theta} E_{(x,y) \in p_t(x,y)} [l(x, y, \theta)],$$

where $l(x, y, \theta)$ is a loss function. Considering the assumption of covariate shift, $p_t(x, y) = p_t(x)p_t(y|x) = \frac{p_t(x)}{p_s(x)}p_s(x, y)$, we have:

$$\theta^* = \arg \min_{\theta \in \Theta} E_{(x,y) \in p_s(x,y)} \frac{p_t(x)}{p_s(x)} [l(x, y, \theta)]. \quad (2.1)$$

From Eq. (2.1), we can see that by assigning different weights to source instances, we can train a precise model for the target domain. Thus, the key point is to estimate the weight $\frac{p_t(x)}{p_s(x)}$.

2.1.2 Weights Estimation Methods

Various works have been done to estimate $\frac{p_t(x)}{p_s(x)}$. One of the most effective ideas is based on distribution alignments. The objective is to make the distribution of weighted source instances as close as the distribution of target instances. In (Huang et al., 2006), a novel kernel mean matching (KMM) method is proposed to estimate the weights. Maximum mean discrepancy, MMD, is used to measure the distribution discrepancy between two domains (Borgwardt et al., 2006). More specifically, KMM can be formulated into a quadratic program with respect to the weights:

$$\begin{aligned} \min_{\beta} \quad & \frac{1}{2} \beta^T \mathbf{K} \beta - \kappa^T \beta, \\ \text{s.t.} \quad & \beta_i \in [0, B] \text{ and } \left| \sum_{i=1}^{n_S} \beta_i - n_S \right| \leq n_S \varepsilon, \end{aligned}$$

where

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{SS} & \mathbf{K}_{ST} \\ \mathbf{K}_{TS} & \mathbf{K}_{TT} \end{bmatrix},$$

and $\beta_i = \frac{P(x_{S_i})}{P(x_{T_i})}$ proved by (Huang et al., 2006). The constraints avoid a very large weight and require the sum of all the weights to be close to 1. A similar idea, called ‘‘Kullback-Leibler Importance Estimation Procedure’’ (KLIEP) is adopted in (Sugiyama et al., 2008) and (Yuta et al., 2009). Instead of using MMD, another distribution discrepancy metric, Kullback-Leibler divergence $KL(p_t(x)||w(x)p_s(x))$, is used to measure the distribution discrepancy in KLIEP. Compared KMM to KLIEP, the former is much more desired as it avoids the density estimation of source and target distributions that is extremely challenging when data size is small.

Another idea of weights estimation is introducing an indicator variable to instances. In (Zadrozny, 2004), a binary random variable $s \in \{0, 1\}$ is used to determine whether or not a source domain sample is selected into the training set. As proved by the authors, the weights to be estimated are inversely proportional to selection probabilities $p(s = 1|x)$. In experimental studies, they assume the selection probabilities are known with samples (x, s) provided. This is a strong assumption as it only happens when source instances are actually a subset of target instances so that one can have knowledge for data x about whether it is selected or not. (Bickel et al., 2007) also defines a random indicator variable δ for weights estimation. However, this indicator variable δ is used to determine whether an instance is from source domain or target domain. This makes the value of δ_i easily be obtained for instances in both source and target domains. The weight function $\omega(x)$ can be written in terms of the ratio $\frac{p(\delta=0|x)}{p(\delta=1|x)}$, and a logistic regression model $p(\delta|x; \boldsymbol{\theta}, \boldsymbol{\Omega})$, where $\boldsymbol{\theta}$ is model parameters and $\boldsymbol{\Omega}$ corresponds to the weights, is then trained to predict the domain where x is from. Rather than learning $\boldsymbol{\Omega}$ first and then training a prediction model $p(y|x, \boldsymbol{\theta})$ with the learned $\boldsymbol{\Omega}$, they propose to learn them jointly in one objective function:

$$\sum_{(x_t, y_t) \in D_S} \omega(x; \boldsymbol{\Omega}) \log(p(y_i|x_i; \boldsymbol{\theta})) + \sum_{x_t \in D_s, D_t} \log(p(\delta_i|x_i; \boldsymbol{\Omega})) - \eta \|\boldsymbol{\Omega}\|^2 - \gamma \|\boldsymbol{\theta}\|^2.$$

An extended work (Bickel and Scheffer, 2007) applies this idea to the task of spam classification. However, (Bickel and Scheffer, 2007) assumes multiple inboxes, which represent different target domains, and develops a model relating the prior distributions over $\boldsymbol{\Omega}$ in different inboxes.

Cluster-based methods are also popular for weights estimation. Estimating the weights according to the empirical frequencies of observed instances is a straightforward way to calculate $\frac{p_t(x)}{p_s(x)}$. Based on this intuition, (Cortes et al., 2008) conducts data clustering first, and then estimates weights for the instances in each cluster. However, a simple and straightforward data clustering may result in unbalance data distributions in each cluster, which makes weights estimation inaccurate. To tackle this shortcoming, (Ren et al., 2008) proposes to balance the distributions across clusters by adaptively selecting training instances.

2.1.3 Other Methods

Except for weights estimation methods, there are also other strategies transferring instances across domains. One of the most popular methods is TrAdaBoost (Dai et al., 2007b), which is a boosting method based on Adaboost algorithm. In TrAdaBoost, the authors assume that some of source instances are good for adaptation, but the others are harmful. Thus, the objective of TrAdaBoost is iteratively reweighting source instances to enhance the importance of those helpful ones and alleviate the effect of those harmful ones. In each iteration, TrAdaBoost utilizes the weighted instances to train a classifier, and then calculate prediction errors on target instances. The calculated errors are used to update the weights for both source and target instances. In TrAdaBoost, those harmful source instances are assigned with very low weights, but still take effect in the model training. In (Jiang and Zhai, 2007), a heuristic method is applied to remove those harmful source instances directly based on the difference between conditional probabilities $p_s(y|x)$ and $p_t(y|x)$. Other methods include (Liao et al., 2005) that takes advantage of the idea of active learning to label target domain instances with the help of source domain instance.

2.2 Parameter-based Domain Adaptation

Parameter-based domain adaptation methods assume that the source and target models share some model parameters or prior distributions of parameters. Although intuitively appealing, there are several limitations for this domain adaptation strategy. Firstly, the application scenario of parameter-based domain adaptation method is very limited as it usually requires both the source and target domains have sufficient knowledge in the model learning. To train a model in the target domain, it needs target labeled data. If target labeled data is unavailable, parameter-based domain adaptation methods are not applicable. If target label data is plentiful, there is no need to do transfer. Only when target labeled data is scarce, parameter-based domain adaptation methods are potential to work. However, the model learning process is using the source and target data together, which makes the learned target parameter easily biased to the source domain since source data is dominated in the learning process. Secondly, parameter-based domain adaptation methods need to face the problem of model hyper-parameter selections. However, as stated in (Li

et al., 2014; Zhong et al., 2010; Sugiyama et al., 2007), under domain adaptation scenarios, standard model hyper-parameter selection techniques, such as cross validation (CV), does not work as desired since the unbiasedness that guarantees the accuracy of CV does not hold in domain adaptation tasks. Thirdly, parameter-based domain adaptation methods are closely related to the model used for adaptation, but no theoretical analysis gives the evidence on what model is more suitable for transferring parameters. In the following sections, we discuss parameter-based domain adaptation methods in terms of the model they built on.

2.2.1 Maximum Entropy Model

Entropy maximization is learning a model to maximize the conditional likelihood of the training data (Berger et al., 1996), given a set of training examples $\{x_i, y_i\}_{i=1}^n$ with feature sets $F = \{f_1, \dots, f_F\}$:

$$p(y_i|x_i) = \frac{1}{Z(x_i)} \exp\left(\sum_{j=1}^F f_j(x_i)\omega_j\right),$$

where $\boldsymbol{\omega} = (\omega_1, \dots, \omega_F)$ represent model parameters, and Z is a normalization term:

$$Z(x_i) = \sum_{y \in Y} \exp\left(\sum_{j=1}^F f_j(x_i)\omega_j\right).$$

Thus, the final optimization problem is as follows:

$$\arg \max \sum_{i=1}^N \log p(y_i|x_i) - \beta \|\boldsymbol{\omega}\|^2.$$

Based on the conventional maximum entropy model, (Chelba and Acero, 2006) proposes a transferred-version by modifying the regularization term. At the first step, a model is trained using source data only, denoted as $\boldsymbol{\omega}_{src}$. Then, a target model is obtained by using the limited target labeled data. However, during the training, instead of regularizing target model parameters using l_2 norm only, a new transfer norm is taken into account to enforce $\boldsymbol{\omega}_{tar}$ to be close to the learned $\boldsymbol{\omega}_{src}$. Correspondingly, the advanced optimization problem becomes:

$$\arg \max \sum_{i=1}^N \log p_{tar}(y_i|x_i) - \beta \|\boldsymbol{\omega}_{src} - \boldsymbol{\omega}_{tar}\|^2.$$

This method actually takes source model parameters as the prior of target model parameters. However, since target data are scarce while source data are much more sufficient, the learned parameters are usually biased to the source parameters.

2.2.2 Support Vector Machine

Support vector machine (SVM) is one of the most widely used model in conventional machine learning. In domain adaptation problems, SVM is also a popular model that attracts many attentions. A straightforward way to adapt SVM to domain adaptation is applying the above idea in entropy model to SVM parameters learning. Next, we mainly focus on other ideas of applying SVM to domain adaptation.

In (Joachims, 1999), the authors propose a transductive support vector machine (TSVM) to iteratively fine-tune source hyperplane to better fit target domain data. Specifically, a hyperplane on source domain is learned first. The learned model is then used to classify the unlabeled target data, and margin values are taken as the confidence measure of predictions. The most confident target data are pseudo-labeled, and incorporated into next phase of training. The intuition behind is to iteratively tune the source hyperplane until it achieves satisfactory results on most of testing target data.

Evgenious and Pontil (Evgeniou and Pontil, 2004) propose a novel idea that assumes model parameters consist of two parts: a common part and a domain specific part. Specifically, we have:

$$\boldsymbol{\omega}_S = \boldsymbol{\omega}_0 + \mathbf{v}_S, \quad \boldsymbol{\omega}_T = \boldsymbol{\omega}_0 + \mathbf{v}_T,$$

where $\boldsymbol{\omega}_0$ are the shared common weights of the two domains, and $\mathbf{v}_S$ and $\mathbf{v}_T$ are the domain specific weights for the two domains. The learning objective is formulated as:

$$\begin{aligned} & \min_{\boldsymbol{\omega}_0, \mathbf{v}_t, \xi_{it}} J(\boldsymbol{\omega}_0, \mathbf{v}_t, \xi_{it}) \\ &= \sum_{t \in \{S, T\}} \sum_{i=1}^{n_t} \xi_{it} + \frac{\lambda_1}{2} \sum_{t \in \{S, T\}} \|\mathbf{v}_t\|^2 + \lambda_2 \|\boldsymbol{\omega}_0\|^2 \\ & s.t. \quad y_{t_i}(\boldsymbol{\omega}_0 + \mathbf{v}_t)\mathbf{x}_{t_i} \geq 1 - \xi_{it}, \\ & \quad \xi_{t_i} \geq 0, \quad i \in \{1, 2, \dots, n_t\} \text{ and } t \in \{S, T\}. \end{aligned}$$

Those three parameters $\boldsymbol{\omega}_0$, $\mathbf{v}_S$, and $\mathbf{v}_T$ can be obtained jointly by optimizing the above problem.

2.2.3 Gaussian Process Model

Maximum entropy model and SVM are used to solve classification problems in domain adaptation scenario, while gaussian process (GP) model is widely used to handle regression problems.

An effective domain adaptation method related to GP is adaptive transfer gaussian process (Cao et al., 2010). The key idea is to construct a transfer covariance function modeling the similarity between domains. In the transfer covariance function, one similarity coefficient is assigned to model the inter-domain similarity between different domains. Specifically, a random variable is introduced to reweight the similarity of the source and target data. The transfer covariance function is constructed as:

$$k_{\lambda}(\mathbf{x}, \mathbf{x}') = \begin{cases} \lambda k(\mathbf{x}, \mathbf{x}'), & \mathbf{x} \in \mathbf{X}^S \text{ \& } \mathbf{x}' \in \mathbf{X}^T \\ & \text{or } \mathbf{x} \in \mathbf{X}^T \text{ \& } \mathbf{x}' \in \mathbf{X}^S, \\ k(\mathbf{x}, \mathbf{x}'), & \text{otherwise,} \end{cases}$$

where λ is the random variable measuring the similarity of source and target domains, and $k(\cdot, \cdot)$ is any valid covariance function. In Cao et al. (2010), the following theorem is given to guarantee valid GP models using $k_{\lambda}(\mathbf{x}, \mathbf{x}')$.

Theorem 2.1. *Let*

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{SS} & \mathbf{K}_{ST} \\ \mathbf{K}_{TS} & \mathbf{K}_{TT} \end{bmatrix}$$

be a PSD matrix with $\mathbf{K}_{ST} = \mathbf{K}_{TS}^T$. Then for $|\lambda| \leq 1$,

$$\mathbf{K}_{\lambda} = \begin{bmatrix} \mathbf{K}_{SS} & \lambda \mathbf{K}_{ST} \\ \lambda \mathbf{K}_{TS} & \mathbf{K}_{TT} \end{bmatrix}$$

is also a PSD matrix.

More methods using GP model are proposed for multi-task learning, for instance (Lawrence and Platt, 2004; Schwaighofer et al., 2004; Bonilla et al., 2007). (Lawrence and Platt, 2004) proposes to learn gaussian parameters over multi-tasks by assuming multiple tasks share the same gaussian prior distributions. In (Bonilla et al., 2007), the authors propose to utilize an intrinsic correlation model to learn the inter-relationships between different tasks. (Schwaighofer et al., 2004) combines the Bayesian framework with GP model for multi-task learning.

2.3 Feature-based Domain Adaptation

In feature-based domain adaptation, the aim is to learn latent feature spaces in which the domain divergence of two domains is minimized. It can be categorized into the homogeneous one and the heterogeneous one according to the structure of feature spaces. In the following sections, we discuss various state-of-the-art methods used to solve these two problem settings.

2.3.1 Homogeneous Domain Adaptation

In homogeneous domain adaptation, the source and target feature spaces are the same. The main objective is to align the feature distributions of two domains. Subspace-based method and deep feature learning method are two popular strategies. They share the same underlying intuition that aims to discover good feature representations either by searching latent shared feature subspaces or by extracting deep high-level features. In the following sections, we first give a theoretical instruction on how to learn a good feature representation that benefits domain adaptation tasks, and then introduce state-of-the-art subspace-based methods and deep feature learning methods.

2.3.1.1 Theoretical Foundation

In (Ben-David et al., 2007; Blitzer et al., 2008), the authors give a theoretical analysis on how a feature representation enables effective domain adaptation. The main outcome is a uniform convergence learning bound for domain adaptation methods that aim to minimize the convex combinations of source and target empirical risks. Specifically, a target risk bound is derived by using the source risk and the distance between distributions of source and target domains. The theorem related to the target risk bound is given below:

Theorem 2.2. *Let $\mathcal{H}$ be a hypothesis space of VC-dimension d and $\mathcal{U}_S, \mathcal{U}_T$ be unlabeled samples of size m each, draw from $\mathcal{D}_S$ and $\mathcal{D}_T$, respectively. Let dis be the empirical distance on $\mathcal{U}_S, \mathcal{U}_T$, induced by the symmetric difference hypothesis space. With probability at least $1 - \delta$ (over the choice of the samples), for every $h \in \mathcal{H}$,*

$$\varepsilon_T(h) \leq \varepsilon_S(h) + \frac{1}{2}dis(\mathcal{U}_S, \mathcal{U}_T) + 4\sqrt{\frac{2d\log(2m) + \log(4/\delta)}{m}} + \lambda$$

where λ is the combined risk of the ideal hypothesis and $\text{dis}(\mathcal{U}_S, \mathcal{U}_T)$ is the domain divergence of two domains.

When λ is large, it implies that there is no model that performs well on both the source and target domains. In this case, one cannot find a good target hypothesis by training only on the source domain data. On the other hand, if λ is small (which is usually a basic assumption in domain adaptation, that is, the source and target domains are closely related), the above theorem guides us that a good feature representation is the one that can reduce source training error and domain divergence simultaneously.

2.3.1.2 Domain Divergence Metric

As shown in the above theorem, domain divergence is one of the most important factors that can enable a small target risk bound. In this section, we introduce three metrics that are widely used in domain adaptation tasks to measure the domain divergence of different domains, namely, maximum mean discrepancy, KL-divergence, and A-distance.

(1) Maximum Mean Discrepancy

Maximum mean discrepancy is widely used to measure the discrepancy of two distributions in domain adaptation (Pan et al., 2011). It avoids density estimations and uses the mean embeddings of two distributions in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ as distance calculation. Given two sets of samples $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_{n_1}\}$ and $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_{n_2}\}$, drawn from the distributions P and Q respectively, the empirical MMD is calculated as:

$$\left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \Phi(\mathbf{x}_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \Phi(\mathbf{y}_i) \right\|_{\mathcal{H}},$$

where $\Phi : \mathbb{R} \rightarrow \mathcal{H}$ with $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ as the kernel of $\mathcal{H}$.

(2) KL-divergence

KL-divergence (Kullback and Leibler, 1951), also known as relative entropy, is another way to quantify the domain divergence of different domains in domain adaptation (Yuta et al., 2009; Wan et al., 2011; Yosinski et al., 2014). It is a parametric metric. Given two probability distributions $\mathbf{P}$ and $\mathbf{Q}$, KL-divergence of $\mathbf{Q}$ from $\mathbf{P}$ is defined as:

$$D_{KL}(\mathbf{P}||\mathbf{Q}) = \sum_{i=1}^k \mathbf{P}(i) \ln\left(\frac{\mathbf{P}(i)}{\mathbf{Q}(i)}\right).$$

Note that the above definition is for discrete distributions. Regarding the continuous density functions, the above summation operation would become to integration. From the definition of KL-divergence, we can see that the calculation is done on probability distributions directly. Thus, one may need to come up with ways to estimate the data probability distributions first. Moreover, KL-divergence is a non-symmetric metric. A symmetrized version as shown in follows is more widely used in domain adaptation.

$$KL(\mathbf{P}, \mathbf{Q}) = D_{KL}(\mathbf{P}||\mathbf{Q}) + D_{KL}(\mathbf{Q}||\mathbf{P}).$$

(3) A-distance

A-distance is proved in (Ben-David et al., 2007; Blitzer et al., 2008) to be an effective metric to measure the domain divergence. It has been frequently adopted in (Ganin and Lempitsky, 2014; Ajakan et al., 2014; Gani et al., 2015). A-distance is approximated as:

$$A_{dis} = 2(1 - 2\epsilon)$$

where ϵ is the generalization error on the problem of discriminating between source and target samples.

2.3.1.3 Subspace-based Methods

Based on the above theorem, various subspace-based methods have been proposed for homogeneous domain adaptation. The main effort is put into the search of a common latent subspace in which the domain divergence of two domains is minimized. The basic idea is shown in Figure 2.1.

MMD-based method is one of the most popular subspace-based methods. The basic idea is utilizing MMD as the domain divergence metric to guide the learning of latent subspaces. In (Pan et al., 2008, 2011), MMD is investigated to find a common latent subspace through PCA-based dimensionality reduction method. In (Pan et al., 2008), the authors propose a dimensionality reduction method to reduce domain divergence and preserve data variance. However, this method suffers from several drawbacks: 1) it cannot generalize to the unseen data; 2) it needs to solve an expensive semi-definite positive (SDP) problem. To overcome those shortcomings, (Pan et al., 2011) proposes a transfer component analysis (TCA) algorithm to learn the shared features in a reproducing kernel Hilbert space (RKHS). (Chen et al.,

Figure 2.1: Basic idea of subspace-based methods

2009) expands the idea in (Pan et al., 2008) by introducing a discriminative analysis term to the objective function so that the latent subspace and model parameters are learned jointly. Dissimilar to TCA that maps original data to a RKHS first and then formulates a mapping in the RKHS to reduce domain divergence, domain invariant projection (DIP) (Baktashmotlagh et al., 2013) conducts a mapping in the original feature space first, and then reduces domain divergence in a RKHS. However, both TCA and DIP predefine a kernel function to facilitate the mapping from original feature space to RKHS. In (Long et al., 2015), a transfer kernel learning (TKL) method is developed to match the source and target distributions in RKHS by learning a domain invariant kernel.

Advanced MMD-based methods combine MMD with some other techniques to further boost adaptation. (Gong et al., 2013) proposes to utilize MMD to discover landmarks to construct an easier auxiliary domain adaptation task. Those landmarks are subsets of source data which distribute similarly with target data. This leads to a better generalization performance in target domain. Transfer joint matching (TJM) (Long et al., 2014a) reduces domain divergence by jointly minimizing MMD and reweighting instances. Instance reweighting is done by adding a l_{21} norm to the learned mapping. Using l_{21} norm, both (Gu et al., 2011) and (Wang et al., 2016) conduct joint subspace learning and feature selection for domain adaptation. Further study (Li et al., 2016a) extends (Gu et al., 2011) by taking data structure preservation into account. In (Wang et al., 2015), a transfer feature rep-

resentation method is proposed by combining MMD and multiple kernel learning methods. It learns multiple kernels as well as a single mapping to minimize the domain divergence and preserve the data discriminative power. (Wang et al., 2014a) applies information theories to metric learning. An objective function satisfying the following properties is proposed. Firstly, it reduces the domain divergence through MMD. Secondly, it preserves the geometry of target data. Lastly, it aligns source data geometry with source label information. (Long et al., 2013b) puts forward a joint distribution adaptation method to align both marginal distributions and conditional distributions of two domains simultaneously.

Except for MMD-based methods, other subspace-based methods include: correspondence learning methods, manifold-based methods and methods that consider various data properties. For correspondence learning methods, (Blitzer et al., 2006, 2007, 2011) are proposed to seek the shared representation through correspondence learning. The efficiency of these methods depends on the selection of pivot features. These pivot features are the key bridge to link up two domains. In (Blitzer et al., 2006), a structural correspondence learning (SCL) method extracts the most frequently appeared features as pivot features. In (Blitzer et al., 2007), it applies SCL to multi-domain sentiment classification problems, and it also improves the selection of pivot features by using mutual information between features. A more recent work (Blitzer et al., 2011) formulates a general low-dimensional coupled subspace shared by the source and target domains so that pivot features can be automatically decided. Regarding manifold-based methods, (Gopalan et al., 2011; Gong et al., 2012; Gopalan et al., 2014) propose to generate intermediate subspaces along the geodesic path between source and target subspaces on the Grassmann manifold. These methods need to compute a large number of intermediate subspaces, which is very time-consuming and computation-expensive. To alleviate the computational issue, Fernando et al. propose a subspace alignment method aligning source and target subspaces directly (Fernando et al., 2013). Other methods also consider important data properties in adaptation, like data sparsity (Long et al., 2013a; Tsai et al., 2016), second-order statistics (Sun et al., 2016), geometric structure (Shu et al., 2014; Long et al., 2014b) or low rank property (Shao et al., 2012, 2014). For instance, (Long et al., 2013b) learns a transfer sparse coding, and simultaneously reduces distribution discrepancies of two domains. (Sun et al., 2016) minimizes the

distribution discrepancies by aligning the second-order statistics of different domains. Another subspace-based method is (Daume III, 2009) that constructs a universal feature space by replicating features and filling zeros. This method is frustratingly easy but very effective in text categorization tasks.

In summary, the essence of all the subspace-based methods is to learn latent subspaces with the objective of minimizing domain divergence and adding some other constraints, for instance, minimizing data variance, conducting discriminative analysis, or preserving data geometric structure etc. As demonstrated in various studies, subspace-based methods have achieved very promising performances on many real-world applications.

2.3.1.4 Deep Feature Learning Methods

Recently, studies on the learning of high-level deep features to mine better and richer feature representations attract increasing interests in domain adaptation. The major motivation of deep feature learning to domain adaptation is that deep features are able to disentangle complex and high-level information underlying data, making final feature representations more discriminative and comprehensive to domain adaptation.

Autoencoder is one of the most widely used deep feature learning techniques in domain adaptation. (Glorot et al., 2011) investigates a deep learning approach for large-scale sentiment classification based on stacked denoising autoencoder (SDA). SDA learns multiple-layer features by reconstructing the input data that are artificially corrupted with random noise. It achieves excellent adaptation performance on multi-domain sentiment classification tasks. However, it suffers from two crucial limitations: high computational cost and low generalization ability to high-dimensional features. To tackle these limitations, (Chen et al., 2012) proposes a marginalized stacked denoising autoencoder (mSDA) method that marginalizes out random feature corruptions. mSDA learns a reconstructive mapping matrix by taking advantage of matrix expectations so that there is no need to explicitly construct a single corruption. Following works extend mSDA to further boost adaptation. In (Ding et al., 2015), mSDA is taken as one of the building blocks in a deep low-rank coding (DLRC) framework. It is used to guarantee the learned low rank feature coding to be robust to noise. Another work, transfer learning with deep autoencoders

(TLDA) (Zhuang et al., 2015a), learns deep feature representations by jointly minimizing domain divergence explicitly and encoding label information. It consists of two layers, namely embedding layer to enhance domain similarity, and label encoding layer to fully exploit label information. (Zhuang et al., 2015b) extends TLDA to multi-task learning. There are also works extending mSDA by forming different regularization terms. In (Clinchant et al., 2016), the authors propose a regularization on the learned mapping for mSDA. Such a regularization term is used to enforce learned feature representations not only to be domain-invariant but also to be as close as possible to target domain distribution. Instead of adding constraints on the feature mapping, (Jiang et al., 2016) uses l_{21} norm as a measure of reconstruction error of mSDA since l_{21} norm is much more robust to outliers.

Another branch of deep feature learning methods for domain adaptation is based on conventional neural network (CNN). (Yosinski et al., 2014) conducts empirical analyses on the transfer ability of CNN deep features in each layer. It demonstrates a dynamic variation from generality to specificity on the transferability of CNN features from the bottom to the top of a neural network. Based on the conclusion in (Yosinski et al., 2014), (Long and Wang, 2015) develops a deep adaptation network (DAN) to learn transferable features. Specifically, the bottom three layer features are considered as general. Layer 4 and layer 5 are slightly transferable, and thus fine-tuning is applied. The last three layers are considered as task-specific. To make them transferable, multiple kernelized MMD is adopted to reduce domain discrepancies. Different with DAN, DTN (deep transfer network) (Zhang et al., 2015) minimizes marginal distributions of two domains in each layer using MMD. More importantly, it also minimizes conditional distributions in the last discriminative layer by using pseudo target labels iteratively. Another work (Hu et al., 2015) also utilizes MMD to align feature distributions in each layer. However, it maximizes inter-class variations and minimizes intra-class variations at the same time. Instead of using MMD, domain adversarial network (DAN) (Ganin and Lempitsky, 2014; Ajakan et al., 2014; Gani et al., 2015) utilizes A-distance to measure domain divergence. They add a novel gradient reversal layer, which is used to distinguish the source and target samples, to guide weights learning. Many extended neural network structures based on DAN are also proposed. In (Luo et al., 2017), the authors design a framework that utilizes a domain adversarial loss to battle domain shift and then use a metric

learning-based approach to apply the learned embedding to new tasks. (Motiian et al., 2017) provides a adversarial learning framework to learn a subspace that maximizes the confusion of different domains and simultaneously align their semantic embeddings.

Generally speaking, deep feature learning methods are becoming more and more popular since they can extract more meaningful and discriminative features for adaptation tasks. However, many ideas of deep feature learning methods are borrowed from subspace-based methods. This is because subspace-based methods can be taken as one layer of hidden feature learning. Once an objective is well formulated for one layer, it can be extended to deep layers by designing an adequate deep structure.

2.3.2 Heterogeneous Domain Adaptation

Heterogeneous domain adaptation attempts to transfer knowledge across domains with different feature spaces. Compared to the homogeneous one, heterogeneous domain adaptation is more difficult to tackle due to the heterogeneity of feature spaces. Since there is no correspondence between the feature spaces of different domains, supplementary information, like label, translator, or correspondence instances, is needed to establish the bridge of the domains.

Label information is the most widely used supplementary information to link up the feature spaces of different domains. Figure 2.2 shows the basic idea of using labels to connect different domains in heterogeneous domain adaptation. The intuition is to find a latent subspace gathering the data with same labels and separating the data with different labels. Many state-of-the-art heterogeneous domain adaptation methods are based on this idea. (Wang and Mahadevan, 2011) learns a latent subspace through a manifold alignment approach (DAMA). DAMA aims to match the instances with identical labels and separate the instances with different labels. At the same time, the data topology of domain is preserved in the subspace learning. To achieve these goals, DAMA creates three adjacency graphs, namely domain similarity graph, domain dissimilarity graph and data topology graph. The label information plays a key role in constructing these adjacency graphs. Different from DAMA that exploits labels to construct a similarity graph and a dissimilarity graph, heterogeneous transfer discriminant analysis of canonical correlations (HTD-CC) (Wu et al., 2013) is proposed to maximize data inter-class canonical correlations

Figure 2.2: Heterogeneous domain adaptation using labels

and minimize data intra-class canonical correlations using labels. Another method, called heterogeneous feature augmentation method (HFA) (Li et al., 2014), does not construct similarity or dissimilarity matrix explicitly. It takes advantage of kernel methods to exploit the similarity of different domains. Specifically, the proposed HFA is based on SVM, that is trained using labeled data from both domains. It first maps the samples of two domains to a common latent subspace to obtain the shared common features. Next, it augments the samples of each domain with original features and zeros through a feature mapping function. To avoid learning the mapping function explicitly, HFA converts the learning problem to a multiple kernel learning (MKL) formulation. Similar to HFA using kernel methods, (Xiao and Guo, 2015) propose a kernel matching SSKMDA1 that aligns the source and target kernel matrix (submatrix) based on a Hilbert Schmidt Independence Criterion. In (Shi et al., 2010), a three-step framework, HeMap, is proposed for heterogeneous domain adaptation. In the first step, HeMap unifies feature spaces using spectral embedding. The principle is to formulate an objective function that preserves data structure and maximizes the similarities of different domains. In the second step, sample selections are done to select those source samples which are closely related to the target ones. Lastly, a Bayesian-based approach is applied to model the relationship between different domains. In (Kulis et al., 2011), Kulis et.al propose an ARC-t framework that applies the idea of metric learning to heterogeneous domain adaptation, and learns an asymmetric nonlinear transformation matrix across domains. Labels from two domains are utilized to construct both similarity and dissimilarity constraints in ARC-t. Instead of learning transformations on the input

data of different domains, (Zhou et al., 2014b) learns mappings based on the source and target predictive structures. Model parameters, which are learned using the labeled data in each domain, are used to represent the domain predictive structure. Recently, a novel method, CDLS (Hubert Tsai et al., 2016), is proposed to learn cross-domain landmarks for heterogeneous domain adaptation. Instead of viewing all cross-domain data to be equally important in adaptation, CDLS exploits the representative source and target data as landmarks to improve adaptation performance. The selection of landmarks is done by utilizing the labels of the source and target domains.

Other supplementary information include: translator, (Dai et al., 2008), correspondence instances, (Zhu et al., 2011; Zhou et al., 2014a), and previous classifier (Mozafari and Jamzad, 2014). These supplementary information is more difficult to obtain compared to limited number of target labels in reality. Thus, they are less studied than methods using labels in heterogeneous domain adaptation.

In summary, in heterogeneous domain adaptation, the key step is to align the feature spaces of different domains. Various studies pay attention to this step by utilizing auxiliary information as less as possible. Once the heterogeneity of feature spaces is eliminated, a heterogeneous domain adaptation problem can be treated as a homogeneous domain adaptation problem. Then, the ideas in homogeneous domain adaptation can be readily applied.

Chapter 3

Feature Analysis of mSDA for Domain Adaptation

Stacked denoising autoencoder (SDA) (Glorot et al., 2011) is one of the typical deep feature learning methods in domain adaptation. Multi-layer deep features are extracted by reconstructing input data from the artificially corrupted ones contaminated by random noises. In (Chen et al., 2012), a marginalized stacked denoising autoencoder (mSDA) is proposed to further improve adaptation performance. By averaging over an infinite number of corruptions, robust features can be extracted efficiently without materializing any data corruptions. Figure 3.1 shows the framework of mSDA. As shown in (Chen et al., 2012), mSDA has achieved impressive performance on the cross-domain sentiment classification problem. Moreover, many other domain adaptation methods (Zhou et al., 2014a; Ding et al., 2015; Clinchant et al., 2016) have been proposed by employing mSDA as a building block.

Despite its success, the underlying rationale for why mSDA can bring benefits to domain adaptation is unknown. As emphasized in (Chen et al., 2012), mSDA is specified by two parameters: feature corruption probability and the number of layers. Regarding the feature corruption probability, each feature is independently and identically corrupted by a dropout noise in Chen et al. (2012). The dropout noise variable draws from a *Bernoulli* distribution parameterized by a probability p . Correspondingly, we are interested in questions: why is the denoising phase with *Bernoulli dropout* noise beneficial for domain adaptation? Will other types of noise, for instance, the widely used *Gaussian* noise, also work? More specifically, we are keen to answer a more essential question: what role does the denoising phase of

Figure 3.1: The framework of mSDA

mSDA play in domain adaptation?

In this chapter, we investigate the rationale of mSDA to domain adaptation with respect to the above questions from the perspective of adaptive regularization. We theoretically demonstrate that the effect of the denoising phase in the learning of new feature representations is equivalent to adding a regularization term to the learned feature mapping. Specifically, using *Gaussian* noise to mSDA (mSDA_g) is adding a Frobenius regularization. Such a regularization term controls the complexity of the learned mapping, but may not have impact on the adaptation performance as empirically shown in existing works Pan et al. (2008, 2011); Long et al. (2014a). On the contrary, using *Bernoulli dropout* noise to mSDA (mSDA_{bd} (Chen et al., 2012)) is to add a data-dependent low-rank constraint to the learned mapping. Such a regularization term is very similar to the standard low-rank regularization which has been shown to explore domain-invariant features across different domains (Jhuo et al., 2012; Shao et al., 2012, 2014; Li et al., 2017). We believe this regularization term is beneficial for domain adaptation, and it is verified in the experimental studies. Note that from this paragraph, we distinguish mSDA from mSDA_{bd}, where the former refers to the general stacked denoising framework, and the latter refers to the case using *Bernoulli dropout* noise proposed in (Chen et al., 2012).

