Active Transfer Learning

Xuezhi Wang

CMU-CS-16-119

June 2016

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Jeff Schneider, Chair
Christos Faloutsos
Geoff Gordon
Jerry Zhu, University of Wisconsin-Madison

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2016 Xuezhi Wang

This research was sponsored by the US Department of Agriculture under grant 20123702119958; Air Force Research Laboratory under grant numbers FA-875010C0210 and FA-87501220324; National Science Foundation under grant IIS0911032; and BAE Systems under grant 739529.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Transfer Learning, Active Learning, Model Shift, Multi-Task Learning, Stability Analysis, Distribution Learning

Abstract

Transfer learning algorithms are used when one has sufficient training data for one supervised learning task (the source task) but only very limited training data for a second task (the target task) that is similar but not identical to the first. These algorithms use varying assumptions about the similarity between the tasks to carry information from the source to the target task. Common assumptions are that only certain specific marginal or conditional distributions have changed while all else remains the same. Moreover, not much work on transfer learning has considered the case when a few labels in the test domain are available. Alternatively, if one has only the target task, but also has the ability to choose a limited amount of additional training data to collect, then active learning algorithms are used to make choices which will most improve performance on the target task. These algorithms may be combined into active transfer learning, but previous efforts have had to apply the two methods in sequence or use restrictive transfer assumptions.

This thesis focuses on active transfer learning under the model shift assumption. We start by proposing two transfer learning algorithms that allow changes in all marginal and conditional distributions but assume the changes are smooth in order to achieve transfer between the tasks. We then propose an active learning algorithm for the second method that yields a combined active transfer learning algorithm. By analyzing the risk bounds for the proposed transfer learning algorithms, we show that when the conditional distribution changes, we are able to obtain a generalization error bound of $O(\frac{1}{\lambda_* \sqrt{n_l}})$ with respect to the labeled target sample size n_l, modified by the smoothness of the change (λ_*) across domains. Our analysis also sheds light on conditions when transfer learning works better than no-transfer learning (learning by labeled target data only). Furthermore, we consider a general case where both the support and the model change across domains. We transform both X (features) and Y (labels) by a parameterized-location-scale shift to achieve transfer between tasks.

On the other hand, multi-task learning attempts to simultaneously leverage data from multiple domains in order to estimate related functions on each domain. Similar to transfer learning, multi-task problems are also solved by imposing some kind of “smooth” relationship among/between tasks. We study how different smoothness assumptions on task relations affect the upper bounds of algorithms proposed for these problems under different settings.

Finally, we propose methods to predict the entire distribution $P(Y)$ and $P(Y|X)$ by transfer, while allowing both marginal and conditional distributions to change. Moreover, we extend this framework to multi-source distribution transfer.

We demonstrate the effectiveness of our methods on both synthetic examples and real-world applications, including yield estimation on the grape image dataset, predicting air-quality from Weibo posts for cities, predicting whether a robot successfully climbs over an obstacle, examination score prediction for schools, and location prediction for taxis.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Jeff Schneider, who has been the biggest help during my whole PhD life. His brilliant insights have helped me formulate the problems of this thesis, brainstorm on new ideas and exciting algorithms. I have learnt many things about research from him, including how to organize ideas in a paper, how to design experiments, and how to give a good academic talk. This thesis would not have been possible without his guidance, advice, patience and encouragement.

I would like to thank my thesis committee members Christos Faloutsos, Geoff Gordon and Jerry Zhu for providing great insights and feedbacks on my thesis. Christos has been very nice and he always finds time to talk to me even if he is very busy. Geoff has provided great insights on extending my work to classification and helped me clarified many notations/descriptions in my thesis. Jerry has been very helpful in extending my work on the text data and providing me the air quality dataset. I feel very fortunate to have them as my committee members. I would also like to thank Professor Barnabás Póczos, Professor Roman Garnett and Professor Artur Dubrawski, for providing very helpful suggestions and collaborations during my PhD.

I am very grateful to many of the faculty members at Carnegie Mellon. Eric Xing's Machine Learning course has been my introduction course for Machine Learning at Carnegie Mellon and it has taught me a lot about the foundations of machine learning, including all the inspiring machine learning algorithms and the theories behind them. Larry Wasserman's Intermediate Statistics and Statistical Machine Learning are both wonderful courses and have been keys to my understanding of the statistical perspective of many machine learning algorithms. Geoff Gordon and Ryan Tibshirani's Convex Optimization course has been a great tutorial for me to develop all the efficient optimizing techniques for the algorithms I have proposed.

Further I want to thank all my colleagues and friends at Carnegie Mellon, especially people from the Auton Lab and the Computer Science Department at CMU. I would like to thank Dougal Sutherland, Yifei Ma, Junier Oliva, Tzu-Kuo Huang for insightful discussions and advices for my research. I would also like to thank all my friends who have provided great support and help during my stay at Carnegie Mellon, and to name a few, Nan Li, Junchen Jiang, Guangyu Xia, Zi Yang, Yixin Luo, Lei Li, Lin Xiao, Liu Liu, Yi Zhang, Liang Xiong, Ligia Nistor, Kirthevasan Kandasamy, Madalina Fiterau, Donghan Wang, Yuandong Tian, Brian Coltin.