Although the denoising with *Bernoulli dropout* noise plays a positive role in domain adaptation, mSDA_{bd} has two issues. On the one hand, how to determine the optimal parameter p is unknown. Existing mSDA methods (Chen et al., 2012; Ding et al., 2015) take advantage of a cross validation procedure on the source domain to do a brute-force search in a predefined value range. However, this may easily result in overfitting to the source domain. On the other hand, features are

Table 3.1: Notations and descriptions

Notation	Description
$\mathcal{D}_{src}, \mathcal{D}_{tar}$	source or target domain
$\mathbf{X}_{src}, \mathbf{X}_{tar}$	source or target data matrix
$\mathbf{x}^s, \mathbf{x}^t$	source or target instance
$\mathbf{Y}_{src}, \mathbf{Y}_{tar}$	source or target label matrix
y^s, y^t	label of $\mathbf{x}^s$ and $\mathbf{x}^t$
n_s, n_t	the number of source or target data instances
d	the dimensionality of the feature space
$n = n_s + n_t$	total number of instances
$\mathbf{X} \in \mathbb{R}^{d \times n}$	source and target joint data matrix
$[\mathbf{X}]_{i,j}$	the i -th row, j -th column entry of $\mathbf{X}$
$\mathbf{x}, \mathbf{f}$	instance vector and feature vector in $\mathbf{X}$
$\widetilde{\mathbf{X}}^r$	the r -th corrupted version of $\mathbf{X}$
$\overline{\mathbf{X}}$	the m -times repeated version of $\mathbf{X}$
$\widetilde{\mathbf{X}}$	the m different corruptions of $\mathbf{X}$

correlated with each other, and in practice, some features may be more informative than the others in adaptation. Compared to mSDA_{bd} that independently corrupts each feature with the same p , it makes more sense to assign different probabilities to different features. This also takes the competition of features into account. To address these two issues and simultaneously preserve the low-rank regularization effect in adaptation, we consider a new type of noise, *Multinomial dropout* noise (Li et al., 2016b), and propose a new type of mSDA, mSDA_{md} . In mSDA_{md} , we use a *Multinomial dropout* noise corruption phase, which allows non-uniform probabilities $p_1, \dots, p_d$ for different features. We incorporate a constraint $\sum_{\alpha} p_{\alpha} = 1$ to enforce the competition among features. To compute a set of data-dependent probabilities, we design each p_{α} by using the second-order statistic of features.

The rest of the chapter is organized as follows. Section 3.1 gives some preliminaries including notations and basic knowledge on mDA. Section 3.2 investigates the rationale of mSDA to domain adaptation by analyzing mSDA_g and mSDA_{bd} from the perspective of adaptive regularization. Section 3.3 describes our proposed mSDA_{md} , a new type of mSDA with data-dependent *Multinomial dropout* noise.

Section 3.4 conducts empirical studies. Finally, section 3.5 summarizes the chapter.

3.1 Problem Setting

In this section, we define variable notations and the domain adaptation problem setting we are working on. Then, we introduce some preliminaries on mSDA.

3.1.1 Problem Statement

We summarize the frequently used notations in this chapter in Table 3.1. Without loss of generality, we consider the case: one target domain $\mathcal{D}_{tar}$ and one source domain $\mathcal{D}_{src}$. Herein, $\mathcal{D}_{src}$ and $\mathcal{D}_{tar}$ share a homogeneous feature space with feature dimensionality d . We denote the source data matrix as $\mathbf{X}_{src} = \{\mathbf{x}_1^s, \dots, \mathbf{x}_{n_s}^s\}$ with n_s samples, and source label vector as $\mathbf{Y}_{src} = \{y_1^s, \dots, y_{n_s}^s\}$. Note that y_i^s is the corresponding class label of $\mathbf{x}_i^s$. We focus on unsupervised domain adaptation, where target labeled data is unavailable. Thus, we denote the target data matrix as $\mathbf{X}_{tar} = \{\mathbf{x}_1^t, \dots, \mathbf{x}_{n_t}^t\}$ with n_t unlabeled samples. We further denote the source and target joint data matrix as $\mathbf{X} = \mathbf{X}_{src} \cup \mathbf{X}_{tar} = [\mathbf{x}_1, \dots, \mathbf{x}_{n_s}, \dots, \mathbf{x}_n] = [\mathbf{f}_1; \mathbf{f}_2; \dots; \mathbf{f}_d] \in \mathbb{R}^{d \times n}$, where $n = n_s + n_t$, $\mathbf{x}_i \in \mathbb{R}^d$ is the i -th instance, and $\mathbf{f}_\alpha \in \mathbb{R}^n$ is the α -th dimensional feature vector. Our objective is to predict labels $\mathbf{Y}_{tar}$ in $\mathcal{D}_{tar}$, by leveraging the labeled data from $\mathcal{D}_{src}$.

3.1.2 The Original mDA

Herein, we are introducing the general denoising framework, marginalized denoising autoencoder (mDA). It is a type of autoencoder that corrupts input data before mapping them into deep representations (Chen et al., 2012). By artificially inserting random noise to the original input data and then reconstructing them from these corrupted ones, mDA learns a linear mapping $\mathbf{W}$ that gives a new feature representation of the original input data. Its objective function is formulated as:

$$\mathcal{L}^m(\mathbf{W})_{noise} = \frac{1}{2mn} \text{tr}[(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}})^\top (\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}})], \quad (3.1)$$

where $\overline{\mathbf{X}} = [\mathbf{X}, \mathbf{X}, \dots, \mathbf{X}]$, is the m -times repeated version of the input $\mathbf{X}$; $\widetilde{\mathbf{X}} = [\widetilde{\mathbf{X}}^1, \widetilde{\mathbf{X}}^2, \dots, \widetilde{\mathbf{X}}^m]$ is the corrupted version of $\overline{\mathbf{X}}$ with m different corruptions of $\mathbf{X}$;

and $tr(\cdot)$ is the operator to calculate the trace of a matrix. To avoid the non-trivial process of actually reconstructing corruptions, mDA learns the robust mapping $\mathbf{W}$ in Eq. (3.1) by averaging over infinitely many m corruptions. This process is named marginalization. As stated in (Chen et al., 2012), marginalization enables an efficient closed-form solution for the mapping $\mathbf{W}$ as shown in Eq. (3.2):

$$\mathbf{W} = \mathbb{E}[\mathbf{P}](\mathbb{E}[\mathbf{Q}])^{-1}, \quad (3.2)$$

where $\mathbf{P} = \bar{\mathbf{X}}\tilde{\mathbf{X}}^\top$, $\mathbf{Q} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top$, and $\mathbb{E}[\cdot]$ is the expectation operator with respect to the noise. By stacking mDA to multiple layers, one can obtain mSDA.

3.2 Rationale of mSDA to Domain Adaptation

In this section, we investigate the underlying rationale of mSDA to domain adaptation. Without loss of generality, we focus on the single layer case mDA. As stated in Chen et al. (2012), the key point of mDA is to learn robust feature representations by reconstructing features from the randomly corrupted ones. It is natural to have a sense that it is the denoising phase in mDA that accounts for the good performance in adaptation tasks. According to Bishop (1995); Wager et al. (2013), in conventional model training, feature denoising phase performs a form of adaptive regularization to avoid overfitting issues. In this work, we also exploit the functionality of the denoising phase in mDA to domain adaptation from the perspective of adaptive regularization. We demonstrate that the denoising phase in mDA plays a role in regularizing the learned feature mapping. Moreover, according to Wager et al. (2013), the effect of regularization is influenced by the type of noise used. Thus, except for the noise used in Chen et al. (2012), which is a *Bernoulli dropout* noise using an independent and identical corruption probability for each feature, we also consider another type of noise that is widely studied, *Gaussian* noise. We give the formal definitions of these two types of noise as follows:

Definition 3.1. Bernoulli dropout noise: $\tilde{\mathbf{x}} = \mathbf{x} \odot \boldsymbol{\xi}$, where $\odot$ is the element-wise product of two vectors. Each component of $\boldsymbol{\xi}$, $\xi_\alpha \in \{0, 1\}$, draws independently a Bernoulli distribution with a probability p , which means each feature is corrupted to 0 with the probability p and retains with the probability $1 - p$.

Definition 3.2. Gaussian noise: $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\tau}$, where $\boldsymbol{\tau} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ is a diagonal matrix with δ_α^2 as the diagonal entry. In another word, given an instance, the α -th feature is corrupted with a Gaussian noise $\tau_\alpha \sim \mathcal{N}(0, \delta_\alpha^2)$.

Correspondingly, we have mSDA_{bd} and mSDA_g . Next, we investigate how the denoising phase using the above two noises regularizes the feature mapping.

3.2.1 Denoising in mDA

We first consider the following simplified object function without denoising and repeated version:

$$\mathcal{L}(\mathbf{W}) = \text{tr}[(\mathbf{X} - \mathbf{W}\mathbf{X})^\top(\mathbf{X} - \mathbf{W}\mathbf{X})]. \quad (3.3)$$

This makes the optimal mapping $\mathbf{W}$ an identity matrix, which is an idle work for adaptation. The corresponding m -times repeated version is:

$$\mathcal{L}^m(\mathbf{W}) = \text{tr}[(\bar{\mathbf{X}} - \mathbf{W}\bar{\mathbf{X}})^\top(\bar{\mathbf{X}} - \mathbf{W}\bar{\mathbf{X}})].$$

Now, we corrupt the data with some noises and denote the corrupted data matrix as $\tilde{\mathbf{X}}^r$:

$$\mathcal{L}(\mathbf{W})_{noise} = \text{tr}[(\mathbf{X} - \mathbf{W}\tilde{\mathbf{X}}^r)^\top(\mathbf{X} - \mathbf{W}\tilde{\mathbf{X}}^r)].$$

According to (Bishop, 1995; Wager et al., 2013; Rifai et al., 2011), the optimal $\mathbf{W}$ can be obtained by solving the following objective function integrating over the feature noise:

$$\mathbf{W} = \underset{\mathbf{W}}{\text{argmin}} \mathbb{E}[\mathcal{L}(\mathbf{W})_{noise}],$$

where

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\mathbf{W})_{noise}] &= \text{tr}(\mathbf{X}^\top \mathbf{X}) - 2\text{tr}(\mathbf{W}^\top \mathbf{X} \mathbb{E}[\tilde{\mathbf{X}}^r]^\top) \\ &\quad + \text{tr}(\mathbf{W}^\top \mathbf{W} \mathbb{E}[\tilde{\mathbf{X}}^r \tilde{\mathbf{X}}^r]^\top). \end{aligned} \quad (3.4)$$

We further denote $\tilde{\mathcal{L}}(\mathbf{W})_{noise} := \mathbb{E}[\mathcal{L}(\mathbf{W})_{noise}]$ for a simplified expression. From Eq. (3.4), we can see that there are two terms, namely $\mathbb{E}[\tilde{\mathbf{X}}^r]^\top$ and $\mathbb{E}[\tilde{\mathbf{X}}^r \tilde{\mathbf{X}}^r]^\top$, that are related to the noise variable. Similar procedure can be done when generalized to the m -times repeated version:

$$\mathbf{W}^* = \underset{\mathbf{W}}{\text{argmin}} \mathbb{E}[\mathcal{L}^m(\mathbf{W})_{noise}],$$

where we denote $\tilde{\mathcal{L}}^m(\mathbf{W})_{noise} := \mathbb{E}[\mathcal{L}^m(\mathbf{W})_{noise}]$.

Next, we start to investigate $\mathbb{E}[\tilde{\mathbf{X}}^r]^\top$ and $\mathbb{E}[\tilde{\mathbf{X}}^r \tilde{\mathbf{X}}^r]^\top$, and then generalize to the m -times repeated version for *Gaussian noise* and *Bernoulli dropout* noise, respectively.

3.2.2 Frobenius Regularization in mDA_g

In mDA_g, with the zero mean *Gaussian* noise, it is easy to obtain: $\mathbb{E}[\widetilde{\mathbf{X}}^r]_g = \mathbf{X}^\top$. For $\mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}]_g$, we have

$$\begin{aligned}
[\mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}]_g]_{\alpha,\beta} &= \mathbb{E}[\sum_{i=1}^n \widetilde{f}_{\alpha,i}^r \widetilde{f}_{\beta,i}^r] \\
&= \mathbb{E}[\sum_{i=1}^n (f_{\alpha,i} + \tau_\alpha)(f_{\beta,i} + \tau_\beta)] \\
&= \mathbb{E}[\sum_{i=1}^n (f_{\alpha,i} f_{\beta,i} + f_{\alpha,i} \tau_\beta + f_{\beta,i} \tau_\alpha + \tau_\alpha \tau_\beta)] \\
&= \sum_{i=1}^n f_{\alpha,i} f_{\beta,i} + n \mathbb{E}[\tau_\alpha \tau_\beta].
\end{aligned} \tag{3.5}$$

From Eq. (3.5), we can see that $\mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}]_g = \mathbf{X} \mathbf{X}^\top + \mathbf{E}$ where when $\alpha \neq \beta$,

$$[\mathbf{E}]_{\alpha,\beta} = 0,$$

and when $\alpha = \beta$,

$$[\mathbf{E}]_{\alpha,\alpha} = n \delta_\alpha^2.$$

We plug $\mathbb{E}[\widetilde{\mathbf{X}}^r]_g$ and $\mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}]_g$ in Eq. (3.4) and obtain:

$$\begin{aligned}
\widetilde{\mathcal{L}}(\mathbf{W})_g &= tr(\mathbf{X}^\top \mathbf{X}) - 2tr(\mathbf{W}^\top \mathbf{X} \mathbf{X}^\top) \\
&\quad + tr(\mathbf{W}^\top \mathbf{W} (\mathbf{X} \mathbf{X}^\top + \mathbf{E})) \\
&= \mathcal{L}(\mathbf{W}) + tr(\mathbf{W}^\top \mathbf{W} \mathbf{E}).
\end{aligned}$$

Then we generalize to the m -times repeated version:

$$\begin{aligned}
\widetilde{\mathcal{L}}^m(\mathbf{W})_g &= \frac{1}{2mn} (tr[(\overline{\mathbf{X}} - \mathbf{W} \overline{\mathbf{X}})^\top (\overline{\mathbf{X}} - \mathbf{W} \overline{\mathbf{X}}]) + R(\mathbf{W})_g) \\
&= \frac{1}{2mn} (\mathcal{L}^m(\mathbf{W}) + R(\mathbf{W})_g),
\end{aligned} \tag{3.6}$$

where $R(\mathbf{W})_g = tr(\mathbf{W}^\top \mathbf{W} \overline{\mathbf{E}})$ and $\overline{\mathbf{E}}$ is the m -times repeated version of $\mathbf{E}$. From Eq. (3.6), we can see that adding the *Gaussian* noise in definition 3.2 is equivalent to adding a Frobenius regularization term $R(\mathbf{W})_g$ to the uncontaminated version of the objective function. On the one hand, it avoids the idle identity solution, and on the other hand, it controls the complexity of $\mathbf{W}$. However, as shown in many domain adaptation existing works (Pan et al., 2008, 2011; Long et al., 2014a), only Frobenius regularization may not bring benefits to adaptation.

3.2.3 Data-dependent Low-rank Regularization in mDA_{bd}

In mDA_{bd}, each feature is independently and identically corrupted to be zero with a probability p . Thus, we can readily derive $\mathbb{E}[\widetilde{\mathbf{X}}^r]_{bd} = (1-p)\mathbf{X}^\top$. For $\mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}]_{bd}$, we utilize the conclusions in (Chen et al., 2012). When $\alpha \neq \beta$,

$$[\mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}]_{bd}]_{\alpha,\beta} = (1-p)^2[\mathbf{X}\mathbf{X}^\top]_{\alpha,\beta},$$

and when $\alpha = \beta$,

$$[\mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}]_{bd}]_{\alpha,\alpha} = (1-p)[\mathbf{X}\mathbf{X}^\top]_{\alpha,\alpha}.$$

Thus, by denoting $\mathbf{D}$ as a diagonal matrix that shares the same diagonal entry with $\mathbf{X}\mathbf{X}^\top$, we can obtain:

$$\begin{aligned} & \widetilde{\mathcal{L}}(\mathbf{W})_{bd} \\ &= tr(\mathbf{X}^\top \mathbf{X}) - 2(1-p) \times tr(\mathbf{W}^\top \mathbf{X}\mathbf{X}^\top) \\ & \quad + tr(\mathbf{W}^\top \mathbf{W}[(1-p)^2 \mathbf{X}\mathbf{X}^\top - (1-p)^2 \mathbf{D} + (1-p)\mathbf{D}]) \\ &= \mathcal{L}(\mathbf{W}) + 2p \times tr(\mathbf{W}^\top \mathbf{X}\mathbf{X}^\top) \\ & \quad + tr(\mathbf{W}^\top \mathbf{W}[(p^2 - 2p)\mathbf{X}\mathbf{X}^\top + (1-p)p\mathbf{D}]). \end{aligned}$$

Then we generalize to the m -times repeated version:

$$\widetilde{\mathcal{L}}^m(\mathbf{W})_{bd} = \frac{1}{2mn}(\mathcal{L}^m(\mathbf{W}) + R(\mathbf{W})_{bd}), \quad (3.7)$$

where

$$R(\mathbf{W})_{bd} = R_{lr}(\mathbf{W})_{bd} + R_F(\mathbf{W})_{bd},$$

$$R_{lr}(\mathbf{W})_{bd} = 2p \times tr(\mathbf{W}^\top \mathbf{X}\mathbf{X}^\top),$$

$$R_F(\mathbf{W})_{bd} = tr((\mathbf{W}\mathbf{L})^\top (\mathbf{W}\mathbf{L}))$$

with

$$\mathbf{L}\mathbf{L}^\top = (p^2 - 2p)\mathbf{X}\mathbf{X}^\top + (1-p)p\mathbf{D}$$

From Eq. (3.7), we conclude that using the *Bernoulli dropout* noise in definition 3.1 in mDA is equivalent to adding two regularization constraints, $R_{lr}(\mathbf{W})_{bd}$ and $R_F(\mathbf{W})_{bd}$, to the feature mapping $\mathbf{W}$. Note that $R_{lr}(\mathbf{W})_{bd}$ is a data-dependent low-rank constraint on $\mathbf{W}^\top \mathbf{X}\mathbf{X}^\top$ and $R_F(\mathbf{W})_{bd}$ is a data-dependent Frobenius constraint on $\mathbf{W}\mathbf{L}$. Considering that $\mathcal{L}^m$ simply minimizes the reconstruction error, we believe

that it is these two data-dependent constraints that enable an effective adaptation in mSDA_{bd} . As shown in many existing domain adaptation methods (Jhuo et al., 2012; Shao et al., 2012; Li et al., 2017), regularizing the learned feature mapping to be low-rank is to explore the shared features of different domains and thus accounts for the effectiveness of adaptation. Although the data-dependent low-rank constraint is not exactly the same as the standard low-rank norm, it also benefits adaptation, which is verified empirically in our experiments.

3.2.4 Discussions

The above theoretical analysis demonstrates us that the objective function of mSDA_g differs from that of mSDA_{bd} on regularization terms, as shown in Eq. (3.6) and Eq. (3.7). With different effects of the regularization terms on the learned feature mapping $\mathbf{W}$, mSDA_{bd} succeeds in benefiting adaptation tasks, but mSDA_g fails to do so. We further verify this conclusion empirically in experimental studies.

Regarding mSDA_{bd} , we notice that the corruption probability p is a vital factor since it weights the importance of low-rank regularization term in the objective function. However, how to determine the optimal uniform feature corruption probability p is unknown. Empirical studies in (Clinchant et al., 2016) guide us to set p as large as possible, but it is contradictory with the minimization objective in Eq. (3.7) that encourages a small p . In (Chen et al., 2012), it has been shown that the optimal p is data/task-dependent, which means the generality of p obtained on specific datasets/tasks is not guaranteed. Existing works (Chen et al., 2012; Ding et al., 2015) conduct a cross-validation (CV) procedure to select p for each task. The CV is done using the source labeled data by a brute-force search on a predefined value range. Nevertheless, as stated in (Li et al., 2014; Zhong et al., 2010; Sugiyama et al., 2007), under the domain adaptation scenario, standard model selection methods, such as CV, do not work as desired since the unbiasedness that guarantees the accuracy of CV does not hold in domain adaptation tasks. This may result in an overfitting to the source domain, and subsequently lead to a major setback when adapting to the target domain.

3.3 mSDA_{md}

In this section, we propose a new mSDA_{md} that utilizes a data-dependent *Multinomial dropout* noise to corrupt features, which can deal with the above issues in mSDA_{bd}.

3.3.1 Motivation

As discussed in section 3.2.3, mSDA_{bd} provides low-rank new feature representations, which are the shared features between the source and target domains. This well answers the question why *Bernoulli dropout* noise can benefit adaptation tasks. However, it also suffers from an issue on the selection of the optimal feature corruption probability p , as discussed in 3.2.4. Moreover, mSDA_{bd} assumes that each feature is corrupted independently with an identical probability. In practice, features are inherently correlated, and some of them may be more significant than the others in adaptation. It makes more sense to assign different corruption probabilities to different features based on their characteristics.

To overcome the above two issues of mSDA_{bd}, we propose an mSDA_{md} that corrupts features using a data-dependent *Multinomial dropout* noise. In mSDA_{md}, the α -th feature is corrupted to be zero with a probability p_α , and the sum of probabilities is equal to 1, i.e., $\sum_\alpha p_\alpha = 1$. By doing so, we naturally take the competition of features into account. To avoid the CV procedure using the source data, we design the probabilities using second-order statistic of features. Next, we elaborate the proposed mSDA_{md}.

3.3.2 A Data-dependent Multinomial Dropout Noise

Inspired by (Li et al., 2016b), we introduce the following data-dependent *Multinomial dropout* noise.

Definition 3.3. *Data-dependent Multinomial Dropout Noise* is defined as: $\tilde{\mathbf{x}} = \mathbf{x} \odot \boldsymbol{\epsilon}$ where each element of $\boldsymbol{\epsilon}$, $\epsilon_\alpha = \frac{m_\alpha}{k}$, is the noise for the α -th feature. Herein, k is the number of independent trials of corruptions, and m_α is the number of times that the α -th feature is selected to be corrupted in the k independent corruptions. In each trial, the α -th feature is selected with a probability p_α . The

variables $\{m_1, m_2, \dots, m_d\}$ thus follow a multinomial distribution $\text{Mult}(p_1, \dots, p_d; k)$ with $\sum_{\alpha=1}^d p_\alpha = 1$, $p_\alpha \geq 0$. Each feature corruption probability, p_α is defined as

$$p_\alpha = \frac{\sqrt{\frac{1}{n} \sum_{i=1}^n f_{\alpha,i}^2}}{\sum_{\alpha'=1}^d \sqrt{\frac{1}{n} \sum_{i=1}^n f_{\alpha',i}^2}} \quad (3.8)$$

where $f_{\alpha,i}$ is the i -th element of the feature vector $\mathbf{f}_\alpha$.

Regarding the definition 3.3, we remark that:

- The value of k controls the degree of feature corruptions. The larger the value of k is, the more corruptions are conducted. In our work, we assign the default value of k to be equal to the dimensionality of input feature space, i.e., $k = d$.
- We have the following statistic characteristics for ϵ_α :

Expectation of ϵ_α :

$$\mathbb{E}[\epsilon_\alpha] = \frac{1}{k} \mathbb{E}[m_\alpha] = p_\alpha;$$

Covariance of ϵ_α and ϵ_β :

$$\text{Cov}(\epsilon_\alpha, \epsilon_\beta) = \frac{1}{k^2} \text{Cov}(m_\alpha, m_\beta) = -\frac{1}{k} p_\alpha p_\beta;$$

Expectation of $\epsilon_\alpha \epsilon_\beta$:

$$\begin{aligned} \mathbb{E}[\epsilon_\alpha \epsilon_\beta] &= \mathbb{E}[\epsilon_\alpha] \mathbb{E}[\epsilon_\beta] + \text{Cov}(\epsilon_\alpha, \epsilon_\beta) \\ &= p_\alpha p_\beta - \frac{1}{k} p_\alpha p_\beta. \end{aligned}$$

Note that the design of ϵ_α is different from that proposed in (Li et al., 2016b), where $\epsilon_\alpha = \frac{m_\alpha}{k p_\alpha}$. Our design makes the statistic characteristics of ϵ_α associated with corruption probabilities, and thus better adapts to mSDA.

- The *Multinomial dropout* noise enables non-uniform probabilities $p_1, \dots, p_d$ for different features. With the constraint $\sum_{\alpha=1}^d p_\alpha = 1$, it gives rise to a competition among different features.
- The *Multinomial dropout* noise takes feature correlations into account. However, we still assume that the corruptions of different instances are independent.

- The aim of denoising is to discover useful information for domain adaptation by reconstructing informative features. Following this aim, the principle of setting p_α is as follows. The more informative a feature is, the larger corruption probability it should be assigned. In this work, we design p_α using second-order statistic of features in Eq. (3.8). Each feature is normalized to be in $[0,1]$. This design considers how features are distributed in samples. If a feature is evenly distributed in the samples, it has a smaller second-order statistic and is more likely to be a trivial feature. Its corruption probability computed by Eq. (A.7) is thus smaller. In contrast, if a feature appears unevenly in the samples, it has a higher second-order statistic and is more likely to be informative. Thus, it is assigned with a higher corruption probability. For example, in sentiment analysis for book reviews, the word ‘book’ appears in almost every review but it is not helpful at all for sentiment analysis. On the contrary, the word ‘comprehensive’ may just appear in a few samples but it is very informative. Note that any other ways of setting p_α that satisfy the above principle are acceptable.
- Once a layer of new features is generated, we update p_α using those new feature representations for corruptions in the next layer.
- Compared to the *Bernoulli dropout* noise that only allows two possible values of ξ_α (0 or 1), *Multinomial dropout* noise enriches the values of ϵ_α with the value diversity of m_α . However, the expectation value of ϵ_α retains to be p_α . If we assign a uniform probability $p_\alpha = 1/d$ for all features, *Multinomial dropout* noise acts similarly to the *Bernoulli dropout* noise with $p = 1/d$.

3.3.3 Data-dependent Multinomial Dropout Noise mSDA

Now, we introduce our mSDA_{md}. We start from the single layer case, mDA_{md}. To obtain mDA_{md}, we need to apply the data-dependent *Multinomial Dropout* noise in definition 3.3 to Eq. (3.1). Correspondingly, the optimal solution of Eq. (3.2) should be adapted to such new type of noise. Similar to (Chen et al., 2012), we also derive the expectations of $\mathbf{P}_{md}$ and $\mathbf{Q}_{md}$ to calculate the optimal solution $\mathbf{W}$. By

defining $\mathbf{R} = \mathbf{X}\mathbf{X}^\top$, for $\mathbf{Q}_{md}$, we have:

$$\begin{aligned}\mathbb{E}[\mathbf{Q}_{md}] &= \mathbb{E}[\widetilde{\mathbf{X}}^r \widetilde{\mathbf{X}}^{r\top}] \\ \Rightarrow [\mathbb{E}[\mathbf{Q}_{md}]]_{\alpha,\beta} &= \mathbb{E}[\widetilde{\mathbf{f}}_\alpha^r \widetilde{\mathbf{f}}_\beta^{r\top}] = \mathbb{E}\left[\sum_{i=1}^n \widetilde{f}_{\alpha,i}^r \widetilde{f}_{\beta,i}^r\right]\end{aligned}$$

When $\alpha \neq \beta$,

$$\begin{aligned}[\mathbb{E}[\mathbf{Q}_{md}]]_{\alpha,\beta} &= \sum_{i=1}^n \mathbb{E}[f_{\alpha,i} f_{\beta,i} \epsilon_\alpha \epsilon_\beta] \\ &= \sum_{i=1}^n f_{\alpha,i} f_{\beta,i} \mathbb{E}[\epsilon_\alpha \epsilon_\beta] \\ &= [\mathbf{R}]_{\alpha,\beta} (p_\alpha p_\beta - \frac{1}{k} p_\alpha p_\beta).\end{aligned}$$

When $\alpha = \beta$,

$$\begin{aligned}[\mathbb{E}[\mathbf{Q}_{md}]]_{\alpha,\beta} &= \sum_{i=1}^n \mathbb{E}[f_{\alpha,i} f_{\alpha,i} \epsilon_\alpha] \\ &= \sum_{i=1}^n f_{\alpha,i} f_{\alpha,i} \mathbb{E}[\epsilon_\alpha] \\ &= [\mathbf{R}]_{\alpha,\alpha} p_\alpha.\end{aligned}$$

Equivalently, $\mathbb{E}[\mathbf{Q}_{md}] = \mathbf{R} \odot \mathbf{U}$ where

$$[\mathbf{U}]_{\alpha,\beta} = \begin{cases} p_\alpha p_\beta - \frac{1}{k} p_\alpha p_\beta, & \text{if } \alpha \neq \beta, \\ p_\alpha, & \text{if } \alpha = \beta. \end{cases}$$

Similarly, we can obtain $[\mathbb{E}[\mathbf{P}_{md}]]_{\alpha,\beta} = [\mathbf{R}]_{\alpha,\beta} p_\beta$. Furthermore, $\mathbb{E}[\mathbf{P}_{md}] = \mathbf{R} \odot \mathbf{V}$ where $[\mathbf{V}]_{\alpha,\beta} = p_\beta$ for all α .

After obtaining these expectation matrices, the mapping $\mathbf{W}$ can be computed as:

$$\mathbf{W} = \mathbb{E}[\mathbf{P}_{md}] (\mathbb{E}[\mathbf{Q}_{md}])^{-1}. \quad (3.9)$$

Once single layer features are obtained, we can readily stack to multiple layers by taking the output of the l -th layer as the input of the $(l+1)$ -th layer. Note that we update $p_1, \dots, p_d$ using the new features learned in the l -th layer as the new corruption probabilities for the $(l+1)$ -th layer.

3.3.4 Data-dependent Low-rank Regularization in mSDA_{md}

Similar to the analyses in section 3.2, we analyze the role that the data-dependent *Multinomial dropout* noise plays in the learning of the feature mapping $\mathbf{W}$. Using $\mathbb{E}[\mathbf{P}_{md}]$ and $\mathbb{E}[\mathbf{Q}_{md}]$ obtained above, we can derive:

$$\begin{aligned} & \tilde{\mathcal{L}}(\mathbf{W})_{md} \\ &= tr(\mathbf{X}^\top \mathbf{X}) - 2tr(\mathbf{W}^\top (\mathbf{R} \odot \mathbf{V})) + tr(\mathbf{W}^\top \mathbf{W} (\mathbf{R} \odot \mathbf{U})) \\ &= \mathcal{L}(\mathbf{W}) + 2tr(\mathbf{W}^\top (\mathbf{R} \odot \mathbf{V}')) + tr(\mathbf{W}^\top \mathbf{W} (\mathbf{R} \odot \mathbf{U}')), \end{aligned}$$

where $\mathbf{V}' = \mathbf{1}_{d \times d} - \mathbf{V}$, $\mathbf{U}' = \mathbf{U} - \mathbf{1}_{d \times d}$, and $\mathbf{1}_{d \times d}$ is a $d \times d$ matrix whose entries are all one. Then we generalize to the m -times repeated version and we have:

$$\widetilde{\mathcal{L}}^m(\mathbf{W})_{md} = \frac{1}{2mn} (\mathcal{L}^m(\mathbf{W}) + R(\mathbf{W})_{md}), \quad (3.10)$$

where

$$R(\mathbf{W})_{md} = R_{lr}(\mathbf{W})_{md} + R_F(\mathbf{W})_{md},$$

$$R_{lr}(\mathbf{W})_{md} = 2tr(\mathbf{W}^\top (\mathbf{R} \odot \mathbf{V}')),$$

$$R_F(\mathbf{W})_{md} = tr(\mathbf{W}^\top \mathbf{W} (\mathbf{R} \odot \mathbf{U}')).$$

From Eq. (3.10), we can see that mSDA_{md} also adds a data-dependent low-rank constraint and a data-dependent Frobenius constraint on the learning of $\mathbf{W}$ as mSDA_{bd} does. This indicates the effectiveness of mSDA_{md} to adaptation.

3.3.5 Discussion

Compared to mSDA_{bd}, our proposed mSDA_{md} not only preserves good properties of mSDA_{bd} to domain adaptation, but also overcomes limitations of mSDA_{bd} that may hinder the adaptation performance. Specifically, we conclude that mSDA_{md}:

- gives the simple optimal closed-form solution as stated in Eq. (3.9), which is similar to that of mSDA_{bd};
- preserves the data-dependant low-rank constraints to the feature mapping as mSDA_{bd} does;
- avoids the CV procedure to select the optimal p , which is a problematic step in mSDA_{bd};

- takes the competition of features into account with a constraint $\sum_{\alpha=1}^d p_{\alpha} = 1$.

These good characteristics well support that mSDA_{md} is a better alternative than mSDA_{bd} in domain adaptation tasks. We empirically verify this in the experiments. However, no matter which type of noise is used in mSDA, the objective of mSDA is only to minimize the reconstruction error without considering the distribution discrepancy issue in the feature learning. This leaves much room to be further improved in terms of the adaptation performance of mSDA.

3.4 Experimental Results

In this section, we compare the adaptation performance of mSDA using different types of noise on three benchmark domain adaptation datasets.

3.4.1 Datasets & Experimental Setting

We use three benchmark datasets that are widely used in unsupervised domain adaptation.

Amazon product review dataset (Blitzer et al., 2007) contains sentiment reviews from four product categories: books (B), DVD (D), electronics (E) and kitchen appliance (K). Each review is characterized by unigram and bigram tf-idf features and labeled as positive or negative. Each domain has about 5,000 samples. When a domain is selected as source (target), all the samples in this domain are used as training (test) data. By pairing up domains for adaptation tasks, we have 12 domain pairs, denoted as ‘source→target’. For example, K→E means that K is the source domain and E is the target domain.

20-Newsgroups dataset (Dai et al., 2007a) is a text dataset that consists of about 20,000 documents. The documents are characterized by the count features, and come from four top categories: computer (C), recording (R), science (S), and talk (T). Each top category has four subcategories. Top categories are treated as labels, while subcategories are treated as related domains. Therefore, six binary prediction tasks are formed: C-R, C-S, C-T, R-S, R-T, and S-T. We take the task C-R for instance. Top category C is the positive class and R is the negative class. Two subcategories under each class are selected to constitute the source domain, while

another two subcategories are selected to form the target domain. By exchanging the roles of the two domains, we have two domain pairs for the prediction task C-R, denoted as C-R1 and C-R2.

COIL1-COIL2 (Long et al., 2013b) is an image dataset for object recognition. Two domains, COIL1 and COIL2, share 20 objects. Each of them has 720 images, and each image is represented by 32×32 pixels with 256 gray levels per pixel. The images in the two domains are taken in different directions, and thus are drawn from different distributions. COIL1 contains images taken in the directions of $[0^\circ, 85^\circ] \cup [180^\circ, 265^\circ]$. COIL2 contains images taken in the directions of $[90^\circ, 175^\circ] \cup [270^\circ, 355^\circ]$. We construct two adaptation tasks: COIL₁₂ and COIL₂₁.

We compare the adaptation performance of mSDA using different types of noise. In theoretical analyses, we conclude that mSDA with *Bernoulli dropout* noise or *Multinomial dropout* noise (mSDA_{bd} or mSDA_{md}) successfully brings benefits to domain adaptation tasks, but mSDA with *Gaussian* noise (mSDA_g) fails to do so. We empirically verify this conclusion by comparing the adaptation performance of these three methods on the three benchmark datasets. For all the three methods, we set the number of layers to be three as default. For mSDA_{bd} and mSDA_g, we do CV on the source domain to select the best corruption probability p between 0.1 and 0.9 with step size 0.1, and the best noise variance δ^2 in the set $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.

3.4.2 Comparison of mSDA with Different Noises

The accuracies of mSDA_g, mSDA_{bd} and mSDA_{md} on the Amazon dataset, the 20-Newsgroups dataset, and the COIL1-COIL2 dataset are shown in Figures 3.2-3.4, respectively. We also show the results of NN as a reference.

As can be seen in these 3 figures, the adaptation performance of mSDA_g varies significantly on different domain adaptation tasks. A clear positive adaptation is observed on the tasks C-T2 and R-T2, and a clear negative adaptation is observed on the tasks C-R1, C-R2, C-S1, C-T1 and R-S2. For the rest majority tasks, mSDA_g performs similarly with the no transfer baseline NN. In terms of the average adaptation performance, mSDA_g achieves comparable results with NN on all the three datasets. Such marginal effects are negligible in the sense that mSDA_g does not significantly contribute to adaptation. This is consistent with the theoretical con-

Figure 3.2: Results of mSDA with different noises on the Amazon dataset

Figure 3.3: Results of mSDA with different noises on the 20-Newsgroup dataset

clusion on mSDA_g. That is, only Frobenius regularization may not bring benefits to adaptation.

On the contrary, mSDA_{bd} consistently achieves better adaptation performance in terms of the average results on the three datasets. Significant improvements are observed compared to NN: 11.93% on the Amazon dataset, 13.36% on the 20-Newsgroups dataset, and 7.16% on the COIL1-COIL2 dataset. Such great success of mSDA_{bd} in adaptation has already been revealed in many existing works. However, in this work, we achieve the unity of theoretical analysis and empirical evaluation. More specifically, we theoretically demonstrate that it is because the data corruption using *Bernoulli dropout* noise is equivalent to a data-dependent low-rank constraint and a data-dependent Frobenius constraint on the feature mapping that facilitates the significant adaptation performance of mSDA_{bd}.

Furthermore, it is worthy noting that the proposed mSDA_{md} further expands the adaptation improvements. Our mSDA_{md} consistently outperforms mSDA_{bd} on all the domain adaptation tasks. Such improvements are due to the different manners of corrupting features between mSDA_{bd} and mSDA_{md}. mSDA_{md} corrupts features with

Figure 3.4: Results of mSDA with different noises on the COIL1-COIL2 dataset

the consideration of feature correlations, and designs data-dependent corruption probabilities. By doing so, it avoids the problematic CV process on the source domain to select p in mSDA_{bd} .

In summary, we conclude that the noise used to corrupt features is a crucial factor to the success of mSDA. Feature corruption with noise is equivalent to adding adaptive regularization terms to the learned feature mapping. Some regularization terms (mSDA_{bd} and mSDA_{md}) are beneficial to domain adaptation, while others (mSDA_g) may not help. Also, the degree of the benefit to adaptation highly depends on the manner of feature corruptions (mSDA_{md} is better than mSDA_{bd} for the adaptation).

3.5 Summary

We summarize the contributions of this chapter as follows:

- We investigate the underlying rationale for why mSDA_{bd} is beneficial for domain adaptation from the perspective of adaptive regularization. The theoretical analyses demonstrate us mSDA with *Bernoulli dropout* noise (mSDA_{bd}) is equivalent to regularizing the learned mapping with a data-dependent low-rank constraint and a data-dependent Frobenius constraint, while mSDA with *Gaussian* noise (mSDA_g) is equivalent to a Frobenius regularization on the mapping.
- We propose a mSDA_{md} , mSDA with data-dependent *Multinomial dropout* noise. The proposed mSDA_{md} inherits the advantages of mSDA_{bd} . On the one hand, it can be efficiently solved by a closed-form solution, which is sim-

ilar to that of mSDA_{bd} . On the other hand, it preserves the data-dependant low-rank constraints to the feature mapping. Moreover, mSDA_{md} overcomes limitations of mSDA_{bd} , that is, mSDA_{md} avoids the problematic CV procedure to select the optimal p and takes feature competitions into account.

- We conduct experiments on real-world datasets to demonstrate the adaptation performance of mSDA with different noises. The results well support our theoretical analyses that mSDA_{bd} and mSDA_{md} can bring benefits to adaptation tasks, but mSDA_g fails to do so.

Chapter 4

Deep Nonlinear Feature Coding

No matter which type of noise is used in mSDA, the objective of mSDA is only to minimize the reconstruction error to guide the feature learning. This makes mSDA suffer from two limitations. First, it does not pay attention to the domain divergence that might arise in the new feature space. As a result, distribution discrepancy on the extracted features may still be large. Second, it injects the nonlinearity after the feature learning. Consequently, the extracted features may not explicitly capture the nonlinear relationship in the data, which compromises one of the biggest strengths of deep learning.

In this chapter, we propose a Deep Nonlinear Feature Coding (DNFC) approach. Our DNFC is dedicated to addressing these two limitations of mSDA for unsupervised domain adaptation. In DNFC, the minimization of domain divergence and the exploitation of data nonlinearity are incorporated seamlessly into the deep feature learning. More specifically, we introduce two new elements, Maximum Mean Discrepancy (MMD) and kernelization, into 1-layer mSDA and stack the single-layer nonlinear feature coding to create higher-level deep features. As illustrated in Figure 4.1, at each layer, we measure the distribution discrepancy between the source and target domains in the new deep feature space by MMD and minimize it in the feature learning process. This enforces the distribution discrepancy in the new feature representation, on which domain adaptation is performed, to be minimized. Moreover, we also develop a kernelization that fuses the nonlinearity exploitation with the feature learning. Both elements play important roles in learning good features for domain adaptation and are enclosed in a unified framework. The proposed DNFC framework is shown in figure 4.1.

Figure 4.1: Deep nonlinear feature coding framework

The rest of the chapter is organized as follows. Section 4.1 describes the details of our proposed DNFC approach, and discusses the applications of DNFC framework to both *Bernoulli dropout* noise case and *Multinomial dropout* noise case. Section 4.2 demonstrates the effectiveness of our DNFC by extensive experimental studies. Finally, section 4.3 summarizes the chapter.

4.1 A Deep Nonlinear Coding Approach

In this section, we introduce the proposed DNFC approach in details.

4.1.1 Problem Statement

We focus on the unsupervised domain adaptation problem setting, which is well stated in section 3.1.1. We follow notations defined in Table 3.1.

4.1.2 Domain Divergence Minimization by MMD

We propose to use MMD to quantify the domain divergence in the learned deep feature space and incorporate it in the learning of $\mathbf{W}$. This is particularly essential for unsupervised learning of deep features since no labeled data in the target domain is available to guide the feature learning.

Denote $\tilde{\mathbf{x}}_i^r$ as the i -th sample in the r -th corrupted version of $\mathbf{X}$. Following the definition of MMD, the empirical distance between source and target domains in

the r -th reconstructed space is:

$$\begin{aligned}\mathcal{D}(\mathbf{W})_{noise} &= \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{W} \tilde{\mathbf{x}}_i^r - \frac{1}{n_t} \sum_{i=n_s+1}^n \mathbf{W} \tilde{\mathbf{x}}_i^r \right\|^2 \\ &= \text{tr}(\mathbf{W} \tilde{\mathbf{X}}^r \mathbf{M} \tilde{\mathbf{X}}^r{}^\top \mathbf{W}^\top)\end{aligned}\quad (4.1)$$

where $\mathbf{M} = [M_{i,j}]_{n \times n}$ with $M_{i,j} = \frac{1}{n_s^2}$ if $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_{src}$; $M_{i,j} = \frac{1}{n_t^2}$ if $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_{tar}$; otherwise $M_{i,j} = -\frac{1}{n_s n_t}$.

Combining Eq. (3.1) and Eq. (4.1), and considering m corrupted versions, we formulate the objective function as:

$$\begin{aligned}\mathcal{F}^m(\mathbf{W})_{dnc} &= \mathcal{L}^m(\mathbf{W})_{noise} + \theta \mathcal{D}^m(\mathbf{W})_{noise} \\ &= \text{tr}[(\bar{\mathbf{X}} - \mathbf{W} \tilde{\mathbf{X}})^\top (\bar{\mathbf{X}} - \mathbf{W} \tilde{\mathbf{X}})] \\ &\quad + \theta \text{tr}(\mathbf{W} \tilde{\mathbf{X}} \tilde{\mathbf{M}} \tilde{\mathbf{X}}^\top \mathbf{W}^\top),\end{aligned}\quad (4.2)$$

where $\mathcal{D}^m(\mathbf{W})_{noise}$ is the m -times repeated version of $\mathcal{D}(\mathbf{W})_{noise}$, $\tilde{\mathbf{M}}$ is a block diagonal matrix with m copies of $\mathbf{M}$ as the diagonal elements, and $\theta > 0$ is the balancing parameter. Eq. (4.2) has a closed-form solution:

$$\begin{aligned}\mathbf{W} &= \mathbf{P}(\mathbf{Q}_1 + \theta \mathbf{Q}_2)^{-1} \text{ with} \\ \mathbf{P} &= \bar{\mathbf{X}} \tilde{\mathbf{X}}^\top, \quad \mathbf{Q}_1 = \tilde{\mathbf{X}} \tilde{\mathbf{X}}^\top, \quad \mathbf{Q}_2 = \tilde{\mathbf{X}} \tilde{\mathbf{M}} \tilde{\mathbf{X}}^\top.\end{aligned}\quad (4.3)$$

The computation of $\mathbf{W}$ relies on the construction of m corruptions of $\mathbf{X}$. The more corruptions are constructed, the more robust $\mathbf{W}$ is. Similar to (Chen et al., 2012), we apply the weak law of large numbers and derive the expectations of $\mathbf{P}$, $\mathbf{Q}_1$ and $\mathbf{Q}_2$ as $m \rightarrow \infty$. Specifically, we focus on feature corruptions using *Bernoulli dropout* noise and the *Multinomial dropout* noise, respectively. We further define $\mathbf{R} = \mathbf{X} \mathbf{X}^\top$, $\mathbf{S} = \mathbf{X} \mathbf{M} \mathbf{X}^\top$, and $\mathbf{T} = \mathbf{X} \mathbf{N} \mathbf{X}^\top$ ($\mathbf{N}$ is a diagonal matrix whose diagonal elements are the same as those in $\mathbf{M}$).

4.1.2.1 Bernoulli Dropout Noise Case

We use the *Bernoulli dropout* noise defined in definition 3.1. Note that in this case all feature corruptions are mutually independent. For $\mathbf{Q}_1$, we have:

$$\begin{aligned}\mathbb{E}[\mathbf{Q}_1] &= \mathbb{E}[\tilde{\mathbf{X}}^r \tilde{\mathbf{X}}^r{}^\top] \\ \Rightarrow [\mathbb{E}[\mathbf{Q}_1]]_{\alpha, \beta} &= \mathbb{E}[\tilde{\mathbf{f}}_\alpha^r \tilde{\mathbf{f}}_\beta^r{}^\top] = \mathbb{E}\left[\sum_{i=1}^n \tilde{f}_{\alpha, i}^r \tilde{f}_{\beta, i}^r\right].\end{aligned}$$

When $\alpha \neq \beta$,

$$\begin{aligned} [\mathbb{E}[\mathbf{Q}_1]]_{\alpha,\beta} &= \sum_{i=1}^n f_{\alpha,i} f_{\beta,i} \mathbb{E}[\xi_\alpha] \mathbb{E}[\xi_\beta] \\ &= (1-p)^2 [\mathbf{R}]_{\alpha,\beta}. \end{aligned}$$

When $\alpha = \beta$,

$$\begin{aligned} [\mathbb{E}[\mathbf{Q}_1]]_{\alpha,\alpha} &= \sum_{i=1}^n f_{\alpha,i} f_{\alpha,i} \mathbb{E}[\xi_\alpha] \\ &= (1-p) [\mathbf{R}]_{\alpha,\alpha}. \end{aligned}$$

For $\mathbf{Q}_2$, we have:

$$[\mathbb{E}[\mathbf{Q}_2]]_{\alpha,\beta} = \mathbb{E}[\widetilde{\mathbf{f}}_\alpha^r \mathbf{M} \widetilde{\mathbf{f}}_\beta^{r\top}] = \mathbb{E}\left[\sum_{i=1}^n \sum_{j=1}^n \widetilde{f}_{\alpha,i}^r M_{i,j} \widetilde{f}_{\beta,j}^r\right].$$

When $\alpha \neq \beta$,

$$\begin{aligned} [\mathbb{E}[\mathbf{Q}_2]]_{\alpha,\beta} &= \sum_{i=1}^n \sum_{j=1}^n f_{\alpha,i} M_{i,j} f_{\beta,j} \mathbb{E}[\xi_\alpha] \mathbb{E}[\xi_\beta] \\ &= (1-p)^2 [\mathbf{S}]_{\alpha,\beta}. \end{aligned}$$

When $\alpha = \beta$,

$$\begin{aligned} &[\mathbb{E}[\mathbf{Q}_2]]_{\alpha,\alpha} \\ &= \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n f_{\alpha,i} M_{i,j} f_{\alpha,j} \mathbb{E}[\xi_\alpha]^2 + \sum_{i=1}^n f_{\alpha,i}^2 M_{i,i} \mathbb{E}[\xi_\alpha] \\ &= (1-p)^2 [\mathbf{S}]_{\alpha,\alpha} + p(1-p) [\mathbf{T}]_{\alpha,\alpha}. \end{aligned}$$

Similarly, we can obtain the expectation of $\mathbf{P}$ as $[\mathbb{E}[\mathbf{P}]]_{\alpha,\beta} = (1-p) [\mathbf{R}]_{\alpha,\beta}$.

4.1.2.2 Multinomial Dropout Noise Case

We use the *Multinomial dropout* noise defined in definition 3.3. Note that in this case only the corruptions of different instances are independent. For $\mathbf{Q}_1$, we have: when $\alpha \neq \beta$,

$$\begin{aligned} [\mathbb{E}[\mathbf{Q}_1]]_{\alpha,\beta} &= \sum_{i=1}^n f_{\alpha,i} f_{\beta,i} \mathbb{E}[\epsilon_\alpha \epsilon_\beta] \\ &= (p_\alpha p_\beta - \frac{1}{k} p_\alpha p_\beta) [\mathbf{R}]_{\alpha,\beta}. \end{aligned}$$

When $\alpha = \beta$,

$$\begin{aligned}\mathbb{E}[\mathbf{Q}_1]_{\alpha,\alpha} &= \sum_{i=1}^n f_{\alpha,i} f_{\alpha,i} \mathbb{E}[\epsilon_\alpha] \\ &= p_\alpha [\mathbf{R}]_{\alpha,\alpha}.\end{aligned}$$

For $\mathbf{Q}_2$, we have: when $\alpha \neq \beta$,

$$\begin{aligned}[\mathbb{E}[\mathbf{Q}_2]]_{\alpha,\beta} &= \sum_{i=1}^n \sum_{j=1}^n f_{\alpha,i} M_{i,j} f_{\beta,j} \mathbb{E}[\epsilon_\alpha \epsilon_\beta] \\ &= (p_\alpha p_\beta - \frac{1}{k} p_\alpha p_\beta) [\mathbf{S}]_{\alpha,\beta}.\end{aligned}$$

When $\alpha = \beta$,

$$\begin{aligned}[\mathbb{E}[\mathbf{Q}_2]]_{\alpha,\alpha} &= \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n f_{\alpha,i} M_{i,j} f_{\alpha,j} \mathbb{E}[\epsilon_\alpha]^2 + \sum_{i=1}^n f_{\alpha,i}^2 M_{i,i} \mathbb{E}[\epsilon_\alpha] \\ &= p_\alpha^2 [\mathbf{S}]_{\alpha,\alpha} + p_\alpha (1 - p_\alpha) [\mathbf{T}]_{\alpha,\alpha}.\end{aligned}$$

Similarly, we can obtain $[\mathbb{E}[\mathbf{P}]]_{\alpha,\beta} = [\mathbf{R}]_{\alpha,\beta} p_\beta$.

For both noise cases, after obtaining these expectation matrices, the mapping $\mathbf{W}$ can be computed as:

$$\mathbf{W} = \mathbb{E}[\mathbf{P}] (\mathbb{E}[\mathbf{Q}_1] + \theta \mathbb{E}[\mathbf{Q}_2])^{-1}.$$

Our new feature coding at a single layer is then defined as:

$$\mathbf{Z} = \mathbf{W}\mathbf{X}.$$

By incorporating MMD into mDA, we not only take advantage of mDA in obtaining discriminative deep features but also ensure that the new feature space brings the two domains as close as possible. As evidenced by our experimental results, this leads to much better adaptation performance. We also remark that the incorporation of MMD is seamless in the sense that the marginalization property of mDA is preserved and thus the efficiency of the proposed approach is ensured.

4.1.3 Nonlinear Coding by Kernelization

With the learning of $\mathbf{W}$, only the linear relationship in the input data is captured. In the original mDA, the nonlinearity is inserted by applying a nonlinear squashing

function on the output of mDA. Consequently, the nonlinearity in the data may not be well exploited because it is not taken care of in the learning of $\mathbf{W}$. Note that with the incorporation of MMD, such a neglect also harms the minimization of domain divergence in the deep feature space.

In order to address this issue, we propose a kernelized solution. Specifically, we use a nonlinear mapping function Φ to map the original data to a RKHS. We then reconstruct the original data by randomly corrupting the mapped data in the RKHS. By doing so, we factor in the nonlinearity in the learning of $\mathbf{W}$ and thus in the extracted deep features.

Suppose that $\mathbf{X}$ is mapped to a RKHS $\mathcal{H}$ by a nonlinear mapping function $\Phi : \mathbb{R}^d \rightarrow \mathcal{H}$. Denote $\Phi(\mathbf{X}) = [\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_{n_s}), \dots, \Phi(\mathbf{x}_n)]$ as the mapped data matrix and $\mathbf{K} = \Phi(\mathbf{X})^\top \Phi(\mathbf{X})$ as the corresponding kernel matrix. The mDA term to reconstruct the original data from the corrupted mapped data now becomes:

$$\mathcal{L}^m(\mathbf{W})_{noise} = \frac{1}{2mn} \text{tr}[(\bar{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})^\top (\bar{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})]$$

where $\widetilde{\Phi(\mathbf{X})} = [\widetilde{\Phi(\mathbf{X})^1}, \widetilde{\Phi(\mathbf{X})^2}, \dots, \widetilde{\Phi(\mathbf{X})^m}]$ represents the corrupted mapped data with m different corruptions of $\Phi(\mathbf{X})$.

With the incorporation of nonlinearity, the objective function that combines mDA and MMD terms becomes:

$$\begin{aligned} \mathcal{F}^m(\mathbf{W})_{dnfc} = & \text{tr}[(\bar{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})^\top (\bar{\mathbf{X}} - \mathbf{W}\widetilde{\Phi(\mathbf{X})})] \\ & + \theta \text{tr}(\mathbf{W}\widetilde{\Phi(\mathbf{X})} \widetilde{\mathbf{M}} \widetilde{\Phi(\mathbf{X})}^\top \mathbf{W}^\top) \end{aligned}$$

According to (Niyogi, 2004), the linear mapping $\mathbf{W}$ in $\mathcal{H}$ can be represented as a linear combination of the data points in $\mathcal{H}$, that is, $\mathbf{W} = \mathbf{W}_k \Phi(\mathbf{X})^\top$. The objective function thus becomes:

$$\begin{aligned} \mathcal{F}^m(\mathbf{W}_k)_{dnfc} = & \text{tr}[(\bar{\mathbf{X}} - \mathbf{W}_k \widetilde{\mathbf{K}})^\top (\bar{\mathbf{X}} - \mathbf{W}_k \widetilde{\mathbf{K}})] \\ & + \theta \text{tr}(\mathbf{W}_k \widetilde{\mathbf{K}} \widetilde{\mathbf{M}} \widetilde{\mathbf{K}}^\top \mathbf{W}_k^\top), \end{aligned} \tag{4.4}$$

where $\widetilde{\mathbf{K}} = [\widetilde{\mathbf{K}}^1, \widetilde{\mathbf{K}}^2, \dots, \widetilde{\mathbf{K}}^m]$ is the m corrupted matrix with $\widetilde{\mathbf{K}}^r = [\widetilde{\mathbf{k}}_1^r; \widetilde{\mathbf{k}}_2^r; \dots; \widetilde{\mathbf{k}}_n^r]$, $\widetilde{\mathbf{k}}_i^r = [\widetilde{k}_{i,1}^r, \widetilde{k}_{i,2}^r, \dots, \widetilde{k}_{i,n}^r]$ and $\widetilde{k}_{i,j}^r = \Phi(\mathbf{x}_i)^\top \widetilde{\Phi(\mathbf{x}_j)^r}$. The closed-form solution of Eq. (4.4) is:

$$\begin{aligned} \mathbf{W}_k = & \mathbf{P}_k (\mathbf{Q}_{k1} + \theta \mathbf{Q}_{k2})^{-1} \text{ with} \\ \mathbf{P}_k = & \bar{\mathbf{X}} \widetilde{\mathbf{K}}^\top, \quad \mathbf{Q}_{k1} = \widetilde{\mathbf{K}} \widetilde{\mathbf{K}}^\top, \quad \mathbf{Q}_{k2} = \widetilde{\mathbf{K}} \widetilde{\mathbf{M}} \widetilde{\mathbf{K}}^\top. \end{aligned}$$

4.1.3.1 Bernoulli Dropout Noise Case

We first adapt kernelization to *Bernoulli dropout* noise case. Since corrupting $\Phi(\mathbf{X})$ corresponds to corrupt the kernel matrix $\mathbf{K}$, we assume that each element $\widetilde{k_{i,j}^r}$ is corrupted with probability p . We now define the kernelized counterparts: $\mathbf{q}' = [1 - p, 1 - p, \dots, 1 - p] \in \mathbb{R}^n$, $\mathbf{R}' = \mathbf{K}\mathbf{K}^\top$, $\mathbf{S}' = \mathbf{K}\mathbf{M}\mathbf{K}^\top$, and $\mathbf{T}' = \mathbf{K}\mathbf{N}\mathbf{K}^\top$. The expected values of $\mathbf{Q}_{k1}$ and $\mathbf{Q}_{k2}$ can be obtained in a similar way to those of $\mathbf{Q}_1$ and $\mathbf{Q}_2$:

$$\begin{aligned} E[\mathbf{Q}_{k1}]_{\alpha,\beta} &= E\left[\sum_{i=1}^n \widetilde{k_{\alpha,i}^r} \widetilde{k_{\beta,i}^r}\right] = \mathbf{q}'_\alpha \mathbf{q}'_\beta \mathbf{R}'_{\alpha,\beta}, \\ E[\mathbf{Q}_{k1}]_{\alpha,\alpha} &= E\left[\sum_{i=1}^n \widetilde{k_{\alpha,i}^r} \widetilde{k_{\alpha,i}^r}\right] = \mathbf{q}'_\alpha \mathbf{R}'_{\alpha,\alpha}, \\ E[\mathbf{Q}_{k2}]_{\alpha,\beta} &= E\left[\sum_{i=1}^n \sum_{j=1}^n \widetilde{k_{\alpha,i}^r} M_{i,j} \widetilde{k_{\beta,j}^r}\right] = \mathbf{q}'_\alpha \mathbf{q}'_\beta \mathbf{S}'_{\alpha,\beta}, \\ E[\mathbf{Q}_{k2}]_{\alpha,\alpha} &= E\left[\sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n \widetilde{k_{\alpha,i}^r} M_{i,j} \widetilde{k_{\alpha,j}^r} + \sum_{i=1}^n \widetilde{k_{\alpha,i}^r} M_{i,i} \widetilde{k_{\alpha,i}^r}\right] \\ &= \mathbf{q}'_\alpha \mathbf{q}'_\alpha \mathbf{S}'_{\alpha,\alpha} + \mathbf{q}'_\alpha (1 - \mathbf{q}'_\alpha) \mathbf{T}'_{\alpha,\alpha}. \end{aligned}$$

Similarly, we define $\mathbf{U} = \mathbf{X}\mathbf{K}^\top$ and obtain the expectation of the matrix $\mathbf{P}_k$: $E[\mathbf{P}_k]_{\alpha,\beta} = \mathbf{q}'_\beta \mathbf{U}_{\alpha,\beta}$. Then, we obtain the mapping matrix:

$$\mathbf{W}_k = E[\mathbf{P}_k](E[\mathbf{Q}_{k1}] + \theta E[\mathbf{Q}_{k2}])^{-1}. \quad (4.5)$$

The nonlinear feature coding at a single layer is then defined as:

$$\mathbf{Z} = \mathbf{W}_k \mathbf{K}. \quad (4.6)$$

It encloses both the data nonlinearity and domain similarity in the deep feature space.

4.1.3.2 Multinomial Dropout Noise Case

We now turn to the scenario that applies kernelization to *Multinomial dropout* noise case. When considering the multinomial dropout noise, the corruption probabilities depend on the input data and are non-uniform. When kernelization is applied, the corruption patterns of the mapped data and the kernel matrix become complex and

Algorithm 4.1 Deep Nonlinear Feature Coding

Input: Source data matrix $\mathbf{X}_{src}$, target data matrix $\mathbf{X}_{tar}$, and the number of layers L .

for $k = 1$ **to** L **do**

1. Select kernel function using cross validation on source;
2. Learn coding $\mathbf{Z}_{src}^k$ and $\mathbf{Z}_{tar}^k$ by Eq. (4.6);
3. Set $\mathbf{X}_{src}^{k+1} = \mathbf{Z}_{src}^k$ and $\mathbf{X}_{tar}^{k+1} = \mathbf{Z}_{tar}^k$.

end for

Output: Feature coding $\{\mathbf{Z}_{src}^k, \mathbf{Z}_{tar}^k\}, (k = 1, \dots, L)$.

the expectation matrices cannot be marginalized and become intractable. This is unlike *Bernoulli dropout* noise, where we can assume each element $\widetilde{k_{i,j}^r}$ is corrupted with a uniform probability p . Thus, for *Multinomial dropout* noise case, the kernelization may not be easily applicable.

4.1.4 Deep Nonlinear Feature Coding (DNFC)

Similar to the stacking of mDA in mSDA, we stack our single-layer feature coding so as to create richer deep feature representations. The framework is illustrated in Figure 4.1. The stacking is performed by feeding the output $\mathbf{Z} = [\mathbf{Z}_{src}, \mathbf{Z}_{tar}]$ of each layer into the next layer as the input. In this way, higher-level deep structures can be captured. To avoid overfitting the data nonlinearity during stacking, we conduct cross validation at each layer on the source data to select a suitable kernel function. Our proposed DNFC is summarized in algorithm 4.1.

For domain adaptation, we concatenate the outputted feature coding at all L layers as the final feature representation. A classifier is then trained on the labeled source data using these features and applied to the target domain to perform the prediction.

4.1.5 Complexity Analysis on DNFC

DNFC is very effective to be solved as it has closed-form solution. The closed form solution can be obtained by solving system of linear equations as shown in Eq. (4.5). The computational time complexity in solving this system of linear equations is dominated by the calculation of matrix inverse. Specifically, the computational time

complexity of DNFC is $\mathcal{O}(n^3)$ with n as the number of instances. When the number of features is much less than the number of samples, mSDA, whose complexity is $\mathcal{O}(d^3)$, is computational cheaper than DNFC. However, when the feature space of two domains is huge and sparse, DNFC may be more efficient.

4.2 Experimental Results

In this section, we evaluate the performance of DNFC by comparing with state-of-the-art domain adaptation methods. Specifically, we only focus on the *Bernoulli dropout* noise case, DNFC_{bd} , as it can seamlessly enclose the two new elements, MMD and kernelization, to mSDA. We also study the properties of DNFC_{bd} and analyze the effect of the two new elements in DNFC_{bd} .

4.2.1 Datasets & Experimental Setting

We use two benchmark datasets, namely *Amazon product review dataset* and *20-Newsgroups dataset*, that have been introduced in the experimental studies of Chapter 3. We follow the same experimental setting with respect to these two datasets stated in Chapter 3.

We compare DNFC_{bd} with five baselines: NN, which predicts target labels using the nearest neighbour classifier trained on the source domain with original features, DASA (Fernando et al., 2013), GFK (Gong et al., 2012), TJM (Long et al., 2014a), JDA (Long et al., 2013b), COR (Sun et al., 2016), and mSDA_{bd} (Chen et al., 2012). Following (Gong et al., 2012; Long et al., 2014a), we use 1-NN as the base classifier since it avoids model parameter tuning. For subspace-based methods DASA, GFK, and TJM, we use the default subspace dimension d if it is specified by the authors, otherwise we set d to be the smallest one that contributes 85% of the energy. For TJM, we use the default kernel function specified by the authors. In DNFC_{bd} , we conduct cross validation on source to automatically select between ‘rbf’ and linear kernels at each layer. For mSDA_{bd} and DNFC_{bd} , we set the default number of layers as three and do the cross validation on source to select the best corruption probability p .

4.2.2 Comparison with Baselines

The classification accuracies of DNFC_{bd} and five baselines on the two benchmarks are reported in Table 4.1. The average performance of each method is also shown. The best result on each domain pair is highlighted in bold, and the runner-up is highlighted in bold-italic.

As can be seen in Table 4.1, our DNFC achieves very promising performance and is a clear winner. Out of 24 domain pairs, DNFC_{bd} performs the best in 21 and the second best in 3. For mSDA_{bd} , on which DNFC_{bd} is based, it is the best for 3 domain pairs and the runner-up for 15 domain pairs. The remaining 6 runner-ups mostly go to TJM, which is the most recent subspace-based method with nonlinear feature mapping. The performance improvement of DNFC_{bd} over the second best method mSDA_{bd} is up to 14.12% and is 5.32% on average. This demonstrates that the two new elements in DNFC_{bd} are very effective in achieving better domain adaptation.

The results also show that the deep feature learning methods, DNFC_{bd} and mSDA_{bd} , significantly outperform the subspace-based methods. This demonstrates the superiority of exploiting deep structures. The success of DNFC_{bd} and TJM (with respect to other subspace-based methods) also reveals the necessity of capturing the data nonlinearity. Our DNFC_{bd} takes advantage of both and thus is able to significantly boost the domain adaptation performance.

4.2.3 Property Study on DNFC

In this section, we study our proposed DNFC_{bd} in four aspects: 1) sensitivity analysis on the balancing parameter θ ; 2) the influence of the number of layers; 3) the kernel functions selected in each layer; and 4) the effect of feature concatenation. We only report the results of three domain pairs: S-T1, R-T1, and K→E as representative examples.

As shown in Figure 4.2(a), with the increase of θ , the classification accuracy generally increases until it gets stable. The larger θ is, the more sensitive the MMD term is to the objective function. When the empirical MMD gets stable, larger values of θ cannot further improve the performance. For all three domain pairs, this happens when θ reaches 10^3 , which is also the case for all other domain pairs. In our experiments, we set $\theta = 10^3$ as default.

Table 4.1: Accuracy (%) on two benchmark datasets

Dataset	NN	DASA	GFK	TJM	COR	JDA	mSDA _{bd}	DNFC _{bd}
B → D	51.04	59.42	55.49	61.92	59.02	64.19	64.79	66.66
B → E	53.26	57.36	55.84	59.92	59.96	61.36	65.79	66.43
B → K	56.37	60.68	58.86	61.65	60.94	63.15	64.01	69.35
D → B	52.51	61.29	56.95	63.34	59.36	64.07	67.09	70.84
D → E	55.23	59.23	57.28	63.16	61.08	63.72	67.07	67.31
D → K	56.53	61.36	59.27	65.52	62.65	63.23	69.06	68.35
E → B	51.45	56.82	54.29	61.16	60.29	60.78	62.13	62.22
E → D	51.37	57.21	54.24	62.94	61.21	61.57	63.19	65.16
E → K	54.84	65.62	61.66	70.84	61.21	70.26	75.49	72.73
K → B	50.93	51.93	51.44	61.74	61.18	53.76	52.95	62.65
K → D	51.86	57.31	54.36	63.35	61.13	59.67	60.79	63.83
K → E	53.55	59.38	54.53	69.37	61.82	65.18	69.67	74.11
C-R1	56.21	57.75	60.33	60.75	55.27	59.21	53.90	68.98
C-R2	55.22	60.73	60.31	63.36	57.68	67.01	79.30	78.03
C-S1	60.31	61.08	55.86	63.13	56.80	64.93	63.47	77.59
C-S2	54.51	61.07	56.73	60.39	55.20	60.14	58.86	72.40
C-T1	59.62	72.15	68.61	77.13	60.10	69.76	85.93	87.18
C-T2	58.78	68.56	67.82	76.45	64.35	72.24	87.38	91.27
R-S1	59.29	62.91	49.54	64.34	59.63	66.02	68.63	77.38
R-S2	62.31	50.67	65.60	62.90	62.39	66.86	71.84	74.70
R-T1	51.89	60.89	53.50	67.99	52.84	55.97	59.75	76.61
R-T2	53.49	57.80	58.32	64.07	58.52	54.21	60.68	69.82
S-T1	54.78	62.63	53.26	65.18	57.05	57.14	69.54	82.02
S-T2	57.39	64.74	59.57	62.77	58.63	66.08	75.28	83.14
Average	55.11	60.36	57.65	64.72	60.52	62.94	67.36	72.68

Figure 4.2(b) shows how the number of layers affects the performance of DNFC_{bd}. The results of mSDA_{bd} are also shown as a reference. In general, both DNFC_{bd} and mSDA_{bd} obtain better results when more layers are stacked. However, they have

(a) Sensitivity analysis on the balancing parameter θ

(b) Influence of the number of layers

(c) Kernel functions selected at each layer

Figure 4.2: Property study on DNFC_{bd}

different trends. With the increase in the number of layers, the performance of DNFC_{bd} converges very fast, while that of mSDA_{bd} improves gradually. Even after 7 layers of stacking, mSDA_{bd} is still not as good as DNFC_{bd} . In particular, the accuracy of mSDA_{bd} is about 5% worse than that of DNFC_{bd} for S-T1 and R-T1 with 7 layers of stacking. The different trends of mSDA_{bd} and DNFC_{bd} may be due to the different ways of exploiting the data nonlinearity. mSDA_{bd} injects the nonlinearity to the output of each layer, while DNFC_{bd} incorporates the consideration of nonlinearity into the feature learning process. As a result, the data nonlinearity can be captured by DNFC_{bd} more quickly with a small number of stacked layers. Considering that more stacked layers incur heavier computational overheads, the quicker convergence of DNFC_{bd} also becomes a big strength, which makes it more practically useful in

real applications. Figure 4.2(c) shows the kernels selected by CV on the source domain at different layers of DNFC_{bd} . It demonstrates the flexibility of selecting different suitable kernels for different layers even for the same domain pair.