I would also like to thank Prof. Alon Halevy, who has been a great mentor during my summer internship at google research and also has been a great help in my job searching process.

Finally I would like to thank my family, my parents Sisi and Tiangui, for their unconditional love, endless support, and unwavering faith in me. I truly thank them for shaping who I am, for teaching me to be a person who would never lose hope and give up.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Transfer Learning and Active Learning

As machine learning is applied to a growing number of domains, many researchers have looked to exploit similarities in machine learning tasks in order to increase performance. For example, one may suspect that data for the classification of one commodity as profitable or not may help in classifying a different commodity. Similarly, it is likely that data for spam classification in one language can help spam classification in another language. Transfer learning is a common technique for leveraging data from different domains for machine learning tasks.

Let X denote the features and Y denote the labels. In a classical transfer learning setting, we have sufficient fully labeled data from the source domain (also denoted as the training domain), (X^s, Y^s), where the data points, X^s, are fully observed and all corresponding labels, Y^s, are also known. We are given data points, X^t, from the target domain (also denoted as the test domain), but few or none of the corresponding labels, Y^t, are given. The source domain and the target domain are related but not identical, thus the joint distributions, $P(X^s, Y^s)$ and $P(X^t, Y^t)$, are different across the two domains. Without any transfer learning, a statistical model learned from the source domain does not directly apply to the target domain. However, it may still be possible to avoid the cost of collecting an entire new labeled training data set. The goal of transfer learning is to reduce the amount of new labeled data needed in the target domain. It learns and transfers a model based on the labeled data from the source domain and the unlabeled data from the target domain. Some real-world applications of transfer learning include adapting a classification model that is trained on some products to help learn classification models for some other products [51], and learning a model on the medical data for one disease and transferring it to another disease.

Alternatively, if one has only the target task but also has the ability to choose a limited amount of additional training data to collect, then active learning algorithms are used to make choices which will most improve performance on the target task. Common active learning criteria include: (1) Active Learning which reduces total predictive covariance [35, 61]; and (2) Active Surveying [22, 41, 72] where the objective is the sum of all the labels in the test domain. These algorithms can be combined into active transfer learning, but previous efforts have had to apply the two methods in sequence ([54, 62]) or use restrictive transfer assumptions ([13, 59]).

1.2 Motivation and Applications

We are motivated by an autonomous agriculture application where we want to manage the growth of grapes in a vineyard [48]. A robot can easily take images of the crop throughout the growing season. At the end of the season the yield will be known for every vine because the product is weighed after harvest. This data can be used to learn a model that predicts yield from images. However, decisions about selling the produce and nurturing the growth must be made mid-season. Acquiring training labels at that time is very expensive because it requires a human to go out and manually estimate yield. Ideally, a model learned from previous years and/or on other grape varieties can be used with a transfer learning algorithm to minimize this manual yield estimation. Furthermore, we would like a simultaneously applied active learning algorithm to tell us which vines to assess manually. Finally, there are two different objectives of interest. A robot that needs to decide which vines to water needs an accurate estimate of the current yield of each vine. However, a farmer that wants to know how big his crop will be this fall so he can pre-sell an appropriate amount of it only needs an estimate of the sum of the vine yields. We call these problems active learning and active surveying respectively and they lead to different selection criteria.

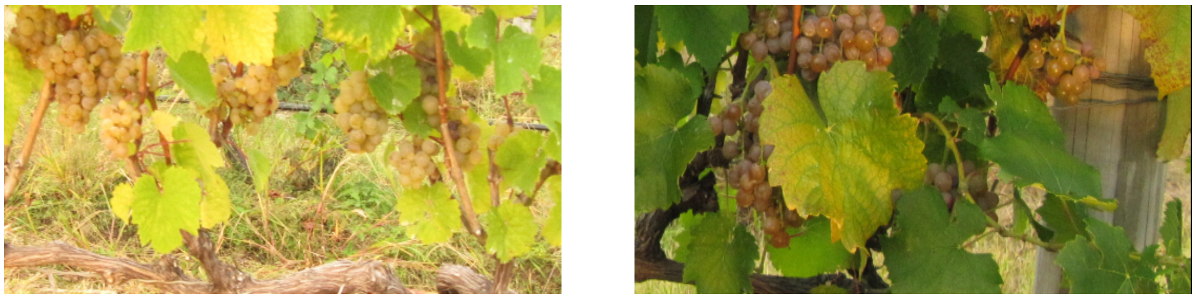


Figure 1.1: A part of one image from each grape dataset (riesling and traminette)

To better illustrate the problem, a part of one image from two real-world grape datasets (riesling and traminette) is shown in Figure 1.1. The goal is to transfer the model learned from one kind of grape dataset to another kind of grape dataset. On the traminette dataset we have achieved a cross-validated R-squared correlation of 0.754. Previously specifically designed image processing methods have achieved an R-squared correlation 0.73 [48]. This grape-detection method takes lots of manual labeling work and cannot be directly applied across different varieties of grapes (due to differences in size and color). Our proposed approach for transfer learning, however, can be directly used for different varieties of grapes or even different kinds of crops.