4.2.4 Effect of New Elements in DNFC

In this section, we assess the individual effect of MMD and kernelization in DNFC_{bd} , as well as their joint effects, with respect to the performance of domain adaptation. We develop two variants of mSDA_{bd} : one with MMD but not kernelization,

Table 4.2: Accuracy (%) on 20-Newsgroups for assessing the effect of new elements in DNFC_{bd}

Dataset	mSDA_{bd}	$\text{mSDA}_{bd}\text{-MMD}$	$\text{mSDA}_{bd}\text{-kernel}$	DNFC_{bd}
C-R1	53.90	71.21	58.27	68.98
C-R2	79.30	77.95	75.40	78.03
C-S1	63.47	70.92	73.48	77.59
C-S2	58.86	63.63	69.59	72.40
C-T1	85.93	89.00	87.46	87.18
C-T2	87.38	87.49	90.75	91.27
R-S1	68.63	75.71	65.85	77.38
R-S2	71.84	71.33	65.26	74.70
R-T1	59.75	72.54	59.94	76.61
R-T2	60.68	67.97	52.87	69.82
S-T1	69.54	78.05	77.29	82.02
S-T2	75.28	76.73	83.35	83.14
Average	69.55	75.21	71.63	78.26

the other with the kernelization but not MMD. The two variants are referred to as $\text{mSDA}_{bd}\text{-MMD}$ and $\text{mSDA}_{bd}\text{-Kernel}$, respectively. We compare DNFC_{bd} with mSDA_{bd} and the two variants on the 20-Newsgroups dataset. The number of layers is set as three for all methods.

The results are reported in Table 4.2. From the table, we can see that, overall-

l, either MMD or kernelization alone improves the performance of mSDA_{bd} . This demonstrates the effectiveness of each individual element in DNFC_{bd} . Moreover, as evidenced by the performance of DNFC_{bd} , integrating MMD and kernelization together is able to further boost the improvement over mSDA_{bd} . We notice that the impacts of MMD and kernelization could vary even within the same domains. This is because their effects are affected by the choices of p and the kernel, which are selected using cross validation on source data. We also observe occasional inferior performance of DNFC_{bd} and the variants to mSDA_{bd} . This may be again due to the parameter selection on source data, which might result in overfitting when generalizing to target domain.

4.3 Summary

We summarize the contributions of this chapter as follows:

- We introduce two new elements, MMD and kernelization, to mSDA , and propose a new deep nonlinear feature coding (DNFC) framework. DNFC incorporates MMD and kernelization into mSDA to extract nonlinear deep features with minimum domain divergence, which overcomes limitations of mSDA .
- We discuss the availability of applying the idea of DNFC to both *Bernoulli dropout* noise and *Multinomial dropout* noise mSDA cases. We find that DNFC can be seamlessly applied to mSDA with *Bernoulli dropout* noise (DNFC_{bd}) as DNFC_{bd} can enclose the two new elements efficiently. In contrast, mSDA with *Multinomial dropout* noise case has difficulty in handling the element of kernelization when adapted to DNFC framework.
- We conduct experimental studies to demonstrate the effectiveness of DNFC_{bd} compared to several state-of-the-art baselines. The effects of MMD and kernelization on the performance of DNFC_{bd} are also investigated. It is shown that DNFC_{bd} benefits from both elements, as well as their joint effects.

Chapter 5

Local Manifolds Information Transfer

In the previous two chapters, we have proposed two unsupervised domain adaptation frameworks, mSDA_{md} and DNFC. Both of them are deep feature learning methods. In this chapter, we focus on another domain adaptation strategy, subspace-based methods. Subspace-based methods (Gong et al., 2012; Zhou et al., 2014a; Long et al., 2014a; Sun et al., 2016) have been shown very effective for unsupervised domain adaptation. They transfer knowledge across domains by mapping the source and target data into a latent subspace. The underlying intuition is as follows. Although two domains may differ considerably in the original feature space, there exists a subspace in which the source and target domains have similar distributions.

Recent works (Wang and Mahadevan, 2011; Long et al., 2014b; Shu et al., 2014) show the tremendous benefits of preserving data geometric information (Cai et al., 2009) to unsupervised domain adaptation in the latent subspace learning. The geometric information is an important data local characteristic, assuming data points that are close to each other are likely to share a label (Long et al., 2014b). Such local information should be preserved in adaptation so that target label inference associates with the latent subspace learning. This is particularly significant in unsupervised domain adaptation where target labels are unavailable. Moreover, preserving data geometric information enables similar source instances to be closer in the latent subspace. This decreases the source training error, which is definitely beneficial for adaptation based on the theoretical analysis in (Ben-David et al., 2007).

Existing unsupervised domain adaptation works (Pan et al., 2011; Shu et al.,

2014; Long et al., 2014b) preserve the geometric information under a single manifold assumption (Roweis and Saul, 2000). That is, data points are sampled from a distribution supported by a single low-dimensional manifold embedded in the original feature space. However, in the real world, data is complex and may well be drawn from multiple low-dimensional manifolds Vidal (2011) (multiple manifolds assumption). For instance, an image may contain several objects, and different low-dimensional manifolds might be used to describe different objects. Another example is in natural language processing. A document may be regarded as the combination of several hidden topics, and each hidden topic corresponds to one low-dimensional manifold.

It makes major differences when knowledge is transferred under the above two different assumptions. Under the single manifold assumption, existing unsupervised domain adaptation methods (Pan et al., 2011; Shu et al., 2014; Long et al., 2014b) model the geometric information by an affinity matrix. This affinity matrix characterizes the similarities of p -nearest neighbors in the original feature space. The intuition is to enforce the data points that are close to each other in the original feature space to share a label. However, such an intuition is not valid under the multiple manifolds assumption. When data are embedded in multiple low-dimensional manifolds, which is a typical case in reality, it is the close data points in the same low-dimensional manifold that are likely to share a label. The following examples further help clarify the differences between these two assumptions.

Firstly, we provide an example in Figure 5.1. Data points are lying in a global space $\mathbb{R}^3$, and come from two manifolds: line $\mathbf{L}$ and plane $\mathbf{P}$. In $\mathbb{R}^3$, c is the nearest neighbor to a (a is on $\mathbf{L}$ and c is on $\mathbf{P}$), and they have different labels. The single manifold assumption does not hold in this case. However, when considering neighbors in the same manifold, the nearest neighbor of a is b (both a and b are on $\mathbf{L}$) that shares the same label with a . This justifies the multiple manifold assumption. We further provide a real-world example, which is sentiment analysis. Given a set of reviews of a product which are labeled by three sentiment classes (positive, neutral, and negative), manifolds may correspond to different aspects (latent topics) of the product. For example, manifold 1 may contain reviews that discuss if the product is worth the price; manifold 2 may be on the product functionality; manifold 3 may be on the appearance, etc. In each manifold, we have samples from all three

Figure 5.1: Multiple manifolds assumption example

classes. For the samples that fall within the same manifold (e.g., on the product appearance), it is natural to assume that positive samples are close to positive ones. In practice, we observe that datasets in unsupervised domain adaptation problems generally satisfy the multiple manifolds assumption, for instance datasets for multiple topic classification (Dai et al., 2007a) and datasets for multiple object classification (Gong et al., 2012; Long et al., 2013b). This motivates us to introduce the multiple manifolds assumption to unsupervised domain adaptation, which is proven to be significantly effective as evidenced in our experiments.

In this chapter, we propose a new subspace-based method called *local manifolds information transfer* (LMIT) for unsupervised domain adaptation. In LMIT, we explore the geometric information under the multiple manifolds assumption, i.e., the local multiple manifolds information embedded in the data of two domains. We take advantage of the data manifold neighborhood structure, which represents a data point by its neighbors in its own manifold, to capture the manifold information. The underlying rationale is the manifold self-expressiveness property of data, which states that each data point in a manifold can be efficiently represented as a linear or affine combination of other points in the same manifold. We propose to preserve the data manifold neighborhood structures in the latent subspace learning for adaptation. To do so, we develop a manifolds neighborhood preservation embedding (MNPE) algorithm. Furthermore, we also minimize the distribution discrepancy of two domains in the latent subspace learning. By doing so, LMIT not only preserves the local data multiple manifolds information, but also aligns the global domain distributions.

The rest of the chapter is organized as follows. Section 5.1 introduces the state-of-the-art subspace-based domain adaptation methods that take the geometric information preservation into account. Section 5.2 first describes a manifolds neighborhood preservation embedding algorithm, and then presents the details of our proposed LMIT approach. Section 5.3 conducts extensive experimental studies to show the performance of our proposed LMIT approach on various adaptation tasks. Finally, section 5.4 summarizes the chapter.

5.1 Related Works

Recently, studies on the preservation of the data geometric information in unsupervised domain adaptation attract increasing interests. (Pan et al., 2011) proposes an improved TCA, called SSTCA, that integrates the locality preservation, the label dependence, and the distribution discrepancy minimization. When preserving the locality, one existing technique called “linear locality embedding” is applied. In (Wang and Mahadevan, 2011), the data similarity topology is preserved through a full squared Euclidean similarity matrix in the feature mapping. (Long et al., 2014b) develops a Graph Co-regularized Transfer Learning (GTL) method combining the maximization of empirical likelihood and the preservation of the geometric information. The geometric information is modeled by a p -nearest neighbors similarity matrix using cosine similarity. (Shu et al., 2014) learns the locality preserving projections for both domains. Similar to GTL, a p -nearest neighbors similarity matrix using cosine similarity is constructed for the target domain. However, labels are used to define the geometric information in a supervised manner for the source domain. Although these methods have been shown effective, they are based on the single manifold assumption, which may jeopardize the adaptation performance due to the complex and diverse data in real-world unsupervised domain adaptation problems.

In this work, we propose an LMIT framework that integrates the multiple manifolds information preservation and the distribution discrepancy minimization. LMIT is built under the multiple manifolds assumption for unsupervised domain adaptation. To the best of our knowledge, this is the first work that considers the multiple manifolds information in unsupervised domain adaptation, and further combines it with the global distribution alignment for adaptation.

5.2 A Local Manifolds Information Transfer Approach

In unsupervised domain adaptation, we are given a source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ with n_s labeled instances, and a target domain $\mathcal{D}_t = \{\mathbf{x}_i^t\}_{i=1}^{n_t}$ with n_t unlabeled instances. The feature spaces of the two domains are identical, but the feature distributions are different. We represent the source data matrix as $\mathbf{X}_s \in \mathbb{R}^{d \times n_s}$ and its corresponding label vector as $\mathbf{y}_s \in \mathbb{R}^{1 \times n_s}$. Similarly, we denote the target data matrix as $\mathbf{X}_t \in \mathbb{R}^{d \times n_t}$, and the joint data matrix $\mathbf{X} = [\mathbf{X}_s, \mathbf{X}_t] \in \mathbb{R}^{d \times n}$ ($n = n_s + n_t$). We further assume $\mathbf{X}$ are sampled from multiple low dimensional manifolds $\{\mathcal{M}_i \in \mathbb{R}^{d_i}\}_{i=1}^N$ ($N \geq 1$) embedded in the original high-dimensional feature space $\mathcal{R} \in \mathbb{R}^d$ ($d_i < d$). Our objective is to learn a latent subspace $\mathcal{S} \in \mathbb{R}^k$, in which a transfer learner trained in the source domain can achieve a good performance for the prediction of target labels $\mathbf{y}_t$.

We propose an LMIT framework to learn a latent subspace which integrates the benefits of multiple manifolds information preservation and distribution discrepancy minimization by a feature mapping $\mathbf{W} \in \mathbb{R}^{d \times k}$. To explore the multiple manifolds information, we first introduce the following assumption to unsupervised domain adaptation:

- *Multiple Manifolds Assumption:* The intrinsic dimension of data is smaller than the dimension of the ambient space, and the data are drawn from multiple underlying low-dimensional manifolds. Data points that are close to each other in the same underlying low-dimensional manifold are more likely to share a label.

Under this assumption, we take advantage of data manifold neighborhood structures to encode the multiple manifolds information embedded in the data. The manifold neighborhood structure of a data point, which represents the data point by its neighbors in its own manifold, is based on the following manifold self-expressiveness property of data:

- *Manifold Self-expressiveness Property:* Each data point in a union of manifolds can be efficiently reconstructed by a combination of other points that are lying in the same manifold with itself.

We then propose to preserve the data manifold neighborhood structures in the latent subspace learning. Consequently, a manifolds neighborhood preservation embedding algorithm is developed. Furthermore, to align the source and target domains, we also fit the mapping to reduce the distribution discrepancies of the two domains in the latent subspace learning.

5.2.1 Manifolds Neighborhood Preservation

We propose a manifolds neighborhood preservation embedding (MNPE) algorithm that aims at preserving the local multiple manifolds information in the subspace learning. Given a set of data points, we are interested in finding an affinity matrix $\mathbf{A}$ specifying the relationships of the data points in the same manifold. Specifically, each data point is represented as a linear combination of its neighbors in the same manifold, and the combination coefficients are specified in $\mathbf{A}$. Those combination coefficients are enforced to be consistent in the subspace learning so that the neighborhood structure in each manifold is preserved. The algorithmic procedure is formally stated as follows.

- **Computing the affinity matrix $\mathbf{A}$:** In this step, we discover the manifold neighborhood structures of $\mathbf{X}$ by computing the affinity matrix $\mathbf{A}$. As stated above, for i -th row of $\mathbf{A}$, the nonzero elements correspond to the neighbors of $\mathbf{x}_i$ in the same manifold. In order to obtain such an affinity matrix, we could conduct manifolds clustering on $\mathbf{X}$, and then identify the neighbors of each data point within the same manifold. To avoid such trivial two-phase procedure, we propose to use sparse representation (Elhamifar and Vidal, 2009). The key rationale is that a sparse representation of a data point ideally corresponds to a combination of a few points from its own manifold. Specifically, each data point is written as a sparse linear combination of all other data points, and the sparse coefficients are found by solving the problem:

$$\begin{aligned} \min_{\{A_{ij}\}} \sum_i \|\mathbf{x}_i - \sum_j A_{ij} \mathbf{x}_j\|^2 + \mu \sum_{i,j} |A_{ij}| \\ s.t. \quad \text{diag}(\mathbf{A}) = \mathbf{0}, \end{aligned} \quad (5.1)$$

where $\text{diag}(\mathbf{A})$ is a vector whose elements are the diagonal entries of $\mathbf{A}$. According to (Elhamifar and Vidal, 2009; Vidal, 2011), there exists a sparse solution $\mathbf{A}$, whose nonzero entries of i -th row correspond to data points from

the same manifold with $\mathbf{x}_i$. Thus, the affinity matrix $\mathbf{A}$ obtained in (5.1) encodes the underlying multiple manifolds information. More importantly, for each $\mathbf{x}_i$, it automatically gives the optimal number of neighbors lying in the same manifold with $\mathbf{x}_i$, as well as the corresponding combination coefficients. The optimization problem (7.1) is a standard l_1 -norm optimization problem (Elhamifar and Vidal, 2009), and we use the optimization package provided in (Tierney et al., 2015) to solve it.

- **Computing the projections:** In this step, we compute projections to preserve the manifold neighborhood structures of $\mathbf{X}$ in the latent subspace learning. We formulate a problem of mapping the original data points to a line. Meanwhile, the mapped data points are enforced to be represented by a linear combination of its neighbors with the combination coefficients A_{ij} . We denote such a map by $\mathbf{u} = (u_1, u_2, \dots, u_n)$. According to (Roweis and Saul, 2000), one valid principle to obtain a good map is to solve the following optimization problem:

$$\min_{\mathbf{u}} \sum_i (u_i - \sum_j A_{ij} u_j)^2. \quad (5.2)$$

Note that, compared to (5.1), optimization problem (5.2) optimizes the coordinates $\mathbf{u}$ with the combination coefficients A_{ij} fixed. We denote:

$$v_i = u_i - \sum_j A_{ij} u_j,$$

and thus we have the vector form:

$$\mathbf{v} = \mathbf{u} - \mathbf{uA}.$$

Considering a linear mapping $\mathbf{w} \in \mathbb{R}^{d \times 1}$, that is $\mathbf{u} = \mathbf{w}^\top \mathbf{X}$, the objective function can be rewritten as:

$$\begin{aligned} & \sum_i (u_i - \sum_j A_{ij} u_j)^2 \\ &= \sum_i (v_i)^2 = \mathbf{v} \mathbf{v}^\top \\ &= (\mathbf{u} - \mathbf{uA})(\mathbf{u} - \mathbf{uA})^\top \\ &= \mathbf{w}^\top \mathbf{X} (\mathbf{I} - \mathbf{A}) (\mathbf{I} - \mathbf{A})^\top \mathbf{X}^\top \mathbf{w}. \end{aligned}$$

To exclude arbitrary scaling factors in the projection, we add a constraint:

$$\mathbf{u} \mathbf{u}^\top = 1 \implies \mathbf{w}^\top \mathbf{X} \mathbf{X}^\top \mathbf{w} = 1.$$

Thus, we obtained the optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & tr(\mathbf{w}^\top \mathbf{X} \mathbf{N} \mathbf{X}^\top \mathbf{w}) \\ s.t. \quad & \mathbf{w}^\top \mathbf{X} \mathbf{X}^\top \mathbf{w} = 1, \end{aligned} \quad (5.3)$$

where $\mathbf{N} = (\mathbf{I} - \mathbf{A})(\mathbf{I} - \mathbf{A})^\top$. This is a generalized eigenvector problem. We can obtain $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_k$ corresponding to the k smallest eigenvalues. Then, we have the mapping matrix:

$$\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_k).$$

Note that the proposed MNPE is a new dimensionality reduction algorithm, which is independent of domain adaptation. It can be applied to alleviate the curse of dimensionality, but we use it for domain adaptation in this paper. The flow of MNPE is similar to some existing dimensionality reduction methods, for instance Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Locality Preservation Projection (He and Niyogi, 2004) (LPP), and Neighborhood Preserving Embedding (NPE) (He et al., 2005). However, a major difference of MNPE with the others is that the affinity matrix in MNPE encodes the multiple low-dimensional manifolds information, while the others are based on the single manifold assumption. Moreover, when computing the affinity matrix, the other methods need to construct a graph either using p -nearest neighbors or using ϵ -neighborhood. In contrast, MNPE computes $\mathbf{A}$ without having to specify p or ϵ by solving (5.1).

5.2.2 Local Manifolds Information Transfer Framework

So far, we have introduced the manifolds neighborhood preservation embedding. Next, we present our LMIT that incorporates the benefits of MNPE to unsupervised domain adaptation, and simultaneously reduces the distribution discrepancies.

We design the final objective function of LMIT as follows:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \underbrace{\alpha tr(\mathbf{W}^\top \mathbf{X} \mathbf{N} \mathbf{X}^\top \mathbf{W})}_{Geometric} + \underbrace{\beta tr(\mathbf{W}^\top \mathbf{X} \mathbf{M} \mathbf{X}^\top \mathbf{W})}_{Discrepancy} \\ & + \underbrace{tr(\mathbf{W}^\top \mathbf{W})}_{Regularization} \\ s.t. \quad & \mathbf{W}^\top \mathbf{X} \mathbf{X}^\top \mathbf{W} = \mathbf{I}, \end{aligned} \quad (5.4)$$

where α and β are hyper-parameters, $\mathbf{N} = (\mathbf{I} - \mathbf{A})^\top(\mathbf{I} - \mathbf{A})$ with $\mathbf{A}$ calculated from (5.1), and $\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix}$ with $\mathbf{M}_{11} = \frac{1}{n_s^2} \mathbf{1}_{n_s}^\top \mathbf{1}_{n_s}$, $\mathbf{M}_{12} = \mathbf{M}_{21}^\top = -\frac{1}{n_s n_t} \mathbf{1}_{n_s}^\top \mathbf{1}_{n_t}$ and $\mathbf{M}_{22} = \frac{1}{n_t^2} \mathbf{1}_{n_t}^\top \mathbf{1}_{n_t}$ ($\mathbf{1}$ is a row vector of all ones). It can be seen that the objective function consists of three terms: the discrepancy term, the geometric term, and the regularization term. Specifically, we adopt the objective of MNPE as the geometric to preserve the multiple manifolds information, and use MMD (Borgwardt et al., 2006) as the discrepancy term to measure the distribution discrepancy of domains. The regularization term is used to avoid overfitting. With these terms, we jointly learn a feature mapping $\mathbf{W}$ that maps data from the original space $\mathcal{R} \in \mathbb{R}^d$ to the latent subspace $\mathcal{S} \in \mathbb{R}^k$.

To solve the optimization problem (5.4), we rewrite it as:

$$\begin{aligned} \min_{\mathbf{W}} \quad & tr(\mathbf{W}^\top(\mathbf{I} + \mathbf{X}(\beta\mathbf{M} + \alpha\mathbf{N})\mathbf{X}^\top)\mathbf{W}) \\ s.t. \quad & \mathbf{W}^\top\mathbf{X}\mathbf{X}^\top\mathbf{W} = \mathbf{I}. \end{aligned} \quad (5.5)$$

The lagrangian function of (5.5) is:

$$\begin{aligned} L = & tr(\mathbf{W}^\top(\mathbf{I} + \mathbf{X}(\beta\mathbf{M} + \alpha\mathbf{N})\mathbf{X}^\top)\mathbf{W}) \\ & + tr((\mathbf{I} - \mathbf{W}^\top\mathbf{X}\mathbf{X}^\top\mathbf{W})\Psi), \end{aligned}$$

where $\Psi = \mathbf{diag}([\psi_1, \psi_2, \dots, \psi_k]) \in \mathbb{R}^{k \times k}$ are the lagrange multipliers. By setting the derivative of L with respect to $\mathbf{W}$ equal to zero, we obtain:

$$(\mathbf{I} + \mathbf{X}(\beta\mathbf{M} + \alpha\mathbf{N})\mathbf{X}^\top)\mathbf{W} = \mathbf{X}\mathbf{X}^\top\mathbf{W}\Psi \quad (5.6)$$

This is a generalized eigenvalue problem. The optimal solution $\mathbf{W}$ can be derived by solving the k smallest eigenvectors from (5.6). After obtaining the optimal $\mathbf{W}$, we map the source and target data to $\mathcal{S}$. Thus, we have:

$$\mathbf{X}'_s = \mathbf{W}^\top\mathbf{X}_s, \quad \mathbf{X}'_t = \mathbf{W}^\top\mathbf{X}_t \quad (5.7)$$

We then train a model from $\mathbf{X}'_s$ and $\mathbf{y}_s$ in the source domain, and apply it to predict $\mathbf{y}_t$ for $\mathbf{X}'_t$ in the target domain. For out-of-sample target data, we can use $\mathbf{W}$ to map it to $\mathcal{S}$. The prediction can then be done by the trained model. The proposed LMIT is summarized in Algorithm 5.1.

Algorithm 5.1 Local Manifolds Information Transfer

Input: Source data matrix $\mathbf{X}_s$, source labels $\mathbf{y}_s$, and target data matrix $\mathbf{X}_t$.

Procedure:

1. Solve optimization problem (5.1) and obtain $\mathbf{A}$;
2. Plug $\mathbf{A}$ in (5.4) and solve (5.6) to obtain $\mathbf{W}$;
3. Calculate $\mathbf{X}'_s$ and $\mathbf{X}'_t$ using Eqs. (5.7);
4. Train a model $\mathcal{M}$ from $\mathbf{X}'_s$ and $\mathbf{y}_s$;
5. Predict $\mathbf{y}_t$ for $\mathbf{X}'_t$ using $\mathcal{M}$.

Output: $\mathbf{y}_t$.

5.2.3 Discussion

In LMIT, the affinity matrix $\mathbf{A}$ encodes the underlying multiple manifolds information. However, instead of explicitly computing the multiple manifolds (obtaining multiple manifolds is not our goal) using $\mathbf{A}$ as many spectral-clustering methods (Vidal, 2011) do, we directly use the local neighborhood information from multiple manifolds encoded in $\mathbf{A}$ to learn a latent subspace for adaptation purpose (This is our goal!). Note that although we eventually map to “a single latent subspace”, this “single latent subspace” does not mean “a single manifold”. In fact, we are using “multiple manifolds” because the local neighborhood information from different manifolds is encoded in $\mathbf{A}$ and we use $\mathbf{A}$ in the learning of the latent subspace.

Moreover, regarding the objective function (5.6), we find that it has similar form with the objective function in many existing methods, for instance, TCA (Pan et al., 2009), SSTCA (Pan et al., 2011), JDA (Long et al., 2013b), TJM (Long et al., 2014a), CCLS (Ming Harry Hsu et al., 2015), COR (Sun et al., 2016), DcTC (Tsai et al., 2016). This motivates us to summarize a unified general spectral embedding framework for unsupervised domain adaptation with the following objective function:

$$\begin{aligned} \min_{\mathbf{W}} \quad & tr(\mathbf{W}^T \mathbf{X} \mathbf{U} \mathbf{X}^T \mathbf{W}) \\ s.t. \quad & \mathbf{W}^T \mathbf{X} \mathbf{V} \mathbf{X}^T \mathbf{W} = \mathbf{I}. \end{aligned} \tag{5.8}$$

In the objective function (5.8), the essence is using different $\mathbf{U}$ and $\mathbf{V}$ to encode different data characteristics in adaptation. More specifically, TCA only minimizes the distribution discrepancy; SSTCA improves TCA by incorporating the locality preservation and the label dependence; JDA takes both the marginal and conditional

distribution discrepancies into account; TJM not only aligns features but also selects the instances; CCLS integrates JDA and TCA into one unified framework; COR is a specific case of TCA with the exploration of data second-order statistic; DcTC is also a hybrid approach of instance selection and joint feature space learning. The form of $\mathbf{U}$ and $\mathbf{V}$ for these methods can be found in the corresponding original papers. Our LMIT also follows the above general spectral embedding framework. However, compared to the methods mentioned above, our LMIT is based on the multiple manifolds assumption. Such assumption is a more realistic assumption to fit the complex and diverse real-world data, which may lead to a better adaptation performance. The following experiments verify the effectiveness of our LMIT.

5.3 Experiments

In this section, we evaluate our LMIT on three real-world domain adaptation benchmark datasets. Two of them are *20-Newsgroups* and *COIL1-COIL2*, which have been introduced in the experimental studies of Chapter 3. We follow the same experimental setting with respect to these two datasets stated in Chapter 3. The last one is *Office-Caltech10* (Gong et al., 2012).

Office-Caltech10 (Gong et al., 2012) dataset is a popular benchmark image dataset for domain adaptation. The task is object recognition. Images are from four domains: C (Caltech-256), A (Amazon), W (Webcam), and D (DSLR). These four domains share the same 10 objects, but have different data size. In our experiment, we construct four domain adaptation tasks where the source and target domains have comparable data size, namely C-A, A-C, W-D, and D-W. Following (Gong et al., 2012), We use the SURF features.

We compare our LMIT with various subspace-based unsupervised domain adaptation methods including: GFK (Gong et al., 2012), SA (Fernando et al., 2013), TSC (Long et al., 2013a), GTL (Long et al., 2014b), TJM (Long et al., 2014a), SSTCA (Pan et al., 2011), and COR (Sun et al., 2016). Note that since the target labels are not available in unsupervised domain adaptation, we use SSTCA as the integration of the locality preservation projection and MMD, without considering the label dependence matrix. For baselines that use the kernel trick, we exploit the default “rbf” kernel function. As suggested by (Gong et al., 2012; Long et al.,

Table 5.1: Results (%) on three benchmark datasets

Task	NN	GFK	SA	TSC	GTL	TJM	SSTCA	COR	LMIT
CR-1	56.90	59.64	61.18	50.73	57.84	59.73	62.72	52.96	65.04
CR-2	51.23	58.27	61.41	50.72	55.13	63.44	63.19	60.31	66.33
CS-1	58.77	62.19	68.35	50.81	60.48	66.55	66.55	54.75	69.63
CS-2	52.21	61.24	64.40	50.51	59.45	63.20	60.48	51.87	68.06
CT-1	51.10	74.16	77.32	55.60	52.82	78.47	76.84	54.64	79.71
CT-2	54.99	71.29	76.24	60.46	61.83	79.18	70.35	52.47	80.23
RS-1	64.51	64.17	65.60	50.13	57.86	63.08	68.29	56.27	66.53
RS-2	62.48	66.27	68.38	50.08	53.71	68.97	69.81	59.36	64.92
RT-1	50.57	62.88	65.44	56.06	59.19	68.75	58.81	52.08	73.77
RT-2	55.54	59.75	66.22	61.40	57.60	70.46	56.88	66.22	71.05
ST-1	51.37	64.24	65.56	56.10	54.12	65.85	67.46	65.94	68.40
ST-2	58.84	63.19	69.70	61.12	62.15	70.53	65.77	50.57	70.11
Average	55.71	63.94	67.48	54.46	57.68	68.27	65.60	56.45	70.32
A-C	16.83	36.51	38.91	33.30	31.52	38.47	23.51	21.37	42.48
C-A	16.28	41.23	47.08	47.70	42.28	47.70	31.00	24.74	47.91
D-W	23.73	69.15	80.00	45.76	67.12	82.71	52.54	77.29	86.78
W-D	29.30	79.62	82.80	18.47	70.70	90.45	54.14	73.89	92.36
Average	21.53	56.63	62.20	36.31	52.90	64.83	40.30	49.32	67.38
COIL ₁₂	80.87	78.19	60.00	80.10	76.53	79.17	83.61	85.69	89.17
COIL ₂₁	81.53	79.44	47.91	80.35	73.33	75.28	84.03	82.78	88.33
Average	81.25	78.82	53.96	80.23	74.93	77.22	83.82	84.23	88.75
Average-3	50.95	63.97	64.81	53.30	58.54	68.50	62.00	57.96	71.71

2014a), we utilize the Nearest Neighbor (NN) classifier as the base classifier since it does not require tuning cross-validation parameters. Moreover, in subspace-based UDA methods, the intrinsic subspace dimension k is an important hyperparameter to be determined. Since it is heuristic to set a specific but empirical k for each domain adaptation task, we utilize Maximum Likelihood Estimator (Levina and Bickel, 2004) to determine k in all the methods for fair comparison. For GTL and SSTCA, we use the default p specified by authors.

5.3.1 Experimental Results

The comparison results of LMIT with other baselines on the three datasets are shown in Table 5.1. We also list the average result of the tasks in each dataset, and the average result of all the tasks. The winners are highlighted in bold.

From Table 5.1, we can see that our proposed LMIT outperforms the other baselines on the average performance of all the tasks. Compared to TJM, which is the best baseline in terms of the average performance, LMIT achieves 3.21% improvements. More specifically, the average accuracy of LMIT on the three datasets are 70.32%, 67.38%, and 88.75%. The performance improvements are 2.05%, 2.55%, and 4.52% in comparison with the champion baselines on the three datasets (TJM in 20-Newsgroups and Office-Caltech10, COR in COIL1*vs*COIL2), respectively. Our LMIT sweeps the board in the Office-Caltech10 and COIL1-COIL2 datasets, and prevails in 9 of 12 tasks in the 20-Newsgroups dataset. These all show the superiority of LMIT to the other baselines.

One interesting observation is that our AGLT consistently gains the positive adaptation performance on all the tasks. All other baselines have negative adaptation in some specific tasks, except for SSTCA. This may be because both AGLT and SSTCA minimize the distribution discrepancy and preserve the geometric structure simultaneously. The others either ignore the geometric structure preservation or fail to explicitly reduce the distribution discrepancy. Moreover, our AGLT beats SSTCA with 9.71% average accuracy improvements obtained.

5.3.2 Multiple Manifolds Neighbors

In this section, we verify the effectiveness of LMIT by inspecting the neighbor-class structure in different spaces. For each target instance, we discover its p -nearest source neighbors ($p = 1, 2, \dots, 10$) in the original feature space $\mathcal{R}$, in the low-dimensional manifold $\mathcal{M}_i$ that the target instance lies in, and in the learned latent subspace $\mathcal{S}$. Note that in $\mathcal{M}_i$, we extract the source neighbors that also lie in $\mathcal{M}_i$. If a source neighbor has the same class with the target instance, we weight it as 1, otherwise 0. Thus, we obtain 3 neighbor-class matrices. We utilize D-W as an example to analyze these 3 matrices. The other tasks have similar trends.

Figure 5.2 shows the heatmaps of the 3 matrices. The y-axis is the index of the

Figure 5.2: Neighbor-class structure in three spaces.

target data, and the x-axis is the p -th nearest source neighbor. From Figure 5.2(a), we observe that there are many white spaces, which means most of the source neighbors have different class with the corresponding target instance. Thus, the prediction performance will be unsatisfactory. However, we observe that the black spaces are becoming dense in Figure 5.2(b), and much denser in Figure 5.2(c). This leads to a higher chance of correct predictions for those target instances. Basically, the denser the black space of a matrix is, the higher prediction accuracy can be obtained on the target domain. Table 5.2 lists the number of the black spaces for each p in the three matrices, and some statistical results. It shows that the neighbor density order of the three heatmaps is always: (c) is the best, followed by (b) and lastly (a). The accuracy is positively proportional to the density. More specifically, from the fact that (b) is better than (a), we can see that our multiple manifolds assumption holds, which means the closer points in the same manifold is more likely

Table 5.2: Neighborhood statistics

Space	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$	$p=8$	$p=9$	$p=10$	Sum	Density	Acc
(a)	92	69	59	79	56	63	55	56	51	60	640	21.69%	23.73%
(b)	138	104	91	87	84	94	88	80	74	67	907	30.75%	46.10%
(c)	252	230	181	158	117	110	100	98	83	84	1413	47.90%	86.78%

to share the same label than those in the original feature space. Therefore, working under the multiple manifolds assumption may be more effective for adaptation. By further incorporating MMD, which is LMIT, the two domains are brought to be closer. This can be verified by the fact that (c) that is much denser than (a).