The second real-world example is learning to climb over obstacles with a snake robot (Figure 1.2). The task is to predict whether the robot successfully climbs over an obstacle with different heights. The parameters for the robot controller are closely connected for the climbing tasks with similar heights. Hence we can transfer the model we learned from the data of a robot snake climbing over an obstacle with a lower height to assist the learning of success rate when the robot tries to climb over a higher obstacle.

The third real-world example is predicting the Air Quality Index (AQI) for Chinese cities. The dataset is from [45] and for each city, the input feature x_i is a bag-of-words vector extracted

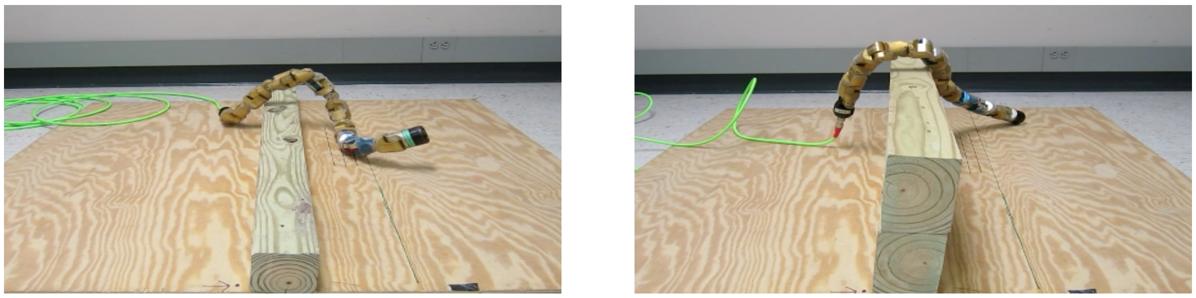
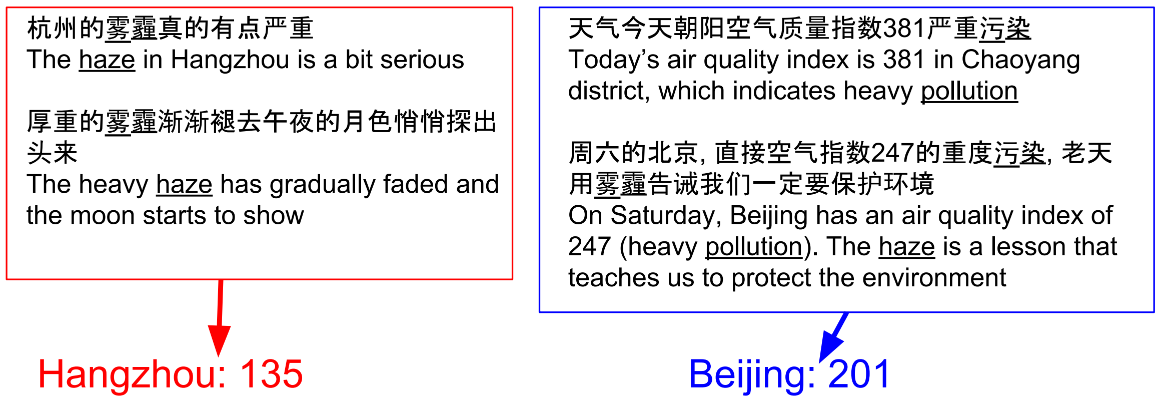


Figure 1.2: Learning to climb over obstacles with a snake robot, obstacle height: 3.5” (left) and 11”(right) [69]

from Weibo posts of each day, with $100,395$ dimensions as the dictionary size. The output label y_i is a real number which is the AQI of that day. Figure 1.3 shows an example of the Weibo posts and the corresponding AQI for that day, for two major cities in China. First we observe from the data that there exist certain words (e.g., the underlined words in the example) which have positive correlations to the AQI. Second, we also discover that, for different cities, even people use the same words to describe bad air qualities, the actual AQI could still be quite different. For example, assume a city A has an overall better air quality than the other city B. On a certain day, if the air quality of city A becomes slightly worse than before, people might start complaining a lot on Weibo posts. However the AQI for city A which people consider as “hazardous” might just be “moderate” for people in city B. In this thesis, we will show how we use our proposed transfer learning methods to resolve this issue.



杭州的雾霾真的有点严重
The haze in Hangzhou is a bit serious

厚重的雾霾渐渐褪去午夜的月色悄悄探出头来
The heavy haze has gradually faded and the moon starts to show

天气今天朝阳空气质量指数381严重污染
Today’s air quality index is 381 in Chaoyang district, which indicates heavy pollution

周六的北京, 直接空气指数247的重度污染, 老天用雾霾告诫我们一定要保护环境
On Saturday, Beijing has an air quality index of 247 (heavy pollution). The haze is a lesson that teaches us to protect the environment

Figure 1.3: Predicting Air Quality Index from Text posts

The fourth real-world example is predicting the examination scores for schools. The dataset come from the Inner London Education Authority (ILEA), consisting of examination records from 139 secondary schools in years 1985, 1986 and 1987 [1] [4, 6]. It is a random 50% sample with 15362 students. The data have been used to study the effectiveness of schools. Table 1.1 shows an example of the school features. If we regard each school as one task, how to jointly

[1]http://cvn.ecp.fr/personnel/andreas/code/mtl/index.html

leverage the information we learned from all the tasks is an interesting multi-task problem which we would like to address in this thesis.