5.3.3 Geometric Information Preservation

In this section, we further investigate the adaptation performance by using different ways to preserve the geometric information. Except for SSTCA, we also consider another baseline that combines Neighborhood Preservation Embedding (NPE) with MMD, called neighborhood structure transfer (NST). Note that SSTCA preserves the similarities of the p -nearest neighbors, while NST preserves the neighborhood structure using combination coefficients of neighbors. These two methods still work under the single manifold assumption. We test on the three datasets stated above, and show the results in Table 5.3.

The comparison results in Table 5.3 show that LMIT is the best, followed by NST, SSTCA, and lastly NN. Specifically, we observe that LMIT achieves much better average result than NST (with 6.95% improvements) and SSTCA (with %9.71 improvements). This again indicates the effectiveness of exploring the multiple manifolds information in adaptation. Moreover, NST attains comparable result with SSTCA. The slight improvement (2.76%) of NST over SSTCA indicates that preserving the neighborhood structure may be more suitable than preserving the similarities of p -nearest neighbors in the various adaptation tasks. This is because the selection of similarity metric in SSTCA is ad-hoc, and the best similarity metric for unsupervised domain adaptation may differ considerably among the various real-world datasets. NST avoids the problematic similarity metric selection, and thus may be more robust for various real-world unsupervised domain adaptation applications. Moreover, we also observe that LMIT achieves much better average

Table 5.3: Comparison results on the neighborhood structure preservation

Task	NN	SSTCA	NST	LMIT
CR-1	56.90	62.72	56.90	65.04
CR-2	51.23	63.19	68.45	66.33
CS-1	58.77	66.55	57.06	69.63
CS-2	52.21	60.48	58.18	68.06
CT-1	51.10	76.84	66.51	79.71
CT-2	54.99	70.35	67.19	80.23
RS-1	64.51	68.29	60.89	66.53
RS-2	62.48	69.81	70.91	64.92
RT-1	50.57	58.81	58.90	73.77
RT-2	55.54	56.88	61.19	71.05
ST-1	51.37	67.46	63.95	68.40
ST-2	58.84	65.77	62.98	70.11
Average	55.71	65.60	62.76	70.32
A-C	16.83	23.51	39.63	42.48
C-A	16.28	31.00	44.99	47.91
D-W	23.73	52.54	87.80	86.78
W-D	29.30	54.14	91.72	92.36
Average	21.54	40.30	66.03	67.38
COIL ₁₂	80.97	83.61	76.11	89.17
COIL ₂₁	81.53	84.03	72.36	88.33
Average	81.25	83.82	74.24	88.75
Average-3	50.95	62.00	64.76	71.71

result than NST with 6.95% improvements, it again indicates the effectiveness of exploring the multiple manifolds information in LMIT.

5.3.4 Sensitivity Analysis

We analyze the effects of the geometric term and the discrepancy term on the adaptation performance by conducting the sensitivity analysis on the parameters α and

Figure 5.3: Sensitivity analysis

β . We show in Figure 5.3 the average results of all the tasks with α and β varying in the different ranges. As shown in Figure 5.3, only one term alone ($\alpha = 0$ or $\beta = 0$) or dominated cannot give the best result, but still achieves improvements compared to the result of no-transfer, 50.95%. This illustrates that either of terms can bring benefits to the adaptation task individually. We also observe that the best result appears when $\alpha \in [0.01, 0.1]$ and $\beta \in [500, 1000]$. It is a balancing result of these two terms since they interact with each other. In our experiments, we empirically set α as 0.01 and β as 1000 for all the tasks.

5.4 Summary

We summarize the contributions of this chapter as follows:

- We introduce a multiple manifolds assumption to unsupervised domain adaptation, and then design a manifold neighborhood preservation embedding algorithm to achieve the preservation of multiple manifolds information in the subspace learning;
- we propose an LMIT framework that conducts local manifolds information preservation and global distribution discrepancy minimization simultaneously. A generalized eigenvalue problem is formulated so that the optimal solution of LMIT framework can be efficiently obtained.

- we conduct extensive experiments on various real-world datasets to demonstrate the effectiveness of our LMIT. We also analyze the benefits of preserving multiple manifolds information to adaptation by demonstrating the neighborhood-class structure of different spaces.

Chapter 6

Domain Specific Feature Transfer

So far, our focus is on the supervised domain adaptation problems, which equip homogeneous feature spaces between different domains. In this chapter, we are dealing with heterogeneous domain adaptation problems. Compared to homogeneous domain adaptation, heterogeneous domain adaptation is more difficult to tackle due to the heterogeneity of feature spaces. Since no correspondence exists between the feature spaces of different domains, supplementary information, such as labels (Wang and Mahadevan, 2011), translators (Dai et al., 2008), or correspondence instances (Zhou et al., 2014a), is needed to link up different domains. If such information is unavailable, which is usually the case in reality, heterogeneous domain adaptation is intractable.

In practice, there are many cases where the feature spaces of two domains are not completely heterogeneous. Two domains share some common features, while each domain also has its own domain specific features. In sentiment analysis, for instance, reviews of different products share some common sentiment words, e.g., “good”, “bad”, etc. In addition, different types of products have their specific sentiment words, e.g., “comprehensive” for books, “noisy” for electronic products, etc. Similarly in biological data, different species may share common traits or characteristics but also have their own biological features. This is a special case of heterogeneous domain adaptation, with the existence of both common features and domain specific features. We name this kind of problems as *hybrid domain adaptation* problem. To the best of our knowledge, this is the first work of studying such a problem setting, in spite of its popularity in real applications.

In hybrid domain adaptation, the main challenge lies in the heterogeneity of

domain specific features. Existing heterogeneous domain adaptation methods cannot be applied because of the lack of supplementary information. Alternatively, we could convert the hybrid problem to a homogeneous one. Two different strategies can be used to do so: “*cut-off*” and “*fill-up*”. *Cut-off* is a straightforward way to align the feature spaces by discarding the domain specific features and retaining only the common features. *Fill-up* is another alternative to eliminate the feature heterogeneities. The idea is to expand the feature space by concatenating common features and specific features of the two domains, after which the feature spaces become homogeneous.

In general, the *fill-up* strategy is preferred to the *cut-off* one. This is because *fill-up* preserves the domain specific features, which typically encode the discriminative information and are thus critical for the learning task. Such information is particularly essential in unsupervised domain adaptation, in which the target domain has no labeled data to guide the learning. The current common way of doing *fill-up* is filling in zeros to those specific features appearing in one domain but not the other, called F_Zero. In the text field, for instance, the vocabulary lexicons of different domains are often merged, with the words specific to one domain filled up with zeros for other domains (Blitzer et al., 2006, 2007; Glorot et al., 2011; Wu and Huang, 2016). However, our analyses demonstrate that F_Zero does not benefit from using domain specific features in the adaptation, and it is actually equivalent to the “*cut-off*” strategy that only uses the common features. We show the evidence and discuss more details in the section 6.4.3.

In this chapter, we propose a general framework called “Domain Specific Feature Transfer” (DSFT) for hybrid domain adaptation, which can effectively utilize the domain specific features. DSFT takes the *fill-up* strategy in a way that it benefits the adaptation. The *fill-up* in the two domains is bridged by the common features, and meanwhile the discriminative power embedded in the specific features is captured and transferred from one domain to the other. More specifically, we learn a feature mapping between the common features and the specific features in each domain. We then apply the learned mappings across domains to do the *fill-up*. During the learning, we enforce the distribution discrepancy between the two domains after the *fill-up* to be minimized. By doing so, the *fill-up* in DSFT not only transfers the knowledge across domains, but also brings the two domains closer for better

adaptation. The learning of the mappings, the *fill-up*, as well as the minimization of the distribution discrepancy, are incorporated into one unified framework. With the learned mappings, an expanded homogeneous feature space is constructed, and DSFT then trains a model in the source domain and makes predictions in the target domain in this new space. In this work, we propose two instantiations of DSFT by considering both the linear and nonlinear cases. A closed-form solution to each case is derived.

Though the idea of DSFT may be simple, its underlying mechanism is very effective. First, the *fill-up* by the learned mappings is essentially to reweight the common features by exploiting the domain specific features. By reweighting, the discriminative power embedded in the domain specific features is brought over to the common features, and the two domains are brought to be closer. Second, the *fill-up* and the minimization of distribution discrepancy enable DSFT to provide a better feature representation for adaptation. The constructed universal features have smaller domain divergence and source training errors. These two points are empirically verified in our experiments.

The rest of the chapter is organized as follows. Section 6.1 gives related works on heterogeneous domain adaptation methods, and discussions on the links of hybrid domain adaptation with both homogeneous and heterogenous ones. Section 6.2 describes our proposed general framework DSFT. Section 6.3 instantiates a linear case and a nonlinear case for our proposed DSFT. Section 6.4 presents the empirical study and comparison. Finally, section 6.5 summarizes the chapter.

6.1 Related Works

Heterogeneous domain adaptation aims to transfer knowledge across domains with different feature spaces. Since there is no correspondence between feature spaces, supplementary information is involved to link up domains. In (Shi et al., 2010; Kulis et al., 2011) and (Wang and Mahadevan, 2011), source and target labels are used to align the feature spaces. Instances with the same labels are mapped to be closer, while instances with different labels are mapped to be far away from each other. (Dai et al., 2008) proposed a translator to obtain the correspondence between two domains. In (Zhou et al., 2014a), correspondence instances were used to learn the

relationship between the source and target domains. In (Li et al., 2014), Li et al. proposed a heterogeneous feature augmentation method. Target labels are used to connect the two domains in the augmented space.

The hybrid domain adaptation studied in this chapter is a new problem setting. Compared to the well-studied homogeneous domain adaptation and heterogeneous domain adaptation, we emphasize the following two points. First, since it is straightforward to convert hybrid domain adaptation to homogeneous DA by the “*cut-off*” and “*fill-up*” strategies, we compare our DSFT proposed for hybrid domain adaptation with an important class of baselines that apply the homogeneous domain adaptation methods after the “*cut-off*” or “*fill-up*”. Second, though hybrid domain adaptation can be treated as a special case of heterogeneous domain adaptation, the existence of common features enables it to be solved without supplementary information, which makes it incomparable to existing heterogeneous methods.

6.2 A General Domain Specific Feature Transfer Framework

In this section, we first give a detailed description of the hybrid domain adaptation problem setting. We then introduce the proposed DSFT framework. Herein, we only focus on the general structure of the DSFT framework, and leave the technical details of two DSFT framework instantiations in the next section.

6.2.1 Problem Statement

In hybrid domain adaptation problem, we are given labeled data in the source domain and unlabeled data in the target domain. The two domains share a set of common input features, while each of them has its own domain specific input features. Let $\mathbf{X}_s = [\mathbf{S}_c; \mathbf{S}_s] \in \mathbb{R}^{(c+s) \times n}$ denote the source data matrix with n samples and $\mathbf{X}_t = [\mathbf{T}_c; \mathbf{T}_t] \in \mathbb{R}^{(c+t) \times m}$ denote the target data matrix with m samples. Herein, $\mathbf{S}_c \in \mathbb{R}^{c \times n}$ and $\mathbf{T}_c \in \mathbb{R}^{c \times m}$ are the feature matrices with c common features, $\mathbf{S}_s \in \mathbb{R}^{s \times n}$ is the feature matrix of s source specific features, and $\mathbf{T}_t \in \mathbb{R}^{t \times m}$ is the feature matrix with t target specific features. The labels of $\mathbf{X}_s$, denoted as $\mathbf{Y}_s$, are available. Our task is to predict the labels $\mathbf{Y}_t$ in the target domain, by leveraging

Figure 6.1: Framework of DSFT

the information from the source domain.

6.2.2 Domain Specific Feature Transfer

In this section, we describe how the domain specific features are utilized to facilitate the domain adaptation.

Figure 6.1 shows the framework of our DSFT. The *fill-up* and adaptation is accomplished by using the common features in the two domains as the bridge. Specifically, we learn the feature mappings Ψ_s and Ψ_t as the translations to link up the common features with the domain specific features, respectively for each domain. The optimization problems are:

$$\min_{\Psi_s} \|\Psi_s(\mathbf{S}_c) - \mathbf{S}_s\|_F^2, \quad \min_{\Psi_t} \|\Psi_t(\mathbf{T}_c) - \mathbf{T}_t\|_F^2. \quad (6.1)$$

After obtaining the translations, we cross-use them to do the *fill-up*. Precisely, we apply the feature mapping Ψ_t (i.e., the translation obtained from the target domain) to the common features of the source domain. This gives a matrix that encodes the target-specific feature expressions for the source samples by using the common-specific rule in the target domain. We denote this matrix as $\mathbf{S}_a \in \mathbb{R}^{t \times n}$, the adapted target-specific feature matrix. We do the corresponding translation for the target domain and obtain the adapted matrix $\mathbf{T}_a \in \mathbb{R}^{s \times m}$. The computation of $\mathbf{S}_a$ and $\mathbf{T}_a$ is given below:

$$\mathbf{S}_a = \Psi_t(\mathbf{S}_c), \quad \mathbf{T}_a = \Psi_s(\mathbf{T}_c). \quad (6.2)$$

We then utilize both the original features and the *fill-up* adapted features for adaptation. This gives a homogeneous feature space as follows:

$$\mathbf{X}_{s,h} = [\mathbf{S}_c; \mathbf{S}_s; \mathbf{S}_a], \quad \mathbf{X}_{t,h} = [\mathbf{T}_c; \mathbf{T}_a; \mathbf{T}_t]. \quad (6.3)$$

6.2.3 Distribution Discrepancy Minimization

So far the translation between two domains is performed individually for each domain, as indicated in Eq. (6.1). However, our objective is to adapt the source domain to the target domain so that it helps the prediction task in the latter. For this purpose, we want the marginal distributions in the two domains to be as close as possible, i.e., $\mathbf{X}_{s,h}$ and $\mathbf{X}_{t,h}$, to have a low distribution discrepancy. More specifically, we want to choose among different possible translations those with better domain adaptation capability.

To do so, we propose to quantify the distribution discrepancy between $\mathbf{X}_{s,h}$ and $\mathbf{X}_{t,h}$ with a divergence metric $Dist(\mathbf{X}_{s,h}, \mathbf{X}_{t,h})$, and incorporate it into the optimization. To minimize the distribution discrepancy, we need to minimize $Dist(\mathbf{X}_{s,h}, \mathbf{X}_{t,h})$ in the feature mappings learning:

$$\min_{\Psi_s, \Psi_t} Dist(\mathbf{X}_{s,h}, \mathbf{X}_{t,h}). \quad (6.4)$$

6.2.4 Final DSFT framework

By combining (6.1) and (6.4), we obtain the final objective function of DSFT with three terms:

$$\begin{aligned} \min_{\Psi_s, \Psi_t} & \underbrace{\|\Psi_s(\mathbf{S}_c) - \mathbf{S}_s\|_F^2 + \|\Psi_t(\mathbf{T}_c) - \mathbf{T}_t\|_F^2}_{Translation} \\ & + \underbrace{\alpha Dist(\mathbf{X}_{s,h}, \mathbf{X}_{t,h})}_{Discrepancy} + \underbrace{\beta(\|\Psi_s\|_F^2 + \|\Psi_t\|_F^2)}_{Regularization}, \end{aligned} \quad (6.5)$$

where the regularization term is used to control the complexity of the mappings, and α and β are hyper-parameters.

After obtaining the optimal Ψ_s and Ψ_t by solving (6.5), we can compute the data matrices $\mathbf{X}_{s,h}$ and $\mathbf{X}_{t,h}$ by Eq. (6.3). We then train a prediction model from $\mathbf{X}_{s,h}$ and $\mathbf{Y}_s$ in the source domain, and apply it to predict the labels $\mathbf{Y}_t$ for $\mathbf{X}_{t,h}$ in the target domain. Our DSFT framework is applicable for both classification and regression tasks. Existing classification/regression methods can be used for training the models. For out-of-sample target data, we can use Ψ_s to do the *fill-up* and form the homogenous feature space. The prediction can then be done by the trained model. The proposed DSFT is summarized in Algorithm 6.1.

Algorithm 6.1 Domain Specific Feature Transfer

Input: Source data matrix $\mathbf{X}_s$, source label $\mathbf{Y}_s$, target data matrix $\mathbf{X}_t$

Procedure:

1. Compute Ψ_s and Ψ_t by solving objective function (6.5);
2. Compute $\mathbf{X}_{s,h}$ and $\mathbf{X}_{t,h}$ by Eq. (6.3)
3. Train a model $\mathcal{M}$ from $\mathbf{X}_{s,h}$ and $\mathbf{Y}_s$;
4. Predict $\mathbf{Y}_t$ for $\mathbf{X}_{t,h}$ using $\mathcal{M}$.

Output: $\mathbf{Y}_t$.

The proposed DSFT framework is general in the sense that it is not tailored to the specific types of feature mappings and distribution discrepancy metrics. In the following sections, we instantiate the framework to a linear case and a nonlinear case by specifying two particular mappings with a popular distribution discrepancy metric.

6.3 Instantiations of the Domain Specific Feature Transfer framework

To instantiate the DSFT framework, we need to specify the feature mappings and the distribution discrepancy metric in the objective function (6.5). In this section, we first introduce a nonparametric metric to measure the distribution discrepancy, and then propose a linear instantiation DSFT_l and a nonlinear instantiation DSFT_{nl} .

6.3.1 A Distribution Discrepancy Metric MMD

In this work, we propose to use Maximum Mean Discrepancy (MMD) (Borgwardt et al., 2006) as a quantifying metric to measure the distribution discrepancy. It is commonly used to measure the distance between two distributions (Pan et al., 2011) since it avoids the density estimations. In our context, the empirical MMD between $\mathbf{X}_{s,h}$ and $\mathbf{X}_{t,h}$ is ($\mathbf{x}_{s,h_i}$ and $\mathbf{x}_{t,h_j}$ is the single instance in source and target domain,

respectively):

$$\begin{aligned} Dist(\mathbf{X}_{s,h}, \mathbf{X}_{t,h}) &= \left\| \frac{1}{n} \sum_{i=1}^n \mathbf{x}_{s,h_i} - \frac{1}{m} \sum_{j=1}^m \mathbf{x}_{t,h_j} \right\|^2 \\ &= tr(\mathbf{X} \mathbf{M} \mathbf{X}^\top), \end{aligned} \quad (6.6)$$

where tr is the the operator to calculate the trace of a matrix, $\mathbf{X} = [\mathbf{X}_{s,h}, \mathbf{X}_{t,h}] \in \mathbb{R}^{(c+s+t) \times (n+m)}$, and $\mathbf{M}$ is defined as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11}, \mathbf{M}_{12} \\ \mathbf{M}_{21}, \mathbf{M}_{22} \end{bmatrix}$$

with $\mathbf{M}_{11} = \frac{1}{n^2} \mathbf{1}_{n \times n}$, $\mathbf{M}_{12} = \mathbf{M}_{21}^\top = -\frac{1}{mn} \mathbf{1}_{m \times n}$ and $\mathbf{M}_{22} = \frac{1}{m^2} \mathbf{1}_{m \times m}$ ($\mathbf{1}_{m \times n}$ is an $m \times n$ matrix with all ones).

6.3.2 A Linear Case DSFT_l

In this section, we instantiate a linear case of DSFT, DSFT_l, by assuming that the mappings between the common features and the domain specific features are linear transformations. Specifically, we represent the two feature mappings using two matrices $\mathbf{W}_s$ and $\mathbf{W}_t$, which means

$$\Psi_s(\mathbf{S}_c) = \mathbf{W}_s^\top \mathbf{S}_c, \quad \Psi_t(\mathbf{T}_c) = \mathbf{W}_t^\top \mathbf{T}_c. \quad (6.7)$$

By applying Eqs. (6.7) to Eqs. (6.2), we can construct the new homogeneous feature space as follows:

$$\mathbf{X}_{s,h}^l = [\mathbf{S}_c; \mathbf{S}_s; \mathbf{S}_a^l], \quad \mathbf{X}_{t,h}^l = [\mathbf{T}_c; \mathbf{T}_a^l; \mathbf{T}_t], \quad (6.8)$$

where $\mathbf{S}_a^l = \mathbf{W}_t^\top \mathbf{S}_c$ and $\mathbf{T}_a^l = \mathbf{W}_s^\top \mathbf{T}_c$. Then we plug $\mathbf{X}_{s,h}^l$ and $\mathbf{X}_{t,h}^l$ of Eq. (6.8) into Eq. (6.6), and further derive:

$$\begin{aligned} Dist_l(\mathbf{X}_{s,h}^l, \mathbf{X}_{t,h}^l) &= tr([\mathbf{X}_{s,h}^l, \mathbf{X}_{t,h}^l] \mathbf{M} [\mathbf{X}_{s,h}^l, \mathbf{X}_{t,h}^l]^\top) \\ &= tr([\mathbf{S}_c, \mathbf{T}_c] \mathbf{M} [\mathbf{S}_c, \mathbf{T}_c]^\top) + tr([\mathbf{S}_s, \mathbf{T}_a^l] \mathbf{M} [\mathbf{S}_s, \mathbf{T}_a^l]^\top) \\ &\quad + tr([\mathbf{S}_a^l, \mathbf{T}_t] \mathbf{M} [\mathbf{S}_a^l, \mathbf{T}_t]^\top) \\ &= tr(\mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top + 2\mathbf{S}_c \mathbf{M}_{12} \mathbf{T}_c^\top + \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top) \\ &\quad + tr(\mathbf{S}_s \mathbf{M}_{11} \mathbf{S}_s^\top + 2\mathbf{W}_s^\top \mathbf{T}_c \mathbf{M}_{21} \mathbf{S}_s^\top + \mathbf{W}_s^\top \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top \mathbf{W}_s) \\ &\quad + tr(\mathbf{T}_t \mathbf{M}_{22} \mathbf{T}_t^\top + 2\mathbf{W}_t^\top \mathbf{S}_c \mathbf{M}_{12} \mathbf{T}_t^\top + \mathbf{W}_t^\top \mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top \mathbf{W}_t). \end{aligned} \quad (6.9)$$

Finally, we can obtain the objective function of DSFT_l as follows:

$$\begin{aligned} \min_{\mathbf{W}_s, \mathbf{W}_t} & \|\mathbf{W}_s^\top \mathbf{S}_c - \mathbf{S}_s\|_F^2 + \|\mathbf{W}_t^\top \mathbf{T}_c - \mathbf{T}_t\|_F^2 \\ & + \alpha \text{Dist}_l(\mathbf{X}_{s,h}^l, \mathbf{X}_{t,h}^l) + \beta (\|\mathbf{W}_s\|_F^2 + \|\mathbf{W}_t\|_F^2), \end{aligned} \quad (6.10)$$

To solve the optimization problem (6.10), we first denote the final objective function as L , and define $L = L_1 + L_2 + L_3$ where

$$\begin{aligned} L_1 &= \|\mathbf{W}_s^\top \mathbf{S}_c - \mathbf{S}_s\|_F^2 + \beta \|\mathbf{W}_s\|_F^2 \\ &= \text{tr}(\mathbf{S}_c^\top \mathbf{W}_s \mathbf{W}_s^\top \mathbf{S}_c - 2\mathbf{S}_c^\top \mathbf{W}_s \mathbf{S}_s + \mathbf{S}_s^\top \mathbf{S}_s) \\ &\quad + \beta \text{tr}(\mathbf{W}_s^\top \mathbf{W}_s), \\ L_2 &= \|\mathbf{W}_t^\top \mathbf{T}_c - \mathbf{T}_t\|_F^2 + \beta \|\mathbf{W}_t\|_F^2 \\ &= \text{tr}(\mathbf{T}_c^\top \mathbf{W}_t \mathbf{W}_t^\top \mathbf{T}_c - 2\mathbf{T}_c^\top \mathbf{W}_t \mathbf{T}_t + \mathbf{T}_t^\top \mathbf{T}_t) \\ &\quad + \beta \text{tr}(\mathbf{W}_t^\top \mathbf{W}_t), \\ L_3 &= \alpha \text{Dist}_l(\mathbf{X}_{s,h}^l, \mathbf{X}_{t,h}^l). \end{aligned}$$

According to Eq. (6.9), L_3 is the summation of three trace terms. These three trace terms correspond to the discrepancy term of the common features, the source specific features, and the target specific features, respectively. Note that none of these three trace terms contain $\mathbf{W}_s$ and $\mathbf{W}_t$ simultaneously. This is a good property which leads to a closed-form solution for the final objective function. Specifically, we calculate the first-order derivatives of L with respect to $\mathbf{W}_s$ and $\mathbf{W}_t$:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{W}_s} &= \frac{\partial L_1}{\partial \mathbf{W}_s} + \frac{\partial L_3}{\partial \mathbf{W}_s}, \\ \frac{\partial L}{\partial \mathbf{W}_t} &= \frac{\partial L_2}{\partial \mathbf{W}_t} + \frac{\partial L_3}{\partial \mathbf{W}_t}, \end{aligned} \quad (6.11)$$

where the first-order derivatives with respect to $\mathbf{W}_s$ and $\mathbf{W}_t$ for L_1 , L_2 and L_3 are:

$$\begin{aligned} \frac{\partial L_1}{\partial \mathbf{W}_s} &= 2\mathbf{W}_s^\top \mathbf{S}_c \mathbf{S}_c^\top - 2\mathbf{S}_s \mathbf{S}_c^\top + 2\beta \mathbf{W}_s^\top, \\ \frac{\partial L_3}{\partial \mathbf{W}_s} &= 2\alpha \mathbf{W}_s^\top \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top + 2\alpha \mathbf{S}_s \mathbf{M}_{12} \mathbf{T}_c^\top, \\ \frac{\partial L_2}{\partial \mathbf{W}_t} &= 2\mathbf{W}_t^\top \mathbf{T}_c \mathbf{T}_c^\top - 2\mathbf{T}_t \mathbf{T}_c^\top + 2\beta \mathbf{W}_t^\top, \\ \frac{\partial L_3}{\partial \mathbf{W}_t} &= 2\alpha \mathbf{W}_t^\top \mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top + 2\alpha \mathbf{T}_t \mathbf{M}_{21} \mathbf{S}_c^\top. \end{aligned}$$

By setting Eqs. (6.11) to zero, we obtain the optimal $\mathbf{W}_s$ and $\mathbf{W}_t$ as follows:

$$\begin{aligned}\mathbf{W}_s &= [A/B]^\top, \\ A &= \mathbf{S}_s \mathbf{S}_c^\top - \alpha \mathbf{S}_s \mathbf{M}_{12} \mathbf{T}_c^\top, \\ B &= \mathbf{S}_c \mathbf{S}_c^\top + \alpha \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top + \beta \mathbf{I}, \\ \mathbf{W}_t &= [C/D]^\top, \\ C &= \mathbf{T}_t \mathbf{T}_c^\top - \alpha \mathbf{T}_t \mathbf{M}_{21} \mathbf{S}_c^\top, \\ D &= \mathbf{T}_c \mathbf{T}_c^\top + \alpha \mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top + \beta \mathbf{I}.\end{aligned}$$

6.3.3 A Nonlinear Case DSFT_{nl}

In this section, we instantiate a nonlinear case of DSFT, DSFT_{nl}, by assuming a nonlinear relationship between the common features and the domain specific features. To take the nonlinearity across the features into account, we propose a kernelized solution. Specifically, we first use a nonlinear mapping Φ to map the source and target common features to RKHS, respectively, where $\Phi : \mathbb{R}^c \rightarrow \mathcal{H}$ with $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$ as the kernel of $\mathcal{H}$. Then we construct the specific features by the mapped common features in $\mathcal{H}$ using a linear transformation for each domain. Thus, we have the nonlinear mappings¹:

$$\Psi_s(\mathbf{S}_c) = \mathbf{Q}_s^\top \Phi(\mathbf{S}_c), \quad \Psi_t(\mathbf{T}_c) = \mathbf{Q}_t^\top \Phi(\mathbf{T}_c). \quad (6.12)$$

Correspondingly, we can obtain the nonlinear version of Eqs. (6.1) and (6.2):

$$\begin{aligned}\min_{\mathbf{Q}_s} \|\mathbf{Q}_s^\top \Phi(\mathbf{S}_c) - \mathbf{S}_s\|_F^2, \quad \min_{\mathbf{Q}_t} \|\mathbf{Q}_t^\top \Phi(\mathbf{T}_c) - \mathbf{T}_t\|_F^2, \\ \mathbf{S}_a^{nl} = \mathbf{Q}_t^\top \Phi(\mathbf{S}_c), \quad \mathbf{T}_a^{nl} = \mathbf{Q}_s^\top \Phi(\mathbf{T}_c),\end{aligned}$$

According to (Schölkopf et al., 2001; Niyogi, 2004), the linear mapping in $\mathcal{H}$ can be represented as a linear combination of the data points in $\mathcal{H}$, that is, for each domain we have $\mathbf{Q}_s^\top = \mathbf{Q}_{sk}^\top \Phi(\mathbf{S}_c)^\top$ and $\mathbf{Q}_t^\top = \mathbf{Q}_{tk}^\top \Phi(\mathbf{T}_c)^\top$. By denoting $\mathbf{K}_{ss} = \Phi(\mathbf{S}_c)^\top \Phi(\mathbf{S}_c)$, $\mathbf{K}_{tt} = \Phi(\mathbf{T}_c)^\top \Phi(\mathbf{T}_c)$, $\mathbf{K}_{st} = \Phi(\mathbf{S}_c)^\top \Phi(\mathbf{T}_c)$, and $\mathbf{K}_{ts} = \mathbf{K}_{st}^\top$, we obtain the kernelized version of Eqs. (6.1) and (6.2):

$$\min_{\mathbf{Q}_{sk}} \|\mathbf{Q}_{sk}^\top \mathbf{K}_{ss} - \mathbf{S}_s\|_F^2, \quad \min_{\mathbf{Q}_{tk}} \|\mathbf{Q}_{tk}^\top \mathbf{K}_{tt} - \mathbf{T}_t\|_F^2, \quad (6.13)$$

¹Note that there are other ways to instantiate the nonlinear relationships. By formulating Eq. (6.12), we can obtain a closed-form solution for the nonlinear case, which is also similar to the linear one.

$$\mathbf{S}_a^{nl} = \mathbf{Q}_{tk}^\top \mathbf{K}_{ts}, \quad \mathbf{T}_a^{nl} = \mathbf{Q}_{sk}^\top \mathbf{K}_{st}.$$

Then we construct the new $\mathbf{X}_{s_h}^{nl}$ and $\mathbf{X}_{t_h}^{nl}$ using Eq. (6.3), and derive the new discrepancy term as:

$$\begin{aligned} & Dist_{nl}(\mathbf{X}_{s_h}^{nl}, \mathbf{X}_{t_h}^{nl}) \\ &= tr(\mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top + 2\mathbf{S}_c \mathbf{M}_{12} \mathbf{T}_c^\top + \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top) \\ &+ tr(\mathbf{S}_s \mathbf{M}_{11} \mathbf{S}_s^\top + 2\mathbf{Q}_{sk}^\top \mathbf{K}_{st} \mathbf{M}_{21} \mathbf{S}_s^\top \\ &\quad + \mathbf{Q}_{sk}^\top \mathbf{K}_{st} \mathbf{M}_{22} \mathbf{K}_{st}^\top \mathbf{Q}_{sk}) \\ &+ tr(\mathbf{T}_t \mathbf{M}_{22} \mathbf{T}_t^\top + 2\mathbf{Q}_{tk}^\top \mathbf{K}_{ts} \mathbf{M}_{12} \mathbf{T}_t^\top \\ &\quad + \mathbf{Q}_{tk}^\top \mathbf{K}_{ts} \mathbf{M}_{11} \mathbf{K}_{ts}^\top \mathbf{Q}_{tk}). \end{aligned} \tag{6.14}$$

Combining (6.13) and (6.14), we obtain the kernelized final objective function of $DSFT_{nl}$:

$$\begin{aligned} & \min_{\mathbf{Q}_{sk}, \mathbf{Q}_{tk}} \left\| \mathbf{Q}_{sk}^\top \mathbf{K}_{ss} - \mathbf{S}_s \right\|_F^2 + \left\| \mathbf{Q}_{tk}^\top \mathbf{K}_{tt} - \mathbf{T}_t \right\|_F^2 \\ & + \alpha Dist_{nl}(\mathbf{X}_{s_h}^{nl}, \mathbf{X}_{t_h}^{nl}) + \beta (\left\| \mathbf{Q}_{sk} \right\|_F^2 + \left\| \mathbf{Q}_{tk} \right\|_F^2). \end{aligned} \tag{6.15}$$

Similar to the linear case, we can obtain the optimal closed-form solutions $\mathbf{Q}_{sk}$ and $\mathbf{Q}_{tk}$ of (6.15) as follows:

$$\begin{aligned} \mathbf{Q}_{sk} &= [A_k/B_k]^\top, \\ A_k &= \mathbf{S}_s \mathbf{K}_{ss}^\top - \alpha \mathbf{S}_s \mathbf{M}_{12} \mathbf{K}_{st}^\top, \\ B_k &= \mathbf{K}_{ss} \mathbf{K}_{ss}^\top + \alpha \mathbf{K}_{st} \mathbf{M}_{22} \mathbf{K}_{st}^\top + \beta \mathbf{I}, \\ \mathbf{Q}_{tk} &= [C_k/D_k]^\top, \\ C_k &= \mathbf{T}_t \mathbf{K}_{tt}^\top - \alpha \mathbf{T}_t \mathbf{M}_{21} \mathbf{K}_{ts}^\top, \\ D_k &= \mathbf{K}_{tt} \mathbf{K}_{tt}^\top + \alpha \mathbf{K}_{ts} \mathbf{M}_{11} \mathbf{K}_{ts}^\top + \beta \mathbf{I}. \end{aligned}$$

6.4 Experimental Study

In this section, we first compare our DSFT, i.e. $DSFT_l$ and $DSFT_{nl}$, with state-of-the-art DA methods, and then analyze the properties of our DSFT.

6.4.1 Datasets & Experimental Setting

We utilize *Amazon product review dataset* and *20-Newsgroups dataset*, which have been introduced in Chapter 3. However, we formulate the experimental setting specific to the hybrid domain adaptation scenario.

Amazon product review dataset (Blitzer et al., 2007) contains sentiment reviews from four product categories: books (B), DVD (D), electronics (E) and kitchen appliance (K). Each review is labeled as positive or negative, and represented by the unigram and bigram tf-idf features. Each category has about 5,000 samples and is regarded as a domain. By pairing up any two domains, we have 6 domain pairs, denoted as ‘domain1-domain2’. By assigning one domain as source and the other as target, we construct one domain adaptation task, denoted as ‘source→target’. For example, in the domain pair B-D, B→D represents the domain adaptation task with B as source and D as target. Similarly we also have D→B with the source and the target flipped over.