Description	Coding
Year	1985=1; 1986=2; 1987=3
% FSM	Percent of students eligible for free school meals
% VR1 band	Percent of students in school in VR band 1
Gender	Male=0; Female=1
VR band of student	VR1=2; VR2=3; VR3=1
Ethnic group of student	ESWI=1 (Students born in England, Scotland, Wales or Ireland); African=2; Arab=3; Bangladeshi=4; Caribbean=5; Greek=6; Indian=7; Pakistani=8; S.E.Asian=9; Turkish=10; Other=11
School gender	Mixed=1; Male=2; Female=3
School denomination	Maintained=1; Church of England=2; Roman Catholic=3

Table 1.1: School Features

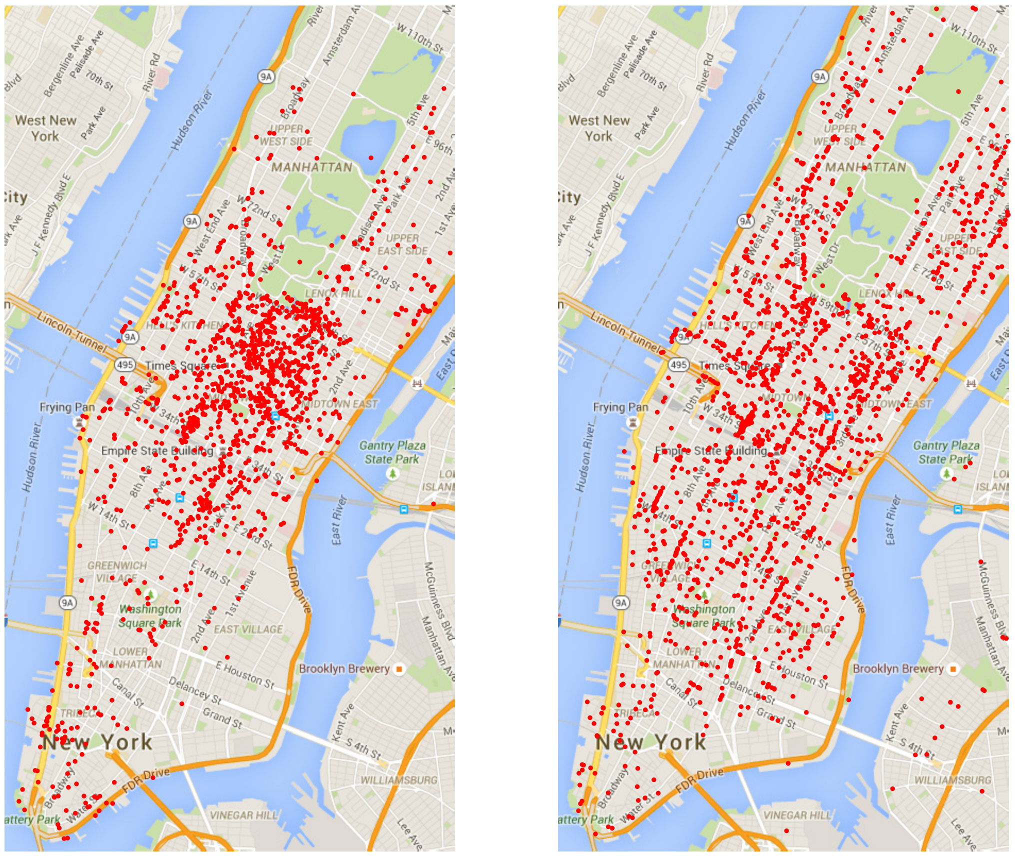


Figure 1.4: Distributions of drop-off locations for taxi data with the same pick-up location around Time Square, New York city: 8-9am (left) and 9-10pm (right)

Another real-world example is predicting the distributions of drop-off locations for taxis. We have the NYC dataset[2] which consists of pickup time, pickup location and dropoff location for taxis in New York city during the year of 2013. Each location is represented by a real-valued pair: (longitude, latitude). In Figure 1.4, we plot the drop-off locations (shown by red dots) for taxies for two different time periods: 8-9am in the morning (left), and 9-10pm in the evening (right), given the start location within a certain radius of $(-73.985, 40.758)$, which is near Time Square, New York city. We can see that the distributions for these two time periods are quite different: in the morning the drop-off locations are mainly concentrated around the business districts for work, while in the evening the drop-off locations are more scattered as people take taxies to restaurants or residential areas. Here lies an interesting transfer problem: given the data from previous time frames/certain geographical regions, how can we transfer the knowledge to predict the drop-off locations for the future/for a different region? In this thesis, we will propose methods to solve this problem of distribution transfer.

1.3 Notations and Acronyms

In Table 1.2 we show a list of common symbols used in this thesis. In Table 1.3 we show a list of acronyms used in this thesis.

1.4 Thesis Overview

1.4.1 Active Transfer Learning under Model Shift

In the first part of this thesis, we propose two algorithms for transfer learning under **model shift**. Model shift means we allow $P(Y|X)$ to change across domains. We analyze the risk bounds for both algorithms and show that we achieve favorable convergence rate compared to no-transfer learning. In addition, we propose a flexible transfer learning algorithm that allow both support and model to change across domains. Specifically,

- In chapter 3, we propose two transfer algorithms that allow both $P(X)$ and $P(Y|X)$ to change across the source and target tasks. We assume only that the change is smooth as a function of X. The first approach builds on the kernel mean matching (KMM) idea [25, 33] to match the conditional distributions, $P(Y|X)$, between the tasks. The second approach uses Gaussian Processes to model the source task, the target task, and the offset between. The assumption here is that although the offset may be a nonlinear function over the input domain, there is some smoothness in that offset over the input domain. If that is not true, we suspect there is little hope for transferring between domains at all. The GP-based approach naturally lends itself to the active learning setting where we can sequentially choose query points from the target dataset (Figure 1.5). Its final predictive covariance, which combines the uncertainty in the transfer function and the uncertainty in the target label prediction, can be plugged into various GP based active query selection criteria. The content of this chapter has been published in [73].

[2]http://www.andresmh.com/nyctaxitrips/

Table 1.2: Notations

Symbol	Definition
X	feature matrix
Y	label vector
X^s, Y^s	data from the source domain
X^t, Y^t	data from the target domain
X^{tL}, Y^{tL}	labeled data in the target domain
X^{tU}, Y^{tU}	unlabeled data in the target domain
m	source sample size
n_l	number of labeled target samples
$\mathbf{w}$	scale vector
$\mathbf{b}$	offset vector
$\odot$	elementwise product
$\mathcal{U}$	conditional embedding operator
$\mu[P(X)]$	kernel mean embedding on distribution $P(X)$
$\phi(x)$	feature map on x
$\psi(y)$	feature map on y
$K_{XX'}$	kernel matrix between feature matrix X and X'
λ	regularization parameters
h	hypothesis
l	loss function
Λ	reweighting matrix representing task relationships
κ	upper bound on the kernel values $k(x, x')$

Table 1.3: Acronyms

Acronym	Definition
AQI	Air Quality Index
KMM	Kernel Mean Matching [26, 33]
SMS	Support and Model Shift
TL	Transfer Learning
AL	Active Learning
AS	Active Surveying
GP	Gaussian Process
KRR	Kernel Ridge Regression
CDM	Conditional Distribution Matching
T/C Shift	Target/Conditional Shift
MT-KRR	Multi-Task Kernel Ridge Regression
MMD	Maximum Mean Discrepancy [26, 33]
CDT	Conditional Distribution Transfer

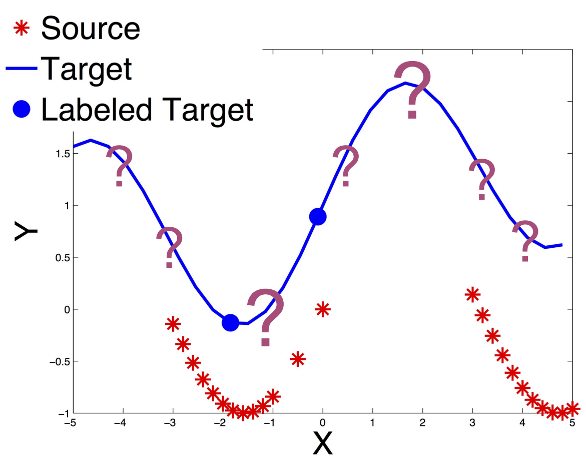


Figure 1.5: Active Learning in the Transfer Learning Framework

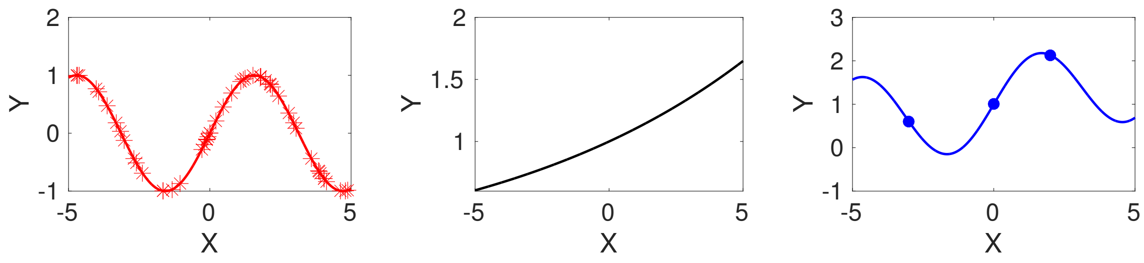


Figure 1.6: Larger λ_* (smoother change)