20-Newsgroups dataset (Dai et al., 2007a) consists of about 20,000 documents characterized by word count features. They come from four top categories: computer (C), recording (R), science (S), and talk (T). Each top category has four subcategories. Top categories are treated as labels and subcategories are used to form related domains. Take computer (C) and recording (R) as an example. Top category C is labeled as positive and top category R as negative. Two subcategories under each class are selected to form domain1, while the other two subcategories constitute domain2. By pairing up domain1 and domain2, we have one domain pair, denoted as ‘CR₁₋₂’. By exchanging the roles of the two domains, we have two domain adaptation tasks, denoted as ‘CR_{1→2}’ and ‘CR_{2→1}’.

Given a domain pair, common features are the features shared by the domain pair while domain specific features are those only appearing in a single domain. We use the top 1,000 most frequent common features in the domain pair and the top 1,000 most frequent domain specific features in each domain. Thus, the input feature space of one domain consists of 1,000 common features and 1,000 domain specific features.

Depending on the different strategies used to handle the hybrid feature space, we have two different baseline settings: ‘*cut-off*’ and ‘*fill-up*’. In ‘*cut-off*’, we use only the common features, denoted as COM. In ‘*fill-up*’, we fill in zeros to those specific features appearing in one domain but not the other, denoted as F_Zero. By doing ‘*cut-off*’ or ‘*fill-up*’, the feature spaces of the two domains become homogeneous, and we can apply existing homogeneous domain adaptation methods (HomoDA). We also compare our DSFT with such class of baselines. Specifically, in our experiments,

we use state-of-the-art HomoDA methods: SA (Fernando et al., 2013), GFK (Gong et al., 2012), FEDA (Daume III, 2009), TKL (Long et al., 2015), and TaR (Clinchant et al., 2016). SVM is used as the base classifier. We do cross validation on the source data to select the best hyperparameter C in $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3]$. For SA and GFK, we use the default subspace dimension d if it is specified by the authors, otherwise we set d to be the smallest one that contributes 85% of the energy. For TKL, we follow the authors’ hyperparameter configuration for text data, and for TaR, we use $p = 0.9$ and single layer with *tanh* activation function as stated by the authors. For DSFT_{nl}, we use the standard RBF kernel function.

6.4.2 DSFT v.s. ‘cut-off’

We compare DSFT_l and DSFT_{nl} with the ‘cut-off’ baseline and its variants ‘cut-off+HomoDA’ on the two benchmark datasets. The results are shown in Table 6.1. The best result on each adaptation task is highlighted in bold, and the runner-up is highlighted in bold-italic. We also show the average results of all the tasks in each dataset.

Table 6.1 shows that both DSFT_l and DSFT_{nl} consistently outperform COM on the two datasets, while the performance of COM variants varies in different tasks. DSFT_{nl} is a clear winner and DSFT_l achieves the runner-up in terms of the average performance among all the baselines. They achieve the best and the second best performance on most of the adaptation tasks. Specifically, DSFT_{nl} performs the best in 19 out of 24 tasks, and the second best in the 4 tasks. DSFT_l makes the best only in 2 tasks, but yields the second best in 18 tasks. These demonstrate the superiority of our DSFT to the ‘cut-off’ based baselines on the hybrid DA problem. Moreover, in general DSFT_{nl} outperforms DSFT_l. This is reasonable since the relationship between the common features and the domain specific features is more likely to be nonlinear in the complex real-world data. However, considering that DSFT_l already yields better average results than other baselines and even achieves the best in the ST₁₋₂ domain pair, it is also an effective and valuable method for hybrid domain adaptation problem.

Table 6.1 also shows that the improvement of DSFT over COM is more pronounced on the 20-Newsgroups dataset (DSFT_l: 4.49%; DSFT_{nl}: 6.07%) than that on the Amazon dataset (DSFT_l: 1.38%; DSFT_{nl}: 2.37%). We also observe marginal

Table 6.1: *Cut-off* comparison results (%)

Data	COM	GFK _C	SA _C	FEDA _C	TKL _C	TaR _C	DSFT _l	DSFT _{nl}
B → D	74.40	70.34	73.41	74.83	51.60	75.24	75.28	75.95
B → E	69.55	69.72	64.26	69.90	62.09	70.51	70.72	73.87
B → K	70.05	71.99	69.00	70.18	59.27	70.71	72.38	73.17
D → B	74.16	67.20	72.75	73.91	55.00	74.56	75.18	76.18
D → E	70.04	73.95	68.36	70.24	63.72	71.04	73.34	74.51
D → K	73.72	72.86	71.43	73.50	63.46	74.36	75.85	76.81
E → B	68.96	67.25	57.15	69.15	52.35	70.38	70.58	72.11
E → D	71.86	70.18	69.79	72.00	57.05	72.57	73.21	73.86
E → K	81.49	76.50	79.82	81.25	58.76	79.88	81.66	81.92
K → B	69.84	68.45	68.60	69.93	64.64	71.05	71.11	71.84
K → D	72.80	68.74	69.77	72.55	58.75	72.70	73.45	74.42
K → E	80.46	76.50	78.83	80.41	66.21	79.32	81.10	81.16
Mean	73.11	71.14	70.26	73.16	59.41	73.53	74.49	75.48
CR _{1→2}	76.50	77.27	72.04	79.59	64.49	72.64	83.28	87.48
CR _{2→1}	72.99	76.32	74.27	75.13	83.61	61.37	79.74	81.11
CS _{1→2}	80.17	82.15	79.06	81.55	80.96	72.53	85.24	88.50
CS _{2→1}	81.09	80.75	82.61	81.85	71.20	77.61	84.39	88.89
CT _{1→2}	88.55	82.21	86.68	90.32	57.01	86.47	90.11	91.16
CT _{2→1}	86.45	90.75	82.76	89.05	52.15	91.02	91.29	93.00
RS _{1→2}	84.21	93.62	84.80	85.22	83.21	89.42	88.33	91.77
RS _{2→1}	76.75	83.91	79.87	76.83	59.73	75.48	81.80	81.89
RT _{1→2}	81.21	75.33	74.01	82.70	56.07	84.16	84.64	85.60
RS _{2→1}	76.89	79.45	74.62	76.33	50.99	80.78	81.06	81.44
ST _{1→2}	71.35	69.92	71.73	70.40	56.06	64.23	74.48	74.10
ST _{2→1}	76.41	63.96	79.33	78.45	51.86	81.98	82.07	80.48
Mean	79.38	79.64	78.48	80.62	63.94	78.14	83.87	85.45

improvements (relative to the average) on the tasks E→K and K→E and CT_{1→2}.

Such an improvement variation among different domain pairs appears to be dataset/task-dependent. It would be interesting to investigate the reason behind.

It is suspected that the variation may be an indication of different discriminative power of domain specific features in different tasks. We thus perform a quantitative analysis on it. To measure the discriminative power of domain specific features, we define a metric, P_{spec} , as follows. Given the data in a domain, we calculate the prediction accuracy under two cases: without domain specific features (i.e., using common features only); and with domain specific features (i.e., using all features). We conduct 5-fold cross-validation on the data to obtain the two accuracies, denoted as Acc_{wo} and Acc_w , respectively. The P_{spec} is then defined as:

$$P_{spec} = \frac{Acc_w - Acc_{wo}}{1 - Acc_{wo}}$$

A positive value of P_{spec} indicates that domain specific features carry the discriminative power and contribute to the prediction task. The larger the value is, the more powerful the discriminative capability is carried by domain specific features. Therefore, a larger P_{spec} gives more room for the performance improvement to be achieved when domain specific features are properly exploited.

Table 6.2 shows the Acc_{wo} , Acc_w and P_{spec} of each domain on the two datasets. In the Amazon dataset, BD_B denotes the domain ‘books’ in the domain pair B-D. In the 20-Newsgroups dataset, CR_1 denotes domain1 in the domain pair CR₁₋₂. From Table 6.2, we can see that P_{spec} of all domains are positive, which verifies the positive role that domain specific features play in the prediction. Also, P_{spec} values on 20-Newsgroups are much higher than those on Amazon, which explains why DSFT achieves more significant improvement on 20-Newsgroups. Interestingly, for those tasks on which DSFT achieves marginal improvements (E→K, K→E, and CT_{1→2}), the P_{spec} values of their target domains (EK_K, EK_E, and CT_2) are relatively small, compared to those of other domains in the same dataset. They indicate a small improvement room, which well explains the improvement variation of DSFT that we observed.

6.4.3 DSFT v.s. ‘fill-up’

Though domain specific features are shown to be useful, they should be exploited in a wise way in order to help with the adaptation. In the current practice, domain specific features are simply concatenated with common features, with features that do not appear in one domain filled with zeros. This corresponds to our baseline

Table 6.2: Discriminative power analysis

Dom	Acc_{wo}	Acc_w	P_{spec}	Dom	Acc_{wo}	Acc_w	P_{spec}
BD_B	75.87	78.71	0.117	CR_1	86.57	93.08	0.477
BD_D	76.32	79.14	0.119	CR_2	95.02	97.00	0.397
BE_B	76.95	80.93	0.173	CS_1	90.47	93.19	0.286
BE_E	79.07	83.27	0.201	CS_2	89.53	93.48	0.377
BK_B	77.60	81.64	0.180	CT_1	92.88	95.41	0.354
BK_K	81.55	84.30	0.149	CT_2	93.33	94.79	0.219
DE_D	77.89	81.19	0.149	RS_1	89.87	95.36	0.542
DE_E	80.27	83.46	0.162	RS_2	95.21	97.73	0.526
DK_D	77.22	81.15	0.173	RT_1	88.15	93.18	0.424
DK_K	81.77	83.91	0.117	RT_2	89.87	95.33	0.539
EK_E	81.58	81.64	0.004	ST_1	89.82	94.16	0.426
EK_K	81.89	82.64	0.042	ST_2	88.19	93.14	0.419

setting: F_Zero. We compare DSFT with this ‘*fill-up*’ baseline and its variants ‘*fill-up*+HomoDA’ on the two datasets. The results are reported in Table 6.3.

Table 6.3 shows that both $DSFT_l$ and $DSFT_{nl}$ consistently achieve better results than F_Zero in all adaptation tasks. For other variants, GFK_0 , SA_0 , $FEDA_0$, TKL_0 , and TaR_0 , the performance varies in different tasks. Again, $DSFT_{nl}$ achieves the best average performance and $DSFT_l$ achieves the second best, which shows the superiority of DSFT to the ‘*fill-up*’ based baselines on the hybrid domain adaptation problem.

We also observe that the results of F_Zero are comparable to those of COM. On average, the accuracy of COM is 73.11% and that of F_Zero is 72.84% on Amazon dataset, and they are 79.38% and 79.42% on 20-Newsgroups dataset. This indicates that the use of domain specific features in F_Zero brings no benefit to the adaptation task. The reason is as follows. For F_Zero, the weights of SVM trained on source have three components, ω_{S_c} , ω_{S_s} and ω_{S_a} , each corresponding to one type of features in the extended feature space. Since $\mathbf{S}_a$ has all zeros in F_Zero, ω_{S_a} trained from $\mathbf{S}_a$ simply induces noise when tested on the target data. As for ω_{S_s} , it is trained from $\mathbf{S}_s$, but the corresponding target feature matrix $\mathbf{T}_a$ is filled with zeros, which

Table 6.3: *Fill-up* comparison results (%)

Dataset	F_Zero	GFK ₀	SA ₀	FEDA ₀	TKL ₀	TrR ₀	DSFT _{<i>l</i>}	DSFT _{<i>nl</i>}
B → D	74.54	60.79	69.95	74.54	50.41	75.85	75.28	75.95
B → E	68.11	72.11	62.13	68.14	50.06	69.24	70.72	73.87
B → K	69.46	70.84	60.76	69.52	52.82	67.83	72.38	73.17
D → B	74.45	65.78	70.27	74.38	60.04	75.85	75.18	76.18
D → E	68.82	71.68	68.12	68.97	51.72	69.72	73.34	74.51
D → K	72.98	74.09	72.40	72.71	53.19	66.74	75.85	76.81
E → B	70.13	71.55	58.78	70.13	60.84	69.60	70.58	72.11
E → D	72.04	65.98	67.33	72.41	59.42	72.98	73.21	73.86
E → K	81.24	73.80	79.02	81.18	54.16	81.38	81.66	81.92
K → B	70.29	68.82	65.25	70.33	69.53	71.08	71.11	71.84
K → D	72.04	66.35	67.10	71.77	57.01	71.96	73.45	74.42
K → E	80.00	71.39	79.88	79.87	50.16	80.49	81.10	81.16
Mean	72.84	69.43	68.42	72.83	55.74	72.73	74.49	75.48
CR _{1→2}	80.36	81.88	73.33	78.22	66.64	59.43	83.28	87.48
CR _{2→1}	62.14	77.78	77.86	62.82	81.80	52.56	79.74	81.11
CS _{1→2}	77.68	86.78	84.21	78.80	84.81	51.07	85.24	88.50
CS _{2→1}	81.42	87.70	79.13	81.51	68.21	50.38	84.39	88.89
CT _{1→2}	89.91	88.76	82.83	90.74	89.59	70.03	90.11	91.16
CT _{2→1}	89.86	89.05	85.46	89.41	52.15	82.76	91.29	93.00
RS _{1→2}	85.22	86.99	80.69	85.14	77.01	70.36	88.33	91.77
RS _{2→1}	78.85	80.39	81.13	77.25	68.83	80.71	81.80	81.89
RT _{1→2}	80.95	80.95	80.60	82.44	56.07	83.23	84.64	85.60
RS _{2→1}	76.89	80.21	79.36	73.20	51.08	82.29	81.06	81.44
ST _{1→2}	71.35	71.73	75.90	70.87	56.06	74.38	74.48	74.10
ST _{2→1}	78.36	80.74	73.85	79.15	51.86	65.28	82.07	80.48
Mean	79.42	82.75	79.53	79.13	67.01	68.30	83.87	85.45

makes ω_{S_s} useless when doing the prediction. As a result, only ω_{S_c} is in effect and consequently domain specific features in F_Zero do not actually play a role. This explains why F_Zero and COM have similar performance.

Figure 6.2: NDCG for COM and DSFT

6.4.4 Adaptation Analysis of DSFT

In this section, we investigate why DSFT is effective in exploiting the domain specific features in the adaptation from two aspects: (1) reweighting and (2) new feature representation. We use the 20-Newsgroup dataset in the empirical evaluation.

6.4.4.1 Feature Reweighting

DSFT reweights the common features by exploiting the domain specific features. By doing so, those common features that can bring the two domains closer are popped up and ranked higher. We verify this using DSFT_l as the shown case.

First, we show how DSFT_l reweights the common features. The final features of DSFT_l shown in Eq. (6.8). We approximate $\mathbf{S}_s = \mathbf{W}_s^\top \mathbf{S}_c$, and $\mathbf{T}_t = \mathbf{W}_t^\top \mathbf{T}_c$ (not exactly equal due to the discrepancy term), and we obtain:

$$\mathbf{X}_{s,h}^l = [\mathbf{S}_c; \mathbf{W}_s^\top \mathbf{S}_c; \mathbf{W}_t^\top \mathbf{S}_c] = \mathbf{P} \mathbf{S}_c,$$

$$\mathbf{X}_{t,h}^l = [\mathbf{T}_c; \mathbf{W}_s^\top \mathbf{T}_c; \mathbf{W}_t^\top \mathbf{T}_c] = \mathbf{P} \mathbf{T}_c,$$

$$\mathbf{P} = [\mathbf{I}; \mathbf{W}_s^\top; \mathbf{W}_t^\top].$$

We use SVM weights as the feature importance for the two domains. For COM, we have:

$$\text{Src} : \mathbf{S}_c^\top * \omega_{com_s}, \quad \text{Tar} : \mathbf{T}_c^\top * \omega_{com_t}. \quad (6.16)$$

For DSFT_l , we have:

$$\begin{aligned} \text{Src} : \mathbf{X}_s^{h\top} \omega_{\text{DSFT}_{l_s}} &= \mathbf{S}_c^\top * \mathbf{P}^\top \omega_{\text{DSFT}_{l_s}}, \\ \text{Tar} : \mathbf{X}_t^{h\top} \omega_{\text{DSFT}_{l_t}} &= \mathbf{T}_c^\top * \mathbf{P}^\top \omega_{\text{DSFT}_{l_t}}. \end{aligned} \quad (6.17)$$

From Eq. (6.16) and Eq. (6.17), we can see that DSFT_l reweights the common feature importance ω_{com_s} and ω_{com_t} to $\mathbf{P}^\top \omega_{\text{DSFT}_{l_s}}$ and $\mathbf{P}^\top \omega_{\text{DSFT}_{l_t}}$ respectively.

Next, we verify that the two domains become closer after reweighting. We use the normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002) to measure the similarity of two ranks. In our case, we have:

$$\gamma = \frac{DCG_s}{DCG_t}.$$

The closer to 1 γ is, the closer the feature importance of the source and target is. We calculate γ_{COM} by using ω_{com_s} and ω_{com_t} . Similarly, we calculate $\gamma_{Reweight}$ by using $\mathbf{P}^\top \omega_{DSFT_s}$ and $\mathbf{P}^\top \omega_{DSFT_t}$. The results are shown in Figure 6.2. From Figure 6.2, we can see that $\gamma_{Reweight}$ is consistently larger than γ_{COM} , which indicates the two domains indeed become closer after reweighting.

DSFT is more than what we state above since the discrepancy term is taken into account. By using the weights of the final homogeneous features directly (e.g., $\omega_{DSFT_{l_s}}$ and $\omega_{DSFT_{l_t}}$ for $DSFT_l$), we can calculate γ_{DSFT_l} and $\gamma_{DSFT_{nl}}$ for $DSFT_l$ and $DSFT_{nl}$, respectively. As shown in Figure 6.2, we observe that both γ_{DSFT_l} and $\gamma_{DSFT_{nl}}$ consistently outperform $\gamma_{Reweight}$. This is because the discrepancy term brings the two domains further closer. Furthermore, $\gamma_{DSFT_{nl}}$ is larger than γ_{DSFT_l} , which indicates the performance improvements of $DSFT_{nl}$ to $DSFT_l$.

6.4.4.2 New Feature Representation

According to (Ben-David et al., 2007), a good feature representation for DA should satisfy (1) reducing the source training error; (2) reducing A-distance. In this section, we verify that DSFT ($DSFT_l$ and $DSFT_{nl}$) provides a good new feature representation reducing the source training error and A-distance simultaneously.

Followed by Ben-David et al. (2007), A-distance is approximated as:

$$A_{dis} = 2(1 - 2\epsilon)$$

where ϵ is the generalization error on the problem of discriminating between the source and target samples. In our experiment, we separate the source and target data into training set and test set. We train a linear SVM on the training set and apply it to the test set. The recognition error is taken as ϵ . Table 6.4 shows the source training error and A-distance for COM, F_Zero, $DSFT_l$, and $DSFT_{nl}$.

As shown in Table 6.4, both $DSFT_l$ and $DSFT_{nl}$ reduce the source training error and A-distance when compared to COM. We also observe that (1) $DSFT_l$ is

Table 6.4: Source training error (e-2) and A-distance

Metric	Source Training Error				A-distance			
Domain	COM	F_Zero	DSFT _l	DSFT _{nl}	COM	F_Zero	DSFT _l	DSFT _{nl}
CR _{1→2}	6.752	2.735	2.735	2.735	1.418	1.703	1.317	1.310
CR _{2→1}	3.173	1.286	1.115	1.286	1.417	1.683	1.314	1.312
CS _{1→2}	4.835	3.732	3.902	3.562	1.223	1.546	1.180	1.170
CS _{2→1}	5.571	3.262	3.262	3.262	1.203	1.514	1.154	1.135
CT _{1→2}	2.783	1.885	1.885	1.795	1.306	1.490	1.190	1.210
CT _{2→1}	2.706	2.289	2.393	2.185	1.288	1.476	1.197	1.190
RS _{1→2}	4.465	1.938	2.106	2.106	1.607	1.848	1.546	1.532
RS _{2→1}	2.435	0.756	0.924	0.840	1.600	1.849	1.566	1.552
RT _{1→2}	5.114	2.557	2.462	2.367	1.505	1.752	1.432	1.453
RT _{2→1}	3.600	1.844	1.756	1.668	1.500	1.762	1.420	1.444
ST _{1→2}	4.064	2.120	2.120	2.120	1.425	1.693	1.361	1.360
ST _{2→1}	5.028	2.182	1.992	2.087	1.441	1.705	1.339	1.328
Average	4.225	2.215	2.221	2.168	1.411	1.668	1.335	1.333

comparable with F_Zero on the source training error but much better on A-distance, and (2) DSFT_{nl} achieves smaller results on both the source training error and A-distance than F_Zero. These show that DSFT (DSFT_l and DSFT_{nl}) provides a better representation for DA than both COM and F_Zero, and thus is superior in the domain adaptation task. Moreover, when compared DSFT_{nl} to DSFT_l, DSFT_{nl} yields smaller average results than DSFT_l on both metrics, which further indicates the superior adaptation performance of DSFT_{nl} to that of DSFT_l. When compared F_Zero to COM, we can see that the source training error is reduced but A-distance is increased. The joint effect to benefit adaptation tasks is by chance, which implies that the adaptation performance of F_Zero is comparable with COM in general.

6.4.5 Property Analysis of DSFT

In this section, we study the proposed DSFT in three aspects: 1) sensitivity analysis on the balancing parameter α and β ; 2) the individual effect of the translation term and the discrepancy term; 3) the individual effect of the source specific features and the target specific features. All the analyses are done on both DSFT_l and DSFT_{nl}.

Figure 6.3: Sensitivity analysis on $RT_{1 \rightarrow 2}$ in $DSFT_l$

6.4.5.1 Sensitivity Analysis

We conduct the sensitivity analysis with respect to α and β , where α takes values in the range $\{1, 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$, and β takes values in the range $\{0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$. We take results of the task $RT_{1 \rightarrow 2}$ in $DSFT_l$ as a showcase in Figure 6.3. As shown in Figure 4.2(a), with the increase of α , the classification accuracy generally increases until it gets stable. The larger α is, the more sensitive the discrepancy term is to the objective function. When the empirical MMD gets stable, larger values of α cannot further improve the performance. For $RT_{1 \rightarrow 2}$, this happens when θ reaches 10^5 , which is also the case for all other domain adaptation tasks. For hyperparameter β , it is used to control the model complexity and avoid overfitting. With the change of β , we can see that the accuracy reaches the peak when β is between 0.5 and 1. In our experiments, we set $\alpha = 10^5$ and $\beta = 1$ as default for both $DSFT_l$ and $DSFT_{nl}$.

6.4.5.2 Term Analysis

We analyze the individual effect of the translation term and the discrepancy term, as well as their joint effect, with respect to the adaptation performance. For each of $DSFT_l$ and $DSFT_{nl}$, We develop two variants: one with the translation term but not the discrepancy term ($DSFT_{l-T}$ and $DSFT_{nl-T}$), the other with the discrepancy term but not the translation term ($DSFT_{l-D}$ and $DSFT_{nl-D}$). We compare the

Table 6.5: Average results (%) for term analysis

Method	F_Zero	DSFT _l -T	DSFT _l -D	DSFT _l
Accuracy	76.26	78.29	77.39	79.18
Method	F_Zero	DSFT _{nl} -T	DSFT _{nl} -D	DSFT _{nl}
Accuracy	76.26	79.91	78.24	80.47

Table 6.6: Domain specific feature analysis

Method	COM	S_DSFT _l	T_DSFT _l	DSFT _l
Accuracy	76.25	77.15	78.04	79.18
Method	COM	S_DSFT _{nl}	T_DSFT _{nl}	DSFT _{nl}
Accuracy	76.25	77.66	79.72	80.47

proposed DSFT_l and DSFT_{nl} with F_Zero and the corresponding two variants. We show the average accuracy of all the tasks on the two datasets in Table 6.5.

Table 6.5 shows that the two variants in both DSFT_l and DSFT_{nl} improve the accuracy of F_Zero, which demonstrates the effectiveness of each individual term in DSFT. Further more, integrating these two terms together, which is DSFT_l or DSFT_{nl}, is able to further boost the improvement.

6.4.5.3 Domain Specific Feature Analysis

The final feature representation consists of the common features, the source specific features, and the target specific features. In this section, we discuss how the individual domain specific features affect the adaptation performance.

For each of DSFT_l and DSFT_{nl}, we design two more variants: one is only combining the common features with the source specific features (S_DSFT_l and S_DSFT_{nl}), and another one is only combining the common features with the target specific features (T_DSFT_l and T_DSFT_{nl}). We compare DSFT_l and DSFT_{nl} with the corresponding two variants in terms of the adaptation performance on the two datasets. We also take the results of COM as a reference. We show the average accuracy of all the tasks in Table 6.6.

Table 6.6 shows that DSFT is the best, followed by T_DSFT, S_DSFT, and lastly

COM in both linear and nonlinear cases. This illustrates that combining common features with specific features from any domain can improve the performance of COM. Concatenation of all the features together can achieve the best results. It also shows that the target specific features can bring more benefit to the adaptation performance than the source specific ones. This is because the prediction task is on the target domain, and thus the target features are more informative.

6.5 Summary

We summarize the contributions of this chapter as follows:

- We propose a new hybrid domain adaptation problem setting, in which two domains share some common features and each domain also has its own specific features. This problem setting is a special case of heterogeneous domain adaptation, but it can be efficiently solved without any supplementary information.
- We develop a novel general domain specific feature transfer (DSFT) framework for hybrid domain adaptation. We utilize common features to link up two domains. The translations that represent the relationships between common features and specific features are learned, and then are cross-used to construct a universal homogeneous feature space for two domains. The domain divergence is also reduced when the homogeneous feature space is built.
- We instantiate a linear case and a nonlinear case for the general DSFT framework. With the selection of MMD as domain divergence metric and the application of kernel trick, the closed-form solution for each case can be easily derived.
- We conduct extensive experiments and demonstrate the superiority of DSFT to state-of-the-art domain adaptation methods with “*cut-off*” and “*fill-up*” strategies. We also demonstrate the effectiveness of DSFT from the perspectives of reweighting and new feature representations. The properties of DSFT including term analysis and domain specific feature analysis are also studied. All the experimental results indicate DSFT is an effective framework for hybrid domain adaptation.

Chapter 7

Transfer Covariance Function Stacking

Note that the frameworks that we have proposed above, no matter for homogeneous domain adaptation problems or for hybrid domain adaptation problems, are within the scope of domain adaptation strictly following the definition 1.1 where the tasks of two domains are the same. However, as stated in (Margolis, 2011), domain adaptation problems also include the case where two domains may share the same label space, but have different conditional distributions. In this chapter, we consider this more difficult problem setting, where source and target tasks are different but they follow a joint probability distribution. More challengingly, we are dealing with a multiple source adaptation problem.

With research efforts on domain adaptation largely confined to the single-source setting (Pan et al., 2011; Zhou et al., 2016), an increasing amount of studies are contributing to a realistic applicability of domain adaptation by addressing the multi-source scenario, mainly for classification (Tommasi et al., 2014; Fang et al., 2015; Bhatt et al., 2016). The problem of regression, however, has been much less studied, despite of the variety of real-world domains in which it arises; for instance wifi or indoor signal location (Pan et al., 2008), biological data analysis (Lam et al., 2016), or mechanical system design (Ghosh et al., 2015). In this work, we concentrate on *multi-source adaptation regression* based on *Gaussian process* (*GP*) models.

All the way through, the transfer learning community has been paying attention to modeling the similarity between different domains so that only the source knowledge that is helpful for the target domain is transferred. This is because designing

a method based on the assumption that domains are mutually relevant may lead to negative transfer (Pan and Yang, 2010). Similarity capture is particularly crucial in multi-source domain adaptation as the transfer capacity to the target task may differ considerably across the diverse source domains. Thus, domain adaptation methods that are capable of tuning the strength of the knowledge transfer to the similarity of the domains are attracting increasing interest (Luo et al., 2008; Wang et al., 2014c; Al-Stouhi and Reddy, 2011).

As regards to multi-source adaptation regression, a key issue is to capture the diverse *Source-Target* (S - T) similarities. The relatively few efforts to date have focused on ensemble methods. Particularly, an amount of works rely on the boosting strategy due to its capability to capture fine-grained S - T similarities by weighting the contribution of train instances individually (Dai et al., 2007b; Pardoe and Stone, 2010; Yao and Doretto, 2010). However, as outlined in (Al-Stouhi and Reddy, 2011), such an instance-based similarity strategy in boosting has shown issues with slow/premature weights convergence that have seriously penalized the computational cost or the adaptation performance. Another type of ensemble strategy for multi-source adaptation is stacking (Wolpert, 1992). Pardoe and Stone propose a meta-model that aggregates the predictions of several base models previously learned with each source in isolation (Pardoe and Stone, 2010). The aggregation is done by assigning each base model a model importance. In this case, the S - T similarities can be captured through the model importance. However, in such stacking-based methods, the base models suffer from a lack of consideration of the dependencies between the different source domains.

Another popular idea to model the S - T similarity is to construct a transfer covariance function that relates two data points from distinct domains through the similarity coefficients (Bonilla et al., 2008; Williams et al., 2009). Such idea has been proposed in multi-task learning (Bonilla et al., 2008), where each task pair is assigned a particular similarity coefficient. Note, however, that multi-task learning differs from the domain adaptation problem in that the former aims at improving performance across all the domains while the objective of the latter focuses on the target domain only. Nevertheless, the idea of transfer covariance function is referential for the domain adaptation problem. In (Cao et al., 2010), a *single source transfer covariance function* (TC_{SS}) was proposed. In the corresponding transfer

covariance matrix, one similarity coefficient was assigned to the S - T block to model the inter-domain similarity. A GP with such TC_{SS} (called GP - TC_{SS}) was then trained for the adaptation task.

When generalizing to multi-source adaptation regression, one may naturally consider a *multi-source transfer covariance function* (TC_{MS}) with different similarity coefficients attached to distinct S - T blocks in the corresponding transfer covariance matrix. In this work, we investigate the feasibility of such covariance function. We theoretically prove that a general GP with TC_{MS} (GP - TC_{MS}) fails to capture the similarity diversities of various S - T domain pairs. Although TC_{MS} intends to utilize different similarity coefficients, the learned GP - TC_{MS} would give the same similarity coefficient for all the S - T domain pairs. The generalization error bounds of the learned GP - TC_{MS} show that this coefficient is taking effect in every source domain. Considering the diverse S - T similarities between the sources and the target, this may jeopardize the adaptation performance, especially when the number of sources increases. Moreover, the learning of GP - TC_{MS} rapidly poses a computational issue with increasing amounts of source domains as usually $\mathcal{O}(m^3)$ computations are required to evaluate a model for m data points.

The unsatisfactory performance of GP - TC_{MS} leads us to exploit the transfer covariance function in another way. Considering that both the stacking strategy and the transfer covariance function can model the S - T similarity and using the transfer covariance function at the base models would therefore add flexibility to the similarity capture capability of the stacking approach, we propose to integrate them into one unified model. Specifically, we first discuss $TC_{SS}Stack$, a method that simply stacks GP - TC_{SS} base models. $TC_{SS}Stack$ alleviates the computational issue of GP since it allows to stretch the number of sources due to its $\mathcal{O}(Nn^3)$ cost for N sources with n points each. However, $TC_{SS}Stack$ still suffers from the aforementioned limitation of conventional stacking. Thus, we propose a more involved $TC_{MS}Stack$. Two salient features make $TC_{MS}Stack$ significantly different from $TC_{SS}Stack$: (i) it associates the similarity coefficients in the base GP - TC_{SS} with the model importance during learning, and (ii) it learns the model importance and the base GP - TC_{SS} jointly. By doing so, on the one hand, $TC_{MS}Stack$ further reduces the computational cost by lowering the number of optimization variables. On the other hand, although the similarity coefficient in $TC_{MS}Stack$ represents bi-

variate S - T similarity relations, they are elicited by pondering all the inter-domain dependencies. In the experiments, we show the superiority of $TC_{MS}Stack$ on the adaptation performance compared to $TC_{SS}Stack$, GP - TC_{MS} , and other multi-source adaptation regression methods.

The rest of the chapter is organized as follows. Section 7.1 states the multi-source adaptation regression problem setting and notations used in this chapter. Section 7.2 investigates a family of transfer covariance function (TC_{MS}), and further demonstrate the pathology of TC_{MS} for multi-source adaptation regression problems. Section 7.3 proposes a novel method that integrates transfer covariance function with stacking. Section 7.4 presents the empirical studies and comparisons. Finally, section 7.5 summarizes the chapter.

7.1 Problem Statement

We denote a domain set for multi-source adaptation regression as $\mathcal{D} = \mathcal{S} \cup \mathcal{T}$ where $\mathcal{S} = \{\mathcal{S}_i : 1 \leq i \leq N\}$ is a set of source domains and $\mathcal{T}$ is the target domain. All source domain data and few target domain data are labeled. Denote the data matrix and its corresponding label vector in each source domain $\mathcal{S}_i$ as $\mathbf{X}^{(\mathcal{S}_i)} \in \mathbb{R}^{n_{\mathcal{S}_i} \times d}$ and $\mathbf{y}^{(\mathcal{S}_i)} \in \mathbb{R}^{n_{\mathcal{S}_i}}$. Likewise, we represent the target data set with $\mathbf{X}^{(\mathcal{T})} = \{\mathbf{X}^{(\mathcal{T}_l)}, \mathbf{X}^{(\mathcal{T}_u)}\}$ where $\mathbf{X}^{(\mathcal{T}_l)} \in \mathbb{R}^{n_{\mathcal{T}_l} \times d}$ is the labeled target data matrix and $\mathbf{X}^{(\mathcal{T}_u)} \in \mathbb{R}^{n_{\mathcal{T}_u} \times d}$ is the unlabeled one. We further define $\mathbf{y}^{(\mathcal{T}_l)} \in \mathbb{R}^{n_{\mathcal{T}_l}}$ as the label vector for $\mathbf{X}^{(\mathcal{T}_l)}$. Moreover, we assume $n_{\mathcal{T}_l} \ll \min(n_{\mathcal{S}_1}, \dots, n_{\mathcal{S}_N}, n_{\mathcal{T}_u})$. Our objective is to utilize $\{\mathbf{X}^{(\mathcal{S}_i)}, \mathbf{y}^{(\mathcal{S}_i)}\}_{i=1}^N$ and $\{\mathbf{X}^{(\mathcal{T}_l)}, \mathbf{y}^{(\mathcal{T}_l)}\}$ to predict labels for $\mathbf{X}^{(\mathcal{T}_u)}$.