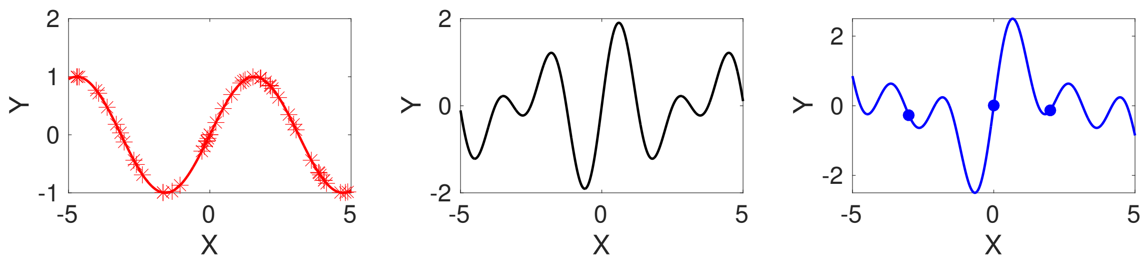


Figure 1.7: Smaller λ_* (less smooth change)

- In chapter 4, we develop theoretical analysis for transfer learning algorithms under the model shift assumption. Our analysis sheds light on conditions when transfer learning works better than no-transfer learning. We show that under certain smoothness assumptions it is possible to obtain a favorable convergence rate with transfer learning compared to no transfer at all. Specifically, our main result is that even when the conditional distributions are allowed to change across domains, we are still able to obtain a generalization bound of $O(\frac{1}{\lambda_* \sqrt{n_l}})$ with respect to the labeled target sample size n_l, modified by the smoothness of the transformation parameters (λ_*, examples are shown in Fig. 1.6 and

Fig. 1.7) across domains. In Fig. 1.6 we have a larger λ_* which corresponds to a smoother change (the offset between the target and the source is a very smooth function w.r.t. X), then we obtain favorable bounds for transfer learning than no-transfer at all. In Fig. 1.7 we have a smaller λ_* which corresponds to a less smooth change (the offset function changes rapidly w.r.t. X), then the bound of transfer learning is similar to the bound of learning by labeled-target samples only, which means we do not benefit much from transfer learning. Furthermore, using the generalization bounds we derived, we are able to extend the transfer learning algorithm from a single source to multiple sources, where each source is assigned a weight that indicates how helpful it is for transferring to the target. The content of this chapter has been published in [71].

- In chapter 5, we propose a transfer learning algorithm that allows both the support on X and Y, and the model $P(Y|X)$ to change across the source and target domains. We assume only that the change is smooth as a function of X. In this way, more flexible transformations are allowed than mean-centering and variance-scaling. Specifically, we build a Gaussian Process to model the prediction on the transformed X, then the prediction is matched with a few observed labels Y (also properly transformed) available in the target domain such that both transformations on X and on Y can be learned. As an illustration, we show a toy problem in Fig. 1.8. As we can see, the support of $P(X)$ in the training domain (red stars) and the support of $P(X)$ in the test domain (blue line) do not overlap, neither do the support of Y across the two domains. The goal is to learn a model on the training data, with a few labeled test data (the filled blue circles), such that we can successfully recover the target function (the blue line). In Fig. 1.9, we show the labels (the yield) of each grape image dataset, along with the 3rd dimension of its feature space. We can see that the real-world problem is quite similar to the toy problem, which indicates that the algorithm we propose in this work will be both useful and practical for real applications. The content of this chapter has also been published in [70].

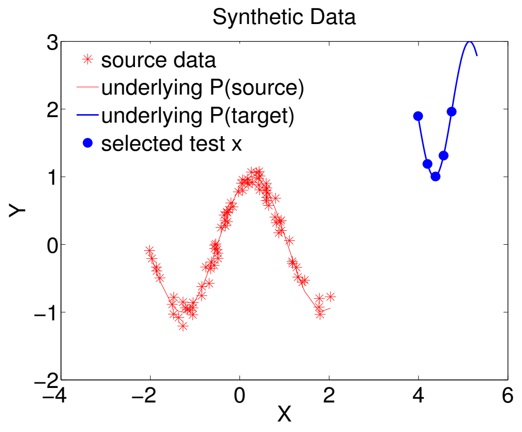


Figure 1.8: Toy problem

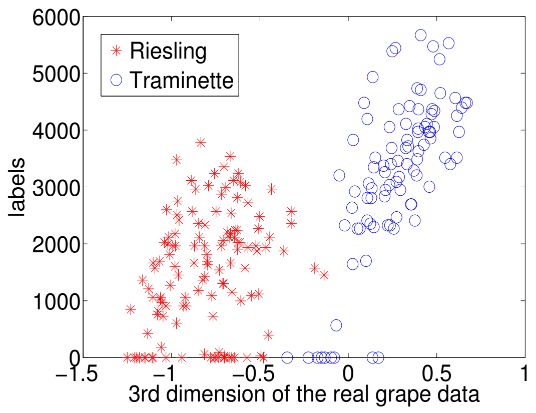


Figure 1.9: Real grape data

1.4.2 Stability Analysis for Multi-Task Learning Problems

In the second part of the thesis, we focus on multi-task learning problems. Similar to transfer learning where one uses data (or an estimator) from a well understood source domain with plentiful data to aid the learning of a target domain with scarce data, multi-task learning pools multiple domains together and couples the learning of several tasks by regularizing separate estimators jointly and dependently (Figure 1.10). Although multi-task learning algorithms are becoming prevalent in machine learning, there are gaps in our understanding of their properties, especially in nonparametric settings. This part of the thesis looks to increase our understanding of fundamental questions such as: What can one say about the true risk of a multi-task estimator given its empirical risk? How do relative sample sizes affect learning among different domains?

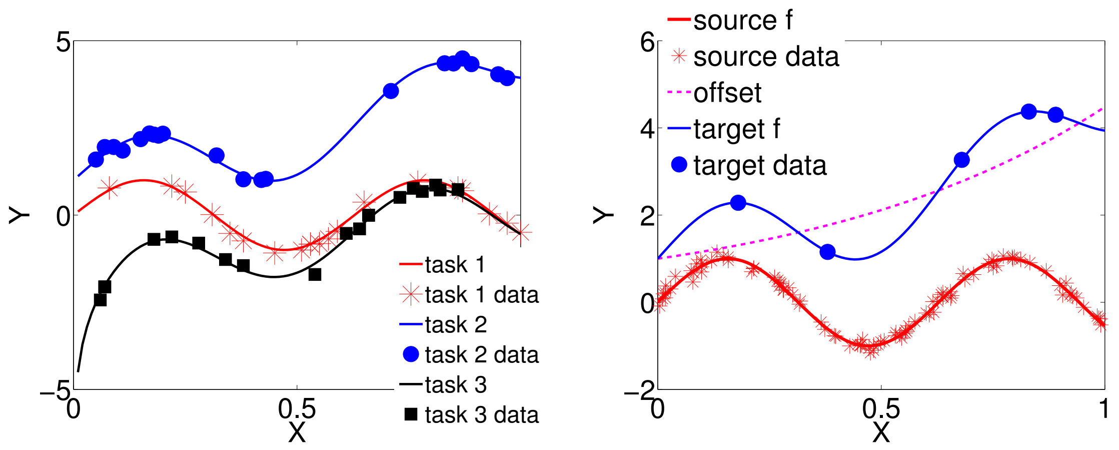


Figure 1.10: Toy example illustrating multi-task learning (left) and transfer learning (right).

Although there are many regularization techniques proposed for multi-task learning, there are very few studies on how the learning bounds change under different parameter regularizations.

- In chapter 6, we analyze the stability bounds under a general framework of multi-task learning using kernel ridge regression. Our formulation places a reweighting matrix Λ on task weights to capture the relationship among tasks. Our analysis shows that most existing work can be cast into this framework by changing the reweighting matrix Λ. More importantly, we show that the stability bounds under this framework depend on the diagonal blocks of Λ^{-1}, thus providing insights on how much we can gain from regularizing the relationship among tasks by using different reweighting matrices. The content of this chapter has been published in [74].

1.4.3 Estimating Distributions in the Transfer Learning Setting

All the above work deals with predictions on the labels Y with real-valued outputs (regression) or categorical outputs (classification). In many real-world applications, a distributional output $P(Y)$ is desired. In addition, in many scenarios the output distribution is multi-modal, and a fixed real-valued prediction (e.g. the mean or some variation of the mean) might be inaccurate. As a more concrete example, on a traffic dataset that consists of pickup/dropoff locations for taxis, given a certain pickup location, instead of predicting a fixed dropoff location, it is more beneficial to predict the distribution of dropoff locations such that a taxi driver can optimize his/her future actions.

- In chapter 7, we propose methods for estimating distributions in a transfer learning setting. We first propose an approach for **distribution transfer** on $P(Y)$ in a nonparametric setting, i.e., estimating $P(Y)$ in the target domain by transferring $P(Y)$ from the source domain. Then we propose an approach for **conditional distribution transfer**, i.e., estimating $P(Y|X = x^*)$ in the target domain by transferring from the source distribution. Finally we extend the algorithm to **multi-source distribution transfer**, where we estimate $P(Y)$ or $P(Y|X = x^*)$ in the target domain by combining multiple source distributions.

1.4.4 Experiments on Real-world Applications

We evaluate our methods on both synthetic data and real-world data. The real-world datasets include:

- Yield estimation based on images across grape varieties
- Air-quality prediction from Weibo posts for different cities
- Predict whether a robot successfully climbs over an obstacle
- Predict examination scores for students to study the effectiveness of schools
- Estimate the distribution of drop-off locations for taxis on the traffic data

In all applications, we show that our proposed methods lead to improved prediction performance compared to state-of-the-art methods.

Chapter 2

Related Work

2.1 Transfer Learning

Traditional methods for transfer learning usually assume that specific parts of the model can be carried over between tasks. [46] uses Markov logic networks to transfer relational knowledge across domains. [47] learns Bayes Net structures by biasing learning toward similar structures for each task. [18, 55] achieves transfer learning by assuming that models for related tasks share some parameters or prior distributions of hyperparameters.