We use the GP model for this regression task. We denote the underlying latent function between the inputs $\mathbf{x}$ and the outputs y as f , and the noise variance as σ^2 . Thus, $\mathbf{f}$ denotes the function vector over $\mathbf{X}$. A GP model defines a Gaussian distribution over the functions, $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ in which $\boldsymbol{\mu}$ is the mean vector and $\mathbf{K}$ is the covariance matrix which is *positive semi-definite* (PSD , or equivalently denoted as $\mathbf{K} \succeq \mathbf{0}$). Usually $\boldsymbol{\mu}$ is assumed to be $\mathbf{0}$, and thus the GP model is completely specified by $\mathbf{K}$ given a covariance function which is parameterized by $\boldsymbol{\Omega}$.

7.2 GP with Transfer Covariance Function

In this section, we analyze the adaptation performance of GP using a specific family of transfer covariance function.

7.2.1 Transfer Covariance Function for Multi-Source

Since the GP model is specified by $\mathbf{K}$, one straightforward way to achieve the knowledge transfer across multiple source domains and the target domain is to design a transfer covariance function for multi-source. Different from a classical GP which uses a fixed covariance function for the data from different domains, we focus on the covariance function of the form (TC_{MS}):

$$k_*(\mathbf{x}, \mathbf{x}') = \begin{cases} \lambda_i k(\mathbf{x}, \mathbf{x}'), & \mathbf{x} \in \mathbf{X}^{(\mathcal{S}_i)} \text{ \& } \mathbf{x}' \in \mathbf{X}^{(\mathcal{T})} \\ & \text{or } \mathbf{x} \in \mathbf{X}^{(\mathcal{T})} \text{ \& } \mathbf{x}' \in \mathbf{X}^{(\mathcal{S}_i)}, \\ k(\mathbf{x}, \mathbf{x}'), & \text{otherwise.} \end{cases} \quad (7.1)$$

where $k(\cdot, \cdot)$ is any valid covariance function, and λ_i is the metric measuring the similarity between the source $\mathcal{S}_i$ and the target $\mathcal{T}$. Through the learning, λ_i is expected to capture the different transfer strengths in different S - T domain pairs. Those highly target-related sources will play a more important role in transfer, while those completely target-unrelated sources will not be considered. However, to guarantee the GP model is always valid, any covariance matrix $\mathbf{K}_*$ constructed by $k_*(\cdot, \cdot)$ should be *PSD*. Theorem 7.1 gives the sufficient and necessary condition for a *PSD* $\mathbf{K}_*$.

Theorem 7.1. *Let $\mathbf{K}_{\mathcal{D}_i \mathcal{D}_j}$ ($\mathcal{D}_i, \mathcal{D}_j \in \mathcal{D}$) denote a covariance matrix for points in $\mathcal{D}_i$ and $\mathcal{D}_j$. A Gram matrix*

$$\mathbf{K}_* = \begin{bmatrix} \mathbf{K}_{\mathcal{S}_1 \mathcal{S}_1} & \dots & \mathbf{K}_{\mathcal{S}_1 \mathcal{S}_N} & \lambda_1 \mathbf{K}_{\mathcal{S}_1 \mathcal{T}} \\ \dots & \dots & \dots & \dots \\ \mathbf{K}_{\mathcal{S}_N \mathcal{S}_1} & \dots & \mathbf{K}_{\mathcal{S}_N \mathcal{S}_N} & \lambda_N \mathbf{K}_{\mathcal{S}_N \mathcal{T}} \\ \lambda_1 \mathbf{K}_{\mathcal{T} \mathcal{S}_1} & \dots & \lambda_N \mathbf{K}_{\mathcal{T} \mathcal{S}_N} & \mathbf{K}_{\mathcal{T} \mathcal{T}} \end{bmatrix}$$

is PSD for any covariance matrix $\mathbf{K}$ in the form

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{\mathcal{S}_1\mathcal{S}_1} & \dots & \mathbf{K}_{\mathcal{S}_1\mathcal{S}_N} & \mathbf{K}_{\mathcal{S}_1\mathcal{T}} \\ \dots & \dots & \dots & \dots \\ \mathbf{K}_{\mathcal{S}_N\mathcal{S}_1} & \dots & \mathbf{K}_{\mathcal{S}_N\mathcal{S}_N} & \mathbf{K}_{\mathcal{S}_N\mathcal{T}} \\ \mathbf{K}_{\mathcal{T}\mathcal{S}_1} & \dots & \mathbf{K}_{\mathcal{T}\mathcal{S}_N} & \mathbf{K}_{\mathcal{T}\mathcal{T}} \end{bmatrix}$$

if and only if $\lambda_1 = \dots = \lambda_N$ and $|\lambda_i| \leq 1$.

From Theorem 7.1, we can see that $|\lambda_i| \leq 1$, which means a highly target-related source results in a full transfer of $\mathbf{K}_{\mathcal{S}_i, \mathcal{T}}$, but a completely target-unrelated source results in a zero block matrix. This indicates the adaptiveness of λ_i . However, Theorem 7.1 also shows that $k_*(\cdot, \cdot)$ can just give one similarity coefficient for all S - T domain pairs to ensure the validity of $GP\text{-}TC_{MS}$. Such single similarity coefficient compromises the diverse similarities between different S - T domain pairs. This violates the original intention of λ_i which is to distinguish the similarity diversity between different S - T domain pairs.

7.2.2 Generalization Bounds of $GP\text{-}TC_{MS}$

To investigate the effect of a single compromised similarity coefficient on the performance of $GP\text{-}TC_{MS}$, we derive its generalization error bounds. In (Chai, 2009), an earlier analysis can be found for the generalization errors and learning curves in multi-task learning (specifically, two learning tasks with the same noise variance). Our investigation is different from that work however as we are working on a domain adaptation setting, and more importantly, on multiple sources with different noise variances.

We denote the single compromised similarity coefficient as λ , and the noise variance for different domains as σ_d^2 , $d \in \{\mathcal{S}_1, \dots, \mathcal{S}_N, \mathcal{T}\}$. Thus, the transfer covariance matrix of the noisy training data is $\mathbf{C}_* = \mathbf{K}_* + \mathbf{\Sigma}$ where

$$\mathbf{K}_* = \begin{bmatrix} \mathbf{K}_{SS} & \lambda \mathbf{K}_{S\mathcal{T}} \\ \lambda \mathbf{K}_{S\mathcal{T}}^\top & \mathbf{K}_{\mathcal{T}\mathcal{T}} \end{bmatrix}, \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_S & \mathbf{0} \\ \mathbf{0} & \sigma_{\mathcal{T}}^2 \mathbf{I}_{\mathcal{T}\mathcal{T}} \end{bmatrix}$$

and $\mathbf{\Sigma}_S$ is a diagonal block matrix with the following diagonal block elements $\{\sigma_{\mathcal{S}_1}^2 \mathbf{I}_{\mathcal{S}_1\mathcal{S}_1}, \dots, \sigma_{\mathcal{S}_N}^2 \mathbf{I}_{\mathcal{S}_N\mathcal{S}_N}\}$. According to (Rasmussen, 2006), if the GP prior is correctly specified, the generalization error at a point is also the posterior variance at

such point. Specifically, for $GP-TC_{MS}$, the posterior variance at the target point $\mathbf{x}_t$ is:

$$\begin{aligned} \delta_{\mathcal{T}}^2(\mathbf{x}_t, \lambda, \{\mathbf{X}^{(S_i)}\}_{i=1}^N, \mathbf{X}_{\mathcal{T}}, \sigma_{\mathcal{T}}^2) \\ = k_{tt} - \mathbf{k}_{*t}^{\top} \mathbf{C}_*^{-1} \mathbf{k}_{*t}, \end{aligned} \quad (7.2)$$

where $\mathbf{k}_{*t}^{\top} = (\lambda \mathbf{k}_{St}^{\top}, \mathbf{k}_{Tt}^{\top})$, $\mathbf{k}_{St}$ ($\mathbf{k}_{Tt}$) is the vector of covariances between $\{\mathbf{X}^{(S_i)}\}_{i=1}^N$ ($\mathbf{X}^{(T)}$) and $\mathbf{x}_t$, and k_{tt} is the prior variance at $\mathbf{x}_t$. Wherever it is not misleading, we will simplify the posterior variance expression using $\delta_{\mathcal{T}}^2(\lambda, \{\sigma_{S_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)$. The generalization error for the target domain can be obtained by averaging eq. (7.2) over $\mathbf{x}_t$:

$$\begin{aligned} \epsilon_{\mathcal{T}}(\lambda, \{\mathbf{X}^{(S_i)}\}_{i=1}^N, \mathbf{X}_{\mathcal{T}}, \sigma_{\mathcal{T}}^2) \\ = \int \delta_{\mathcal{T}}^2(\mathbf{x}_t, \lambda, \{\sigma_{S_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) p(\mathbf{x}_t) d\mathbf{x}_t. \end{aligned} \quad (7.3)$$

To derive the generalization error bounds for $GP-TC_{MS}$, we first rewrite

$$\mathbf{C}_* = \mathbf{\Lambda} \begin{bmatrix} \lambda^{-2}(\mathbf{K}_{SS} + \mathbf{\Sigma}_S) & \mathbf{K}_{S\mathcal{T}} \\ \mathbf{K}_{S\mathcal{T}}^{\top} & \mathbf{K}_{\mathcal{T}\mathcal{T}} + \sigma_{\mathcal{T}}^2 \mathbf{I}_{\mathcal{T}\mathcal{T}} \end{bmatrix} \mathbf{\Lambda},$$

where $\mathbf{\Lambda} = \begin{bmatrix} \lambda \mathbf{I}_{SS} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{\mathcal{T}\mathcal{T}} \end{bmatrix}$. Thus, the posterior variance at point $\mathbf{x}_t$ becomes:

$$\delta_{\mathcal{T}}^2(\lambda, \{\sigma_{S_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) = k_{tt} - \mathbf{k}_t^{\top} \Phi(\lambda, \{\sigma_{S_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)^{-1} \mathbf{k}_t, \quad (7.4)$$

where $\mathbf{k}_t^{\top} = (\mathbf{k}_{St}^{\top}, \mathbf{k}_{Tt}^{\top})$ and

$$\begin{aligned} \Phi(\lambda, \{\sigma_{S_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \\ = \begin{bmatrix} \lambda^{-2}(\mathbf{K}_{SS} + \mathbf{\Sigma}_S) & \mathbf{K}_{S\mathcal{T}} \\ \mathbf{K}_{S\mathcal{T}}^{\top} & \mathbf{K}_{\mathcal{T}\mathcal{T}} + \sigma_{\mathcal{T}}^2 \mathbf{I}_{\mathcal{T}\mathcal{T}} \end{bmatrix}. \end{aligned}$$

Note that the above derivation excludes the situation where $\lambda = 0$. When $\lambda = 0$, all the source domains are unrelated to the target domain, and thus no knowledge is transferred. This is easy to verify by plugging $\lambda = 0$ into eq. (7.4).

Further, we observe that $\delta_{\mathcal{T}}^2$ is equal for λ and $-\lambda$, so we only investigate the

case $\lambda \in (0, 1]$. For eq. (7.4), we further decompose it as:

$$\begin{aligned}
& \Phi(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \\
&= \begin{bmatrix} \mathbf{K}_{SS} & \mathbf{K}_{ST} \\ \mathbf{K}_{ST}^\top & \mathbf{K}_{TT} \end{bmatrix} + \begin{bmatrix} \sigma_{\mathcal{T}}^2 \mathbf{I}_{SS} & \mathbf{0} \\ \mathbf{0} & \sigma_{\mathcal{T}}^2 \mathbf{I}_{TT} \end{bmatrix} + \\
& (\lambda^{-2} - 1) \begin{bmatrix} \mathbf{K}_{SS} + \boldsymbol{\Sigma}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Sigma}_S - \sigma_{\mathcal{T}}^2 \mathbf{I}_{SS} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&= \Phi(1, \{\sigma_{\mathcal{T}}^2, \dots, \sigma_{\mathcal{T}}^2\}_N, \sigma_{\mathcal{T}}^2) + \mathbf{E}_1 + \mathbf{E}_2,
\end{aligned} \tag{7.5}$$

where $\mathbf{E}_1 = (\lambda^{-2} - 1) \begin{bmatrix} \mathbf{K}_{SS} + \boldsymbol{\Sigma}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ and $\mathbf{E}_2 = \begin{bmatrix} \boldsymbol{\Sigma}_S - \sigma_{\mathcal{T}}^2 \mathbf{I}_{SS} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$. Eq. (7.5)

unveils that the posterior variance of having instances from different source domains is equivalent to the posterior variance of having those instances from target domain with two additional correlated noise terms, $\mathbf{E}_1$ and $\mathbf{E}_2$. This shows us the two main factors that matter in the adaptation performance; namely, the S - T similarity and the noise variances. To further analyze how the S - T similarity affects the adaptation performance, we focus on one factor and fix the other. Assuming that all the sources are totally related to the target, i.e. $\lambda = 1$, and consequently, the noise variance for each source becomes $\xi_{\mathcal{S}_i}^2$, we define the difference:

$$\Delta = \delta_{\mathcal{T}}^2(1, \{\xi_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) - \delta_{\mathcal{T}}^2(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2).$$

To obtain the upper (lower) bound of $\delta_{\mathcal{T}}^2(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)$, we are interested in those $\bar{\xi}_{\mathcal{S}_i}^2$ ($\underline{\xi}_{\mathcal{S}_i}^2$) that make $\Delta \geq 0$ ($\Delta < 0$) for all the target points.

Proposition 7.1. *Let $\bar{\delta}$ and $\underline{\delta}$ be the maximum and minimum eigenvalues of $\mathbf{K}_{SS}$, $\bar{\xi}_{\mathcal{S}_i}^2 = \lambda^{-2} \sigma_{\mathcal{S}_i}^2 - (1 - \lambda^{-2}) \bar{\delta}$ and $\underline{\xi}_{\mathcal{S}_i}^2 = \lambda^{-2} \sigma_{\mathcal{S}_i}^2 - (1 - \lambda^{-2}) \underline{\delta}$ for every source $\mathcal{S}_i$. Then, for all the target data points, $\delta_{\mathcal{T}}^2(1, \{\xi_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \leq \delta_{\mathcal{T}}^2(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \leq \delta_{\mathcal{T}}^2(1, \{\bar{\xi}_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)$.*

Proposition 7.1 gives the bounds of the posterior variance $\delta_{\mathcal{T}}^2(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)$. By applying eq. (7.3), we readily obtain the generalization error bounds.

Corollary 7.1. *Let*

$$\begin{aligned}
\bar{\epsilon}_{\mathcal{T}}(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) &= \epsilon_{\mathcal{T}}(1, \{\bar{\xi}_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \\
\underline{\epsilon}_{\mathcal{T}}(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) &= \epsilon_{\mathcal{T}}(1, \{\underline{\xi}_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)
\end{aligned}$$

Then, $\underline{\epsilon}_{\mathcal{T}}(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \leq \epsilon_{\mathcal{T}}(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \leq \bar{\epsilon}_{\mathcal{T}}(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)$.

Proposition 7.1 serves to demonstrate that λ takes effect in every source on the final adaptation performance. With the assumption that different source domains have different S - T similarities with the target domain, a single λ that works for every sources has a great difficulty capturing such S - T similarity diversity. This leads us to exploit the transfer covariance function in another way.

7.3 Transfer Covariance Function Stacking

Considering the effectiveness showed by the stacking strategy for multi-source adaptation regression (Pardoe and Stone, 2010), we propose a framework that can integrate the capability for S - T similarity capture of both the transfer covariance function and stacking. We first introduce $TC_{SS}Stack$, a conventional way of stacking the transfer covariance function. Then, we design a more involved stacking-inspired approach that overcomes some limitations of the conventional stacking method.

7.3.1 Conventional Transfer Stacking $TC_{SS}Stack$

Motivated by the fact that both the stacking strategy and the transfer covariance function can model the S - T similarity and using the transfer covariance function at the base models would therefore add flexibility to the similarity capture capability of the stacking approach, we propose a $TC_{SS}Stack$ method. In $TC_{SS}Stack$, we first train multiple GP - TC_{SS} models for each $\mathcal{S}_i$ and $\mathcal{T}$ (denoted as $\{f^{(\mathcal{S}_i, \mathcal{T})}(\cdot | \boldsymbol{\Omega}_i, \lambda_i)\}_{i=1}^N$) and then apply the conventional stacking strategy to combine their predictions. Given a target point $\mathbf{x}$, the final prediction is given by:

$$f(\mathbf{x}) = \sum_{i=1}^N \omega_i f^{(\mathcal{S}_i, \mathcal{T})}(\mathbf{x} | \boldsymbol{\Omega}_i, \lambda_i), \quad \sum_{i=1}^N \omega_i = 1 \quad (7.6)$$

where ω_i are coefficients learned by minimizing the least square error on the target labeled data.

There are two major issues for the above model. (1) Since each $f^{(\mathcal{S}_i, \mathcal{T})}$ is pre-trained separately, the parameters learned for each $f^{(\mathcal{S}_i, \mathcal{T})}$ do not take the inter-domain dependencies between different source domains into account. (2) Both λ_i and ω_i reflect the S - T domain similarity. However, $TC_{SS}Stack$ takes them as two different variables and learns them separately. Intuitively, the model importance ω_i should be positively correlated with the similarity coefficient λ_i . For example, the

prediction of a $GP\text{-}TC_{SS}$ using an unrelated source is less trustful, and should be assigned a smaller coefficient in the stacking.

7.3.2 Improved Transfer Stacking $TC_{MS}Stack$

To overcome the above issues, we design a new transfer stacking model ($TC_{MS}Stack$) as follows:

$$f^*(\mathbf{x}) = \sum_{i=1}^N (g(\lambda_i)/Z) f^{(S_i, T)}(\mathbf{x}, \boldsymbol{\Omega}_i, \lambda_i). \quad (7.7)$$

where λ_i refers to the similarity coefficient in the $GP\text{-}TC_{SS}$ for the i -th source, $Z = \sum_{i=1}^N g(\lambda_i)$ is the normalization term, and $g(\lambda_i)$ is any function preserving the monotonicity of $|\lambda_i|$ so that it coordinates the model importance and the similarity coefficient. This also reduces the search efforts by lowering the number of free parameters to fit. Moreover, instead of pretraining $f^{(S_i, T)}(\cdot | \boldsymbol{\Omega}_i, \lambda_i)$ separately, we jointly learn $f^{(S_i, T)}(\cdot, \boldsymbol{\Omega}_i, \lambda_i)$ for all the source domains. By doing so, the multiple $GP\text{-}TC_{SS}$ models are learned together with the dependencies between multiple sources taken into account.

Notice that the model in eq. (7.7) allows for multiple options to characterize the relative importance of $GP\text{-}TC_{SS}$ models through $g(\cdot)$. In this work, we use a simple function $g(\lambda_i) = |\lambda_i|$. However, the absolute value function is not smooth at the origin. Thus, we use a smooth function studied in (Yong, 2015) to approximate it as follows:

$$|\lambda_i| \approx \alpha \text{Ln} \left(\frac{1}{2} e^{\frac{\lambda_i}{\alpha}} + \frac{1}{2} e^{-\frac{\lambda_i}{\alpha}} \right).$$

We set $\alpha = 0.01$ which is the best approximation stated in (Yong, 2015). Since Theorem 1 also tells us $-1 \leq \lambda_i \leq 1$, we propose to define $\lambda_i = 2(1/(1 + \mu_i))^{b_i} - 1$ ($\mu_i \geq 0$ and $b_i \geq 0$), as in Cao et al. (2010). Then, we conduct the learning by minimizing the squared errors:

$$\min_{\{\boldsymbol{\Omega}_i, \mu_i, b_i\}_{i=1}^N} \sum_{j=1}^{n_{T_l}} (y_j^{(T_l)} - f^*(\mathbf{x}_j^{(T_l)}))^2. \quad (7.8)$$

In the optimization, we propose to use the conjugate gradient method. Other optimization methods can also be applied to solve this objective function.

Table 7.1: Amazon review products dataset 15.

Top category	Source domains	Target domain
Beauty_Health_Grocery	Beauty, Grocery_Food, Health	Clothing_Shoes_Jewelry
Electronics	Electronic, Office_Product, Kindle_Store	Cellphone_Accessory
Home_Garden_Tool	Kitchen, Pet_Supplies	Tools_Home_Improvements
Movies_Music_Game	CD_Vinyl, Digital_Music, Video_Games	Movies_TV

7.3.3 Complexity Analysis

As in usual *GP* model training, the computational time complexity of each $f^{(\mathcal{S}_i, \mathcal{T})}$ is dominated by the calculation of the inverse of its covariance matrix, i.e. $\mathcal{O}((n_{\mathcal{T}_i} + n_{\mathcal{S}_i})^3)$. Considering $n_{\mathcal{T}_i} \ll n_{\mathcal{S}_i}$ and assuming $n_{\mathcal{S}_1} = \dots = n_{\mathcal{S}_N} = n_{\mathcal{S}}$, the evaluation of a *TC_{MS}Stack* model takes then $\mathcal{O}(Nn_{\mathcal{S}}^3)$. Notice that by following the stacking strategy of eq. (7.6) the training involves the steps of learning each $f^{(\mathcal{S}_i, \mathcal{T})}$ and subsequently learning the ω_i coefficients. The latter calls for some cross-validation approach to evaluate a meta-model, as the $f^{(\mathcal{S}_i, \mathcal{T})}$ from the previous step have been induced using the target data Pardoe and Stone (2010). In the extreme case of leave-one-out, this would take $\mathcal{O}(n_{\mathcal{T}_i}Nn_{\mathcal{S}}^3)$. Even if we also choose a leave-one-out validation to solve eq. (7.8) the cost of a *TC_{MS}Stack* model evaluation would be lower, since the first step of stacking is not required. On the other side, by following TrAdaBoost.R2 or the *GP-TC_{MS}* approach, a *GP* model evaluation requires $\mathcal{O}((Nn_{\mathcal{S}})^3)$, which even exceeds the cost for *TC_{MS}Stack* using leave-one-out whenever $N > \sqrt{n_{\mathcal{T}_i}}$.

7.4 Experimental Study

In the following experimental study we aim at two main goals: (i) to assess the ability of *TC_{MS}Stack* in capturing inter-domain similarity, and (ii) to evaluate its predictive effectiveness compared to other approaches.

7.4.1 Experiment Setting

All the GPs herein build upon a standard squared exponential covariance function. The hyperparameters of each method are optimized using the conjugate gradient

implementation from the *gpml* package (Rasmussen and Nickisch, 2010). For each search, we allow a maximum of 200 evaluations. The reported results correspond to the model providing the best objective function value over 5 independent runs with random initial solutions each. We use one synthetic dataset and two real-world datasets.

Synthetic dataset. We consider a linear function $f(\mathbf{x}) = \mathbf{w}_0^T \mathbf{x} + \epsilon$, where $\mathbf{w}_0 \in \mathbb{R}^{100}$ and ϵ is a zero-mean Gaussian noise term, as the target. We use this function to generate 100 points as target test data, and 20 points as target train data. For the source task, we use $g(\mathbf{x}) = (\mathbf{w}_0^T + \delta \Delta \mathbf{w}) \mathbf{x} + \epsilon$, where $\Delta \mathbf{w}$ is a random fluctuation vector and δ is the variable controlling the similarity between f and g (higher δ indicates lower similarity), to generate 380 points for each source with different δ .

Amazon reviews. We extract the raw data containing 15 product reviews from (McAuley et al., 2015), and categorize the products into four top categories according to the Amazon website. Products in the same category are conceptually similar. Each product is taken as a domain, and we select one as target from each category (see Table 7.1). Reviews in each domain are represented by the count features and are labeled using stars in the set $\{1, 2, 3, 4, 5\}$.

UJIIndoorLoc. The building location dataset covers three buildings of Universitat Jaume I with four floors each (Torres-Sospedra et al., 2014). We build 12 domains by taking the location data from each floor of each building as a domain. The first floor of each building is taken as the target. Domains from the same building are taken as similar. The received signal strength intensity from 520 wireless access points is used as the features, and the location represented by the latitude and longitude is taken as label.

7.4.2 Domain Similarity Capture

We first elucidate the ability of $TC_{MS}Stack$ in capturing the diverse S - T similarities through the λ_i coefficients. To rationalize the assessment, we use the synthetic dataset and consider a variety of problems covering a broad spectrum of TL settings. Precisely, we build four scenarios of $N = 2, 5, 10, 15$ sources. In each scenario, we specify six problems, each given by a different combination of sources. Three problems represent settings in which all the sources are equally similar to the target,

with high ($\delta = 0$), medium ($\delta = 15$) and low ($\delta = 35$) similarity strength. The other three problems reflect diverse S - T similarities. Each source is given by a δ randomly sampled from the set $\{0, 4, 7, 10, 15, 20, 25, 30, 35\}$ and with replacement. We enforce the three problems to be different and avoid all the sources to be equal. We show the results in Figure 7.1. Bar plots on the left show the correspondence between each problem similarity (δ) and the similarity captured by the model (λ); bars for $TC_{MS}Stack$ and black dots for $GP-TC_{MS}$. Bar plots on the right show the RMSE for $TC_{MS}Stack$ (red) and $GP-TC_{MS}$ (green).

In the figure, the first three problems of each scenario are the cases with equal S - T similarities. It can be observed in the bar plots on the left hand side that the λ values learned by $TC_{MS}Stack$ are strictly reverse-correlated with the predefined δ values, which indicates an accurate capture of the high, medium and low strengths of S - T similarity. We further observe from the black dots that $GP-TC_{MS}$ is also able to strongly coordinate δ with a single compromised λ . This is because all the sources share the same δ with the target, and thus can be regarded as a single larger source.

The remaining three problems of each scenario reflect diverse similarities across source domains. In this case, Figure 7.1 shows that the λ values of $TC_{MS}Stack$ reflect the relative differences of δ across different sources fairly well in general. The learned λ is generally reverse-correlated with the predefined δ values, but it is not strict and tight. For instance, in problem 6 of the 5-sources scenario or in the problem 5 of the 15-sources scenario, such reverse-correlated relations do not hold in all the sources. This is because, although the learned λ s only represent the bivariate S - T similarities, each of them is specified during learning by considering its influence relative to the rest of similarity coefficients, i.e. the inter-domain dependencies between different sources are taken into account during the learning of the λ s. Thus, in some cases, the learned λ s may not strictly approximate the real S - T similarities. However, λ s are always learned to guarantee the outcome of the best adaptation performance. That is the reason why the above two cases still achieve satisfactory adaptation performance in terms of RMSE. By contrast, we find that $GP-TC_{MS}$ only gives a trade-off value of λ over the diverse δ values. The right hand side of the figure shows a consistently lower RMSE for $TC_{MS}Stack$ than for $GP-TC_{MS}$ in all the problems. In particular, a dramatic improvement is observed by utilizing $TC_{MS}Stack$ for the

Figure 7.1: Results on 6 problem settings for each scenario of 2, 5, 10 and 15 sources.

diverse problems 4-6. These results indicate the superiority of $TC_{MS}Stack$ over $GP-TC_{MS}$ for multi-source adaptation regression. We further verify this conclusion on the two real-world datasets in the next section.

7.4.3 Performance on Real-World Datasets

We compare $TC_{MS}Stack$ with several multi-source adaptation regression approaches, namely: Tradaboost.R2, $GP-TC_{MS}$, $TC_{SS}Stack$, a variant of $TC_{SS}Stack$ with joint learning of model importance coefficient and λ which we call $TC_{SS}Stack-Joint$, and a variant of $TC_{SS}Stack$ using the learned λ as the model importance which we call λ -Stacking. The evaluation comprises both the Amazon and the UJIIndoorLoc datasets. For each source domain we sample 500 points uniformly at random for training. Likewise, train and test data from each target domain are obtained by sampling 25 points and 1000 points, respectively. For the Amazon dataset, we generated a set of problems by using each target domain in Table 7.1, and by randomly choosing a number of source domains subsets. More precisely, for each scenario of 2, 3 and 5 sources, ten different source combinations were randomly constructed. In addition, a scenario of 11 sources with all the source domains was selected. Thus, we construct 40 transfer problems for each scenario of 2, 3 and 5 sources, and 4 problems for the scenario of 11 sources. For the UJIIndoorLoc dataset, we generate the transfer problems in a similar way to the Amazon dataset described above.

In Figure 7.2, we show the average RMSE results over all the problems in each scenario for the two datasets. Overall, $TC_{MS}Stack$ is the winner among all the baselines on the two datasets, improving the adaptation performance across the different amounts of source domains. This showcases the capability of $TC_{MS}Stack$ to transfer knowledge from various sources with different $S-T$ similarities.

For the other baselines, we observe that TrAdaBoost.R2 gives the poorest results due to the premature weights convergence issue. If, in addition, we consider the high computational cost for the models involved, TrAdaBoost.R2 does not seem to be a good choice for multi-source adaptation regression, especially when the number of source domains is large. As for $GP-TC_{MS}$, it presents a steadily inferior performance than $TC_{MS}Stack$. Overall, the outcomes are in line with those in the synthetic dataset, offering further support to the superiority of $TC_{MS}Stack$ to $GP-TC_{MS}$. Notice that, since the current benchmark was generated randomly, it is likely a scenario to comprise diverse problem settings. Therefore, capturing the diverse similarities through a single λ coefficient may compromise the performance of $GP-TC_{MS}$. In contrast, $TC_{MS}Stack$ offers more robust performance improvements.

Finally, the comparison with the other stacking-based methods exposes the ben-

Figure 7.2: Comparison results on the two datasets.

efits of the two salient features of $TC_{MS}Stack$. Both $TC_{SS}Stack$ and λ -Stacking are beaten by $TC_{SS}Stack$ -Joint and $TC_{MS}Stack$. Since these two sets of methods only differ in the joint learning of the parameters, the outcomes point at the benefits of bringing in the inter-domain dependencies of the other sources during the learning. On the other side, the results for λ -Stacking and $TC_{MS}Stack$ are better or comparable to those by $TC_{SS}Stack$ and $TC_{SS}Stack$ -Joint, respectively. This provides support to the correlation of the model importance with the similarity coefficients, which allows to specify the model by estimating fewer hyper-parameters while preserving the similarity capture capability.

7.5 Summary

We summarize the contributions of this chapter as follows:

- We theoretically demonstrate the pathology of a family of transfer covariance function (TC_{MS}) for multi-source adaptation regression problems. We prove that TC_{MS} gives a uniform similarity coefficient to all the source and target domain pairs. This is a compromised similarity, which cannot capture the diverse similarities between different source and target domain pairs.
- We develop a novel method $TC_{MS}Stack$ that integrates transfer covariance function with stacking for multi-source adaptation regression problems. We

use a parametric function to connect the model importance of a base model with the similarity coefficient in the base model since these two metrics are positively correlated. We also propose a joint learning strategy to take the dependence of different sources into account in the model learning.

- We test our proposed $TC_{MS}Stack$ on one synthetic dataset and two real-world datasets. The results on the synthetic dataset show that $TC_{MS}Stack$ can effectively capture the diverse similarities between different source and target domain pairs. The comparison results on the two real-world datasets demonstrate the superiority of $TC_{MS}Stack$ with respect to the adaptation performance compared to the state-of-the-art methods.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we study and investigate methods for different domain adaptation problem settings including unsupervised (homogeneous) domain adaptation problems, heterogeneous domain adaptation problems, and multi-source adaptation regression problems. In Chapters 3-5, we focus on unsupervised domain adaptation. We propose two deep feature learning methods in Chapter 3 and Chapter 4, and a subspace-based method in Chapter 5. In Chapter 6, we introduce a special heterogeneous domain adaptation problem setting, called hybrid domain adaptation, and design a general framework to efficiently solve such a new problem setting without using any supplementary information. In the last chapter, we work on a more challenging problem setting, where there are multiple source domains and the source and target tasks are heterogeneous. Figure 8.1 summarizes the conclusive remarks. More specifically, we conclude the thesis in details as follows.

In Chapter 3, we first investigate the rationale for why mSDA can bring benefits to domain adaptation. By analyzing feature corruptions using noise as adaptive regularization to the learned mapping, we demonstrate that mSDA with *Bernoulli dropout* noise (mSDA_{bd}), which is equivalent to regularizing the mapping with a data-dependent low-rank constraint and a data-dependent Frobenius constraint, succeeds to boost the adaptation performance, while mSDA with *Gaussian* noise (mSDA_g), which is equivalent to a Frobenius regularization on the mapping, fails to do so. Empirical evaluations sufficiently verify the theoretical analyses: mSDA_g

Figure 8.1: The Conclusive Remarks

yields comparable average results with the no adaptation baseline NN, while mSDA_{bd} achieves significant performance improvements. Then, we propose an mSDA with *Multinomial dropout* noise (mSDA_{md}), which preserves the regularization advantage of mSDA_{bd} and simultaneously overcomes limitations of mSDA_{bd} . On the one hand, mSDA_{md} assigns different corruption probabilities to different features and constraints the summation of all the probabilities to be 1. This is more realistic than mSDA_{bd} that assumes each feature is corrupted independently with an identical probability. On the other hand, we design a data-dependent manner to calculate different probabilities. It avoids the problematic CV process on the source domain to select hyperparameters in mSDA_{bd} . Experimental results show that mSDA_{md} achieves 6.38% average performance improvement compared to mSDA_{bd} .

In Chapter 4, we propose a deep nonlinear feature coding framework for unsupervised domain adaptation. It incorporates two new elements, MMD and kernelization, into mSDA to extract nonlinear deep features with minimum domain divergence. Extensive experimental studies show that DNFC achieves very promising results. It consistently and significantly outperforms all baselines in terms of prediction accuracy. Compared to the best baseline mSDA, DNFC attains up to 14% better accuracy and converges much faster in terms of the number of stacked

layers. The effects of MMD and kernelization on the performance of DNFC are also investigated. It is shown that DNFC benefits from both elements, as well as their joint effects. All the results demonstrate that DNFC is an effective and promising solution to unsupervised domain adaptation.

In Chapter 5, we develop a local manifolds information transfer (LMIT) framework for unsupervised domain adaptation. We propose to preserve the local multiple manifolds information in adaptation, and simultaneously reduce the distribution discrepancies of different domains. Experimental results have shown the superiority of LMIT to other state-of-the-art unsupervised domain adaptation methods. It achieves the best adaptation performance on various adaptation tasks. We further verify the effectiveness of LMIT by inspecting the neighbor-class structure of different spaces. All the experiments show us LMIT is an effective and promising method for unsupervised domain adaptation.

In Chapter 6, we investigate a new problem setting, hybrid domain adaptation, and design a general and effective domain specific feature transfer (DSFT) framework to transfer knowledge across domains without any supplementary information. It learns the translations between common features and domain specific features, and utilizes the learned translations to construct a homogeneous feature space, in which cross domain knowledge is transferred and the domain divergence is minimized. We instantiate the DSFT framework to a linear case and a nonlinear case, i.e., DSFT_l and DSFT_{nl} . Extensive experiments show that both DSFT_l and DSFT_{nl} outperform COM and F_Zero as well as their couplings with state-of-the-art methods. The underlying mechanisms of DSFT, feature reweighting and new feature representation, are empirically investigated. It is also shown that DSFT benefits from both the translation and discrepancy terms. The positive role that domain specific features play in domain adaptation is validated.

In Chapter 7, we investigate a family of transfer covariance functions that represent the pairwise similarity between each source and the target domain for the multi-source adaptation regression problem. We prove that, $GP\text{-}TC_{MS}$, a Gaussian process with such a transfer covariance function can only capture the same similarity coefficient for all the sources. By further analyzing the generalization errors of $GP\text{-}TC_{MS}$, we conclude the bounds depend on the single similarity coefficient, which may penalize the adaptation performance. As an alternative, we propose

$TC_{MS}Stack$, an approach that integrates the transfer covariance function and the stacking strategy into one unified model. $TC_{MS}Stack$ aligns the S - T similarity coefficients with the model importance and jointly learns the base models. Experiments on one synthetic and two real-world datasets, with learning settings of up to 11 sources for the latter, show the superiority of $TC_{MS}Stack$ to other multi-source adaptation regression methods.

8.2 Future Works

8.2.1 Theoretical Foundation

There are quite a lot feature-based domain adaptation methods (including deep feature learning methods and subspace-based methods) that have been proposed in recent years. Most of them share the similar idea of discovering similarities of underlying data properties including density, geometric structure, data sparsity, low-rank and higher statistical moments. Although intuitively appealing, none of these works give a clear picture on why a specific formulation of feature mapping can bring benefits to domain adaptation. For instance, enforcing a feature mapping to be low-rank is widely used in domain adaptation, and it has been empirically demonstrated a great success in specific tasks. However, the underlying mechanism for why the low-rank constraint is beneficial for domain adaptation is still unclear. Thus, it is much more desired to open the black box that encodes the relationships between the feature mapping formulation and the transfer performance.

As shown in (Ben-David et al., 2007), a good representation for transfer is the one that can reduce source training error and domain divergence. Thus, one way to answer the above question is to theoretically demonstrate that a specific feature mapping formulation enables the two principles stated in (Ben-David et al., 2007). To achieve this purpose, the key point is to build two links: one is between the learning of feature mapping and the learning of prediction model, and another one is between the learning of feature mapping and the domain divergence metric used. There are different ways to build the links. Rademacher complexity may be a good start point as it is not only used to represent model complexity in learning theory, but also plays a role in measuring domain divergence in domain adaptation. More technical details need to be further worked out.

8.2.2 Multi-source Adaptive Transfer

In Chapter 7, we introduce a $TC_{MS}Stack$ approach for multi-source adaptation regression. $TC_{MS}Stack$ aligns similarity coefficient with model importance and jointly learns base models. By doing so, it not only successfully captures the diverse domain similarities between different domain pairs, but also alleviates the computational issue. In the future work, we are considering some potential extensions of $TC_{MS}Stack$ to further improve its transfer performance.

One extension is to make $TC_{MS}Stack$ more flexible. In $TC_{MS}Stack$, an absolute value function is used to align similarity coefficient with model importance. This constraints the model importance to be exactly the same with the similarity coefficient, which is very restrictive. To be more flexible, a parametric function can be used to connect similarity coefficient and model importance. Function parameters are learned with the other model parameters. By doing so, the learned parametric function fits to the training data, and thus the relationship of similarity coefficient and model importance is more data-orientated. Another possible extension is to discover the underlying mechanism of $TC_{MS}Stack$. $TC_{MS}Stack$ is a stacking of several GPs, and thus it is still a GP parameterized by a mean function and an unknown transfer covariance function. It will be more interesting to investigate the form of the equivalent transfer covariance function on which $TC_{MS}Stack$ is built. More technical details need to be further worked out.

Bibliography

- Adams, N. (2010). Dataset shift in machine learning. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(1):274–274.
- Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., and Marchand, M. (2014). Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*.
- Al-Stouhi, S. and Reddy, C. (2011). Adaptive boosting for transfer learning using dynamic updates. *Machine Learning and Knowledge Discovery in Databases*, pages 60–75.
- Baktashmotlagh, M., Harandi, M., Lovell, B., and Salzmann, M. (2013). Unsupervised domain adaptation by domain invariant projection. In *ICCV*, pages 769–776.
- Batista, G. E. and Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533.
- Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al. (2007). Analysis of representations for domain adaptation. In *NIPS*, pages 137–144.
- Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.
- Bhatt, H. S., Rajkumar, A., and Roy, S.-r. (2016). Multi-source iterative adaptation for cross-domain classification. In *IJCAI*, pages 3691–3697.
- Bickel, S., Brückner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In *ICML*, pages 81–88.
- Bickel, S. and Scheffer, T. (2007). Dirichlet-enhanced spam filtering based on biased samples. In *NIPS*, pages 161–168.

- Bishop, C. M. (1995). Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *NIPS*, pages 129–136.
- Blitzer, J., Dredze, M., Pereira, F., et al. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, pages 440–447.
- Blitzer, J., Kakade, S., and Foster, D. P. (2011). Domain adaptation with coupled subspaces. In *International Conference on Artificial Intelligence and Statistics*, pages 173–181.
- Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128.
- Bonilla, E. V., Chai, K. M., and Williams, C. (2007). Multi-task gaussian process prediction. In *NIPS*, pages 153–160.
- Bonilla, E. V., Chai, K. M., and Williams, C. (2008). Multi-task gaussian process prediction. In *NIPS*, pages 153–160.
- Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):49–57.
- Cai, D., He, X., Wang, X., Bao, H., and Han, J. (2009). Locality preserving non-negative matrix factorization. In *IJCAI*, pages 1010–1015.
- Cao, B., Pan, S. J., Zhang, Y., Yeung, D.-Y., and Yang, Q. (2010). Adaptive transfer learning. In *AAAI*, pages 7–14.
- Chai, K. M. (2009). Generalization errors and learning curves for regression with multi-task gaussian processes. In *NIPS*, pages 279–287.

- Chelba, C. and Acero, A. (2006). Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.
- Chen, B., Lam, W., Tsang, I., and Wong, T.-L. (2009). Extracting discriminative concepts for domain adaptation in text mining. In *ACM KDD*, pages 179–188.
- Chen, M., Xu, Z., Weinberger, K., and Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. In *ICML*, pages 1627–1634.
- Clinchant, S., Csurka, G., and Chidlovskii, B. (2016). A domain adaptation regularization for denoising autoencoders. In *ACL*, pages 26–31.
- Cortes, C., Mohri, M., Riley, M., and Rostamizadeh, A. (2008). Sample selection bias correction theory. In *Algorithmic Learning Theory*, pages 38–53.
- Dai, W., Chen, Y., Xue, G.-R., Yang, Q., and Yu, Y. (2008). Translated learning: Transfer learning across different feature spaces. In *NIPS*, pages 353–360.
- Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007a). Co-clustering based classification for out-of-domain documents. In *ACM KDD*, pages 210–219.
- Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2007b). Boosting for transfer learning. In *ICML*, pages 193–200.
- Daume III, H. (2009). Frustratingly easy domain adaptation. In *ACL*, pages 256–263.
- Daume III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. *JAIR*, 26(1):101–126.
- Ding, Z., Shao, M., and Fu, Y. (2015). Deep low-rank coding for transfer learning. In *IJCAI*, pages 3453–3459.
- Duan, L., Xu, D., and Tsang, I. W. (2012). Learning with augmented features for heterogeneous domain adaptation. In *ICML*, pages 711–718.
- Elhamifar, E. and Vidal, R. (2009). Sparse subspace clustering. In *CVPR*, pages 2790–2797.

- Evgeniou, T. and Pontil, M. (2004). Regularized multi-task learning. In *ACM KDD*, pages 109–117.
- Fan, W., Davidson, I., Zadrozny, B., and Yu, P. S. (2005). An improved categorization of classifier’s sensitivity on sample selection bias. In *ICDM*, pages 4–12.
- Fang, M., Guo, Y., Zhang, X., and Li, X. (2015). Multi-source transfer learning based on label shared subspace. *Pattern Recognition Letters*, 51(5):101–106.
- Fazel, M., Hindi, H., and Boyd, S. P. (2001). A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference*, pages 4734–4739.
- Fernando, B., Habrard, A., Sebban, M., and Tuytelaars, T. (2013). Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, pages 2960–2967.
- Gani, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2015). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.
- Ganin, Y. and Lempitsky, V. (2014). Unsupervised domain adaptation by back-propagation. In *ICML*, pages 1180–1189.
- Ghosh, S., Jacobs, R., and Mavris, D. N. (2015). Multi-source surrogate modeling with bayesian hierarchical regression. In *AIAA Non-Deterministic Approaches Conference*, pages 1817–1829.
- Glorot, X., Bordes, A., and Bengio, Y.-s. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520.
- Gönen, M. and Margolin, A. A. (2014). Kernelized bayesian transfer learning. In *AAAI*, pages 677–682.
- Gong, B., Grauman, K., and Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML*, pages 222–230.
- Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073.

- Gopalan, R., Li, R., and Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, pages 999–1006.
- Gopalan, R., Li, R., and Chellappa, R. (2014). Unsupervised adaptation across domain shifts by generating intermediate data representations. *IEEE TPAMI*, 36(11):2288–2302.
- Gu, Q., Li, Z., Han, J., et al. (2011). Joint feature selection and subspace learning. In *IJCAI*, pages 1294–1301.
- He, X., Cai, D., Yan, S., and Zhang, H.-J. (2005). Neighborhood preserving embedding. In *ICCV*, pages 1208–1213.
- He, X. and Niyogi, P. (2004). Locality preserving projections. In *NIPS*, pages 153–160.
- Hu, J., Lu, J., and Tan, Y.-P. (2015). Deep transfer metric learning. In *CVPR*, pages 325–333.
- Huang, J., Gretton, A., Borgwardt, K. M., Schölkopf, B., and Smola, A. J. (2006). Correcting sample selection bias by unlabeled data. In *NIPS*, pages 601–608.
- Hubert Tsai, Y.-H., Yeh, Y.-R., and Frank Wang, Y.-C. (2016). Learning cross-domain landmarks for heterogeneous domain adaptation. In *CVPR*, pages 5081–5090.
- Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.
- Jhuo, I.-H., Liu, D., Lee, D., and Chang, S.-F. (2012). Robust visual domain adaptation with low-rank reconstruction. In *CVPR*, pages 2168–2175.
- Jiang, J. (2008). A literature survey on domain adaptation of statistical classifiers. URL: <http://sifaka.cs.wiuc.edu/jiang4/domainadaptation/survey>.
- Jiang, J. and Zhai, C. (2007). Instance weighting for domain adaptation in nlp. In *ACL*, pages 264–271.
- Jiang, W., Gao, H., Chung, F.-l., and Huang, H. (2016). The l2, 1-norm stacked robust autoencoders for domain adaptation. In *AAAI*, pages 1723–1729.

- Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209.
- Kulis, B., Saenko, K., and Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, pages 1785–1792.
- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.
- Lam, K. Y., Westrick, Z. M., Müller, C. L., Christiaen, L., and Bonneau, R. (2016). Fused regression for multi-source gene regulatory network inference. *PLOS Computational Biology*, 12:1–23.
- Lawrence, N. D. and Platt, J. C. (2004). Learning to learn with the informative vector machine. In *ICML*, pages 65–73.
- Levina, E. and Bickel, P. J. (2004). Maximum likelihood estimation of intrinsic dimension. In *NIPS*, pages 777–784.
- Li, J., Zhao, J., and Lu, K. (2016a). Joint feature selection and structure preservation for domain adaptation. In *IJCAI*, pages 1697–1703.
- Li, W., Duan, L., Xu, D., and Tsang, I. W. (2014). Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation. *IEEE TPAMI*, 36(6):1134–1148.
- Li, W., Xu, Z., Xu, D., Dai, D., and Van Gool, L. (2017). Domain generalization and adaptation using low rank exemplar svms. *IEEE TPAMI*, 40(5):1141–1127.
- Li, Z., Gong, B., and Yang, T. (2016b). Improved dropout for shallow and deep learning. In *NIPS*, pages 2523–2531.
- Liao, X., Xue, Y., and Carin, L. (2005). Logistic regression with an auxiliary data source. In *ICML*, pages 505–512.
- Long, M., Ding, G., Wang, J., Sun, J., Guo, Y., and Yu, P. S. (2013a). Transfer sparse coding for robust image representation. In *CVPR*, pages 407–414.

- Long, M. and Wang, J. (2015). Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105.
- Long, M., Wang, J., Ding, G., Sun, J., and Yu, P. S. (2013b). Transfer feature learning with joint distribution adaptation. In *ICCV*, pages 2200–2207.
- Long, M., Wang, J., Ding, G., Sun, J., and Yu, P. S. (2014a). Transfer joint matching for unsupervised domain adaptation. In *CVPR*, pages 1410–1417.
- Long, M., Wang, J., Ding, G.-g., Shen, D., and Yang, Q. (2014b). Transfer learning with graph co-regularization. *Knowledge and Data Engineering, IEEE Transactions on*, 26(7):1805–1818.
- Long, M., Wang, J., Sun, J., and Philip, S. Y. (2015). Domain invariant transfer kernel learning. *TKDE*, 27(6):1519–1532.
- Luo, P., Zhuang, F., Xiong, H., Xiong, Y., and He, Q. (2008). Transfer learning from multiple source domains via consensus regularization. In *ACM conference on Information and Knowledge Management*, pages 103–112.
- Luo, Z., Zou, Y., Hoffman, J., and Fei-Fei, L. F. (2017). Label efficient learning of transferable representations across domains and tasks. In *NIPS*, pages 164–176.
- Margolis, A. (2011). A literature review of domain adaptation with unlabeled data. In *Report Technique, University of Washington*, pages 1–42.
- McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52.
- Ming Harry Hsu, T., Yu Chen, W., Hou, C.-A., Hubert Tsai, Y.-H., Yeh, Y.-R., and Frank Wang, Y.-C. (2015). Unsupervised domain adaptation with imbalanced cross-domain data. In *ICCV*, pages 4121–4129.
- Motiian, S., Jones, Q., Iranmanesh, S., and Doretto, G. (2017). Few-shot adversarial domain adaptation. In *NIPS*, pages 6673–6683.
- Mozafari, A. S. and Jamzad, M. (2014). Heterogeneous domain adaptation using previously learned classifier for object detection problem. In *IEEE TIP*, pages 4077–4081.

- Nguyen, T. and Bonilla, E. (2014). Fast allocation of gaussian process experts. In *ICML*, pages 145–153.
- Niyogi, X. (2004). Locality preserving projections. In *NIPS*, volume 16, pages 153–160.
- Pan, S. J., Kwok, J. T., and Yang, Q. (2008). Transfer learning via dimensionality reduction. In *AAAI*, pages 677–682.
- Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2009). Domain adaptation via transfer component analysis. In *IJCAI*, pages 1187–1192.
- Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE TNNLS*, 22(2):199–210.
- Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE TKDE*, 22(10):1345–1359.
- Pardoe, D. and Stone, P. (2010). Boosting for regression transfer. In *ICML*, pages 863–870.
- Prettenhofer, P. and Stein, B. (2010). Cross-language text classification using structural correspondence learning. In *ACL*, pages 1118–1127.
- Rasmussen, C. E. (2006). Gaussian processes for machine learning. *Citeseer*.
- Rasmussen, C. E. and Nickisch, H. (2010). Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11(2):3011–3015.
- Ren, J., Shi, X., Fan, W., and Philip, S. Y. (2008). Type-independent correction of sample selection bias via structural discovery and re-balancing. In *SDM*, pages 565–576.
- Rifai, S., Glorot, X., Bengio, Y., and Vincent, P. (2011). Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*, pages 1–13.
- Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

- Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *International Conference on Computational Learning Theory*, pages 416–426.
- Schwaighofer, A., Tresp, V., and Yu, K. (2004). Learning gaussian process kernels via hierarchical bayes. In *NIPS*, pages 1209–1216.
- Shao, M., Castillo, C., Gu, Z., and Fu, Y. (2012). Low-rank transfer subspace learning. In *ICDM*, pages 1104–1109.
- Shao, M., Kit, D., and Fu, Y. (2014). Generalized transfer subspace learning through low-rank constraint. *International Journal of Computer Vision*, 109(1-2):74–93.
- Shi, X., Liu, Q., Fan, W., Yu, P. S., and Zhu, R. (2010). Transfer learning on heterogenous feature spaces via spectral transformation. In *ICDM*, pages 1049–1054.
- Shu, L., Ma, T., and Latecki, L. J. (2014). Locality preserving projection for domain adaptation with multi-objective learning. In *AAAI*, pages 2085–2091.
- Si, S., Tao, D., and Geng, B. (2010). Bregman divergence-based regularization for transfer subspace learning. *IEEE TKDE*, 22(7):929–942.
- Sugiyama, M., Krauledat, M., and Mäzller, K.-R. (2007). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5):985–1005.
- Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P. V., and Kawanabe, M. (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *NIPS*, pages 1433–1440.
- Sun, B., Feng, J., and Saenko, K. (2016). Return of frustratingly easy domain adaptation. In *AAAI*, pages 2058–2065.
- Tierney, S., Guo, Y., and Gao, J. (2015). Subspace clustering for sequential data. In *CVPR*, pages 1019–1026.
- Tommasi, T., Orabona, F., and Caputo, B. (2014). Learning categories from few examples with multi model knowledge transfer. *IEEE TPAMI*, 36(5):928–941.

- Torres-Sospedra, J., Montoliu, R., Usó, A.-I. M., Avariento, J. P., Arnau, T. J., Benedito-Bordonau, M., and Huerta, J. (2014). Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 261–270.
- Tsai, Y.-H. H., Hou, C.-A., Chen, W.-Y., Yeh, Y.-R., and Wang, Y.-C. F. (2016). Domain-constraint transfer coding for imbalanced unsupervised domain adaptation. In *AAAI*, pages 3597–3603.
- Uguroglu, S. and Carbonell, J. (2011). Feature selection for transfer learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 430–442.
- Vidal, R. (2011). Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68.
- Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In *NIPS*, pages 351–359.
- Wan, C., Pan, R., and Li, J. (2011). Bi-weighting domain adaptation for cross-language text classification. In *IJCAI*, pages 1535–1541.
- Wang, C. and Mahadevan, S. (2011). Heterogeneous domain adaptation using manifold alignment. In *IJCAI*, pages 1541–1548.
- Wang, H., Wang, W., Zhang, C., and Xu, F. (2014a). Cross-domain metric learning based on information theory. In *AAAI*, pages 2099–2105.
- Wang, H., Wang, W., Zhang, C., and Xu, F. (2014b). Cross-domain metric learning based on information theory. In *AAAI*, pages 2099–2105.
- Wang, K., He, R., Wang, L., Wang, W., and Tan, T. (2016). Joint feature selection and subspace learning for cross-modal retrieval. *IEEE TPAMI*, 38(10):2010–2023.
- Wang, Q., Ruan, L., and Si, L. (2014c). Adaptive knowledge transfer for multiple instance learning in image classification. In *AAAI*, pages 1334–1340.
- Wang, W., Wang, H., Zhang, C., and Xu, F. (2015). Transfer feature representation via multiple kernel learning. In *AAAI*, pages 3073–3079.

- Williams, C., Klanke, S., Vijayakumar, S.-t., and Chai, K. M. (2009). Multi-task gaussian process learning of robot inverse dynamics. In *NIPS*, pages 265–272.
- Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.
- Wu, F. and Huang, Y. (2016). Sentiment domain adaptation with multiple sources. In *ACL*, pages 301–310.
- Wu, P. and Dietterich, T. G. (2004). Improving svm accuracy by training on auxiliary data sources. In *ICML*, pages 110–118.
- Wu, X., Wang, H., Liu, C., and Jia, Y. (2013). Cross-view action recognition over heterogeneous feature spaces. In *ICCV*, pages 609–616.
- Xiao, M. and Guo, Y. (2015). Feature space independent semi-supervised domain adaptation via kernel matching. *IEEE TPAMI*, 37(1):54–66.
- Yang, Q., Chen, Y., Xue, G.-R., Dai, W., and Yu, Y. (2009). Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the Joint Conference of ACL and the International Joint Conference on Natural Language Processing of the the Asian Federation of Natural Language Processing*, pages 1–9.
- Yang, Z., Salakhutdinov, R., and Cohen, W. W. (2017). Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*, pages 1–10.
- Yao, Y. and Doretto, G. (2010). Boosting for transfer learning with multiple sources. In *Computer vision and pattern recognition, 2010 IEEE conference on*, pages 1855–1862.
- Yong, L. (2015). Uniform smooth approximation functions for absolute value function. *Mathematics in Practice and Theory*, pages 250–255.
- Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328.
- Yuta, T., Hisashi, K., Shohei, H., Steffen, B., and Masashi, S. (2009). Direct density ratio estimation for large-scale covariate shift adaptation. *Information and Media Technologies*, 50(4):1347–1364.

- Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *ICML*, pages 114–122.
- Zhang, F. (2006). *The Schur complement and its applications*, volume 4. Springer Science & Business Media.
- Zhang, X., Yu, F. X., Chang, S.-F., and Wang, S. (2015). Deep transfer network: Unsupervised domain adaptation. *arXiv preprint arXiv:1503.00591*, pages 1–7.
- Zhong, E., Fan, W., Yang, Q., Verscheure, O., and Ren, J. (2010). Cross validation framework to choose amongst models and datasets for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562.
- Zhou, J. T., Pan, S. J., Tsang, I.-o. W., and Yan, Y. (2014a). Hybrid heterogeneous transfer learning through deep learning. In *AAAI*, pages 2213–2220.
- Zhou, J. T., Pan, S. J., Tsang, I. W., and Ho, S.-S. (2016). Transfer learning for cross-language text categorization through active correspondences construction. In *AAAI*, pages 2400–2406.
- Zhou, J. T., Tsang, I. W., Pan, S. J., and Tan, M. (2014b). Heterogeneous domain adaptation for multiple classes. In *International Conference on Artificial Intelligence and Statistics*, pages 1095–1103.
- Zhu, Y., Chen, Y., Lu, Z., Pan, S. J., Xue, G.-R., Yu, Y., and Yang, Q. (2011). Heterogeneous transfer learning for image classification. In *AAAI*, pages 1304–1309.
- Zhuang, F., Cheng, X., Luo, P., Pan, S. J., and He, Q. (2015a). Supervised representation learning: transfer learning with deep autoencoders. In *IJCAI*, pages 4119–4125.
- Zhuang, F., Luo, D., Jin, X., Xiong, H., Luo, P., and He, Q. (2015b). Representation learning via semi-supervised autoencoder for multi-task learning. In *ICDM*, pages 1141–1146.

Appendix A

Proofs

Proof of Theorem 7.1:

Proof. Necessary condition: Let $\mathbf{K}_*$ be a *PSD* matrix. We use $\mathbf{K}_{SS}$ to represent the sources-to-sources block matrix, and $\mathbf{K}_{ST}$, $\mathbf{K}_{ST*}$ to represent the sources-to-target block matrix in $\mathbf{K}$ and $\mathbf{K}_*$, respectively. Thus, we have:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{SS} & \mathbf{K}_{ST} \\ \mathbf{K}_{ST}^\top & \mathbf{K}_{TT} \end{bmatrix}, \mathbf{K}_* = \begin{bmatrix} \mathbf{K}_{SS} & \mathbf{K}_{ST*} \\ \mathbf{K}_{ST*}^\top & \mathbf{K}_{TT} \end{bmatrix}.$$

Since $\mathbf{K}$ is *PSD*, according to the Schur complement theorem (Zhang, 2006), we have:

$$(\mathbf{I} - \mathbf{K}_{SS}\tilde{\mathbf{K}}_{SS})\mathbf{K}_{ST} = \mathbf{0}, \quad (\text{A.1})$$

$$\mathbf{K}_{TT} - \mathbf{K}_{ST}^\top \tilde{\mathbf{K}}_{SS} \mathbf{K}_{ST} \succeq \mathbf{0}, \quad (\text{A.2})$$

where $\tilde{\mathbf{K}}_{SS}$ is the generalized inverse of $\mathbf{K}_{SS}$. By rewriting $\tilde{\mathbf{K}}_{SS}$ as a block matrix using $\tilde{\mathbf{K}}_{S_i S_j}$ as the element, we further derive eq. (A.1) and eq. (A.2) as:

$$\begin{bmatrix} \mathbf{K}_{S_1 T} - \sum_{i=1}^N \sum_{j=1}^N \mathbf{K}_{S_1 S_i} \tilde{\mathbf{K}}_{S_i S_j} \mathbf{K}_{S_j T} \\ \dots \\ \mathbf{K}_{S_N T} - \sum_{i=1}^N \sum_{j=1}^N \mathbf{K}_{S_N S_i} \tilde{\mathbf{K}}_{S_i S_j} \mathbf{K}_{S_j T} \end{bmatrix} = \mathbf{0}, \quad (\text{A.3})$$

$$\mathbf{K}_{TT} - \sum_{i=1}^N \sum_{j=1}^N \mathbf{K}_{TS_i} \tilde{\mathbf{K}}_{S_i S_j} \mathbf{K}_{S_j T} \succeq \mathbf{0}. \quad (\text{A.4})$$

Likewise, for the *PSD* matrix $\mathbf{K}_*$, we have the following two Schur complement derivations:

$$\begin{bmatrix} \lambda_1 \mathbf{K}_{\mathcal{S}_1 \mathcal{T}} - \sum_{i=1}^N \sum_{j=1}^N \lambda_j \mathbf{K}_{\mathcal{S}_1 \mathcal{S}_i} \tilde{\mathbf{K}}_{\mathcal{S}_i \mathcal{S}_j} \mathbf{K}_{\mathcal{S}_j \mathcal{T}} \\ \dots \\ \lambda_N \mathbf{K}_{\mathcal{S}_N \mathcal{T}} - \sum_{i=1}^N \sum_{j=1}^N \lambda_j \mathbf{K}_{\mathcal{S}_N \mathcal{S}_i} \tilde{\mathbf{K}}_{\mathcal{S}_i \mathcal{S}_j} \mathbf{K}_{\mathcal{S}_j \mathcal{T}} \end{bmatrix} = \mathbf{0}. \quad (\text{A.5})$$

$$\mathbf{K}_{\mathcal{T} \mathcal{T}} - \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j \mathbf{K}_{\mathcal{T} \mathcal{S}_i} \tilde{\mathbf{K}}_{\mathcal{S}_i \mathcal{S}_j} \mathbf{K}_{\mathcal{S}_j \mathcal{T}} \succeq \mathbf{0}. \quad (\text{A.6})$$

Combining eq. (A.3) and eq. (A.5), we get:

$$\begin{bmatrix} \sum_{i=1}^N \sum_{j=1}^N (\lambda_1 - \lambda_j) \mathbf{K}_{\mathcal{S}_1 \mathcal{S}_i} \tilde{\mathbf{K}}_{\mathcal{S}_i \mathcal{S}_j} \mathbf{K}_{\mathcal{S}_j \mathcal{T}} \\ \dots \\ \sum_{i=1}^N \sum_{j=1}^N (\lambda_N - \lambda_j) \mathbf{K}_{\mathcal{S}_N \mathcal{S}_i} \tilde{\mathbf{K}}_{\mathcal{S}_i \mathcal{S}_j} \mathbf{K}_{\mathcal{S}_j \mathcal{T}} \end{bmatrix} = \mathbf{0}. \quad (\text{A.7})$$

Since Eq. (A.7) must hold for all *PSD* $\mathbf{K}$, we induce $\lambda_1 = \dots = \lambda_N = \lambda$. Based on such conclusion, we combine eq. (A.4) and eq. (A.6):

$$(1 - \lambda^2) \mathbf{K}_{\mathcal{T} \mathcal{T}} + \lambda^2 \mathbf{M} \succeq \mathbf{0}, \quad (\text{A.8})$$

where $\mathbf{M} = \mathbf{K}_{\mathcal{T} \mathcal{T}} - \sum_{i=1}^N \sum_{j=1}^N \mathbf{K}_{\mathcal{T} \mathcal{S}_i} \tilde{\mathbf{K}}_{\mathcal{S}_i \mathcal{S}_j} \mathbf{K}_{\mathcal{S}_j \mathcal{T}}$. Since eq. (A.8) must hold for all *PSD* $\mathbf{K}_{\mathcal{T} \mathcal{T}}$ and *PSD* $\mathbf{M}$, we resolve that $|\lambda| \leq 1$.

Sufficient condition: Let $\lambda_1 = \dots = \lambda_N = \lambda$, and $|\lambda| \leq 1$. According to the Theorem 1 in Cao et al. (2010), we obtain that $\mathbf{K}_*$ is a *PSD* matrix.

To sum up, we conclude that if $\mathbf{K}_*$ is a *PSD* matrix, λ_i should satisfy $\lambda_1 = \dots = \lambda_N$ and $|\lambda_i| \leq 1$. $\square$

Proof of Proposition 7.1:

Proof. By applying eq. (7.4), we have:

$$\begin{aligned} \Delta &= \delta_{\mathcal{T}}^2(1, \{\xi_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) - \delta_{\mathcal{T}}^2(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \\ &= \mathbf{k}_t^T [\Phi(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)^{-1} - \Phi(1, \{\xi_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)^{-1}] \mathbf{k}_t \end{aligned}$$

To make $\Delta \geq 0$ for all the target data points, we need to prove $\Phi(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)^{-1} - \Phi(1, \{\xi_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)^{-1}$ is PSD, which means:

$$\Phi(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)^{-1} - \Phi(1, \{\xi_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)^{-1} \succeq \mathbf{0}$$

$$\begin{aligned}
&\Longleftrightarrow \Phi(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \preceq \Phi(1, \{\xi_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2) \\
&\Longleftrightarrow \begin{bmatrix} (1 - \lambda^{-2})\mathbf{K}_{\mathcal{S}\mathcal{S}} + (\boldsymbol{\Sigma}'_{\mathcal{S}} - \lambda^{-2}\boldsymbol{\Sigma}_{\mathcal{S}}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \succeq \mathbf{0}
\end{aligned}$$

where $\boldsymbol{\Sigma}'_{\mathcal{S}}$ is a diagonal block matrix with the following diagonal block elements $\{\xi_{\mathcal{S}_1}^2 \mathbf{I}_{\mathcal{S}_1 \mathcal{S}_1}, \dots, \xi_{\mathcal{S}_N}^2 \mathbf{I}_{\mathcal{S}_N \mathcal{S}_N}\}$

$$\begin{aligned}
&\Longleftrightarrow (1 - \lambda^{-2})\mathbf{K}_{\mathcal{S}\mathcal{S}} + (\boldsymbol{\Sigma}'_{\mathcal{S}} - \lambda^{-2}\boldsymbol{\Sigma}_{\mathcal{S}}) \succeq \mathbf{0} \\
&\Longleftrightarrow \mathbf{K}_{\mathcal{S}\mathcal{S}} \preceq \frac{(\lambda^{-2}\boldsymbol{\Sigma}_{\mathcal{S}} - \boldsymbol{\Sigma}'_{\mathcal{S}})}{(1 - \lambda^{-2})} \\
&\Longleftrightarrow \bar{\delta} \leq \frac{(\lambda^{-2}\sigma_{\mathcal{S}_i}^2 - \xi_{\mathcal{S}_i}^2)}{(1 - \lambda^{-2})} \text{ for every } \mathcal{S}_i \\
&\Longleftrightarrow \xi_{\mathcal{S}_i}^2 \geq \lambda^{-2}\sigma_{\mathcal{S}_i}^2 - (1 - \lambda^{-2})\bar{\delta} \text{ for every } \mathcal{S}_i
\end{aligned}$$

Note that Δ is a monotonically increasing function of $\xi_{\mathcal{S}_i}^2$, thus we take the minimum $\lambda^{-2}\sigma_{\mathcal{S}_i}^2 - (1 - \lambda^{-2})\bar{\delta}$ as $\bar{\xi}_{\mathcal{S}_i}^2$ to be the smallest upper bound of $\sigma_{\mathcal{T}}^2(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)$. Similarly, we have $\underline{\xi}_{\mathcal{S}_i}^2 = \lambda^{-2}\sigma_{\mathcal{S}_i}^2 - (1 - \lambda^{-2})\underline{\delta}$ to construct the largest lower bound of $\sigma_{\mathcal{T}}^2(\lambda, \{\sigma_{\mathcal{S}_i}^2\}_{i=1}^N, \sigma_{\mathcal{T}}^2)$. $\square$

Appendix B

Publication List

- **Pengfei Wei**, Yiping Ke, Chi Keong Goh. “Deep Nonlinear Feature Coding for Unsupervised Domain Adaptation”, International Joint Conference on Artificial Intelligence (IJCAI), 2016, 2189-2195
- **Pengfei Wei**, Ramon Sagarna, Yiping Ke, Yew Soon Ong, Chi Keong Goh. “Source-Target Similarity Modelings for Multi-Source Transfer Gaussian Process Regression”, International Conference on Machine Learning (ICML), 2017, 3722-3731
- **Pengfei Wei**, Yiping Ke, Chi Keong Goh. “Domain Specific Feature Transfer for Hybrid Domain Adaptation”, International Conference on Data Mining (ICDM), 2017, 1027-1032
- **Pengfei Wei**, Yiping Ke, Chi Keong Goh. “Feature Analysis of Marginalized Stacked Denoising Autoencoder for Unsupervised Domain Adaptation”, IEEE Transaction on Neural Networks and Learning Systems (TNNLS), accepted
- **Pengfei Wei**, Yiping Ke, Chi Keong Goh. “A General Domain Specific Feature Transfer Framework for Hybrid Domain Adaptation”, IEEE Transaction on Knowledge and Data Engineering (TKDE), accepted
- **Pengfei Wei**, Ramon Sagarna, Yiping Ke, Yew Soon Ong. “Uncluttered Domain Sub-similarity Modeling for Transfer Regression”, International Conference on Data Mining (ICDM), 2018, accepted

- Tianbo Li, ***Pengfei Wei***, Yiping Ke. “Transfer Hawkes Processes with Content Information”, International Conference on Data Mining (ICDM), 2018, accepted

,