Recently, a large part of transfer learning work has focused on the problem of covariate shift [25, 33, 64]. They consider the case where the marginal distributions on X are different across domains, i.e., $P(X^s)$ differs from $P(X^t)$, while making the assumption that the conditional distributions $P(Y^s|X^s)$ and $P(Y^t|X^t)$ are the same. The kernel mean matching (KMM) method [25, 33], is one of the algorithms that deal with covariate shift. It minimizes $||\mu(P_t) - \mathbf{E}_{x\sim P_s(x)}[\beta(x)\phi(x)]||$ (here μ is the kernel mean embedding of distributions, and $\phi(x)$ is the feature map on x) over a re-weighting vector β on training data points such that distributions on X are matched with each other. They then incorporate the learned weights $\hat{\beta}$ into the training procedure, e.g., training an SVM with re-weighted source data points, to obtain a model that generalizes well on the target data. The advantage of using kernel mean matching is to avoid density estimation, which is difficult in high dimensions. It has been proved [67] that even if we use the empirical version of mean embeddings we can still achieve a fast convergence rate of $O(m^{-1/2})$, where m is the sample size. However, this work suffers two major problems. First, the conditional distribution $P(Y|X)$ is assumed to be the same, which might not be true under many real-world cases. The algorithm we propose will allow more than just the marginal on X to shift. Second, the KMM method requires that the support of $P(X^t)$ is contained in the support of $P(X^s)$, i.e., the training set is richer than the test set. This is not necessarily true in many real cases either. Consider the task of transferring yield prediction using images taken from different vineyards. If the images are taken from different grape varieties or during different times of the year, the texture/color could be very different across transferring tasks. In these cases one might mean-center (and possibly also variance-scale) the data to ensure that the support of $P(X^t)$ is contained in (or at least largely overlapped with) $P(X^s)$. We provide an alternative way to solve the support shift problem that allows more flexible transformations than mean-centering and variance-scaling.

Some more recent research [80] has focused on modeling target shift ($P(Y)$ changes), conditional shift ($P(X|Y)$ changes), and a combination of both. The assumption on target shift is that X depends causally on Y, thus $P(Y)$ can be re-weighted to match the distributions on X across domains. In conditional shift, the authors apply a parameterized-location-scale transformation on $P(X|Y)$ to match $P(X)$. However, the authors still assume that the support of $P(Y^t)$ is contained in the support of $P(Y^s)$, which is not necessarily true in many cases. Further for transfer learning under model shift, there could be a difference in $P(Y|X)$ that can not simply be captured by the differences in $P(X)$, hence neither covariate shift nor target/conditional shift will work well under the model shift assumption.

There also have been a few papers dealing with differences in $P(Y|X)$. Jiang and Zhai. [36] designed specific methods (change of representation, adaptation through prior, and instance pruning) to solve the label adaptation problem. Liao et al. [39] relaxed the requirement that the training and testing examples be drawn from the same source distribution in the context of logistic regression. They also proposed an active learning approach using the Fisher information matrix, which is a lower bound of the exact covariance matrix. Sun et al. [68] weighted the samples from the source domain to handle the domain adaptation. These settings are relatively restricted while we consider a more general case that there is a smooth transformation from the source domain to the target domain, hence all source data will be used with the advantage that the part of source data which do not help prediction in the target domain will automatically be corrected via the transformation model.

2.1.1 Theory on Transfer Learning

A number of theoretical analyses on transfer learning have also been developed. [58] presented VC-dimension-based generalization bounds for transferring classification tasks. Later [8] extended the work with a bound on the error rate under a weighted combination of the source data. [43] introduced a discrepancy distance suitable for arbitrary loss functions and derived new generalization bounds for domain adaptation for a wide family of loss functions. However, most of the work mentioned above deals with transfer learning under the covariate shift assumption, i.e., they still assume the conditional distribution stays the same across domains, or the labeling functions in the two domains share strong proximity in order for adaptation to be possible. For example, one of the bounds derived in [43] has a term $\mathcal{L}(h_Q^*, h_P^*)$ (here h_Q^*, h_P^* are the hypotheses that minimize the objective in the source domain and target domain, respectively) related to the average loss between the minimizer h_Q^* in the source domain and the minimizer h_P^* in the target domain, which could be fairly large when there exists a constant offset between the two labeling functions.

Similarly, most work in transfer learning with multiple sources focuses only on $P(X)$. For example, [42] proposed a distribution weighted combining rule of source hypotheses using the distribution $P(X)$ for both source and target. This approach requires estimating the distribution $P_i(x)$ of source i on a target point x from large amounts of unlabeled points from the source, which might be difficult in real applications with high-dimensional features. Other existing work focuses on finding the set of sources that are closely related to the target [16], or a reweighting of sources based on prediction errors [76]. [12] proposed a conditional probability based weighting scheme under a joint optimization framework, which leads to a reweighting of sources that prefers more consistent predictions on the target. However, these existing approaches do not consider the problem that there might exist shifts in the conditional distribution from source to the target, and how the smoothness of this shift can help learn the target.

2.2 Active Learning

A research area we draw from is active learning with Gaussian Processes, for which many selection criteria have been proposed, such as choosing the test point with the highest variance (or entropy). We can also utilize mutual information [28], which the same authors further improve by considering both parameter (kernel width) uncertainty reduction (exploration) and model uncertainty reduction under current parameter setting (exploitation) [37]. Another popular criterion is minimizing the total variance conditioned on the point to be selected [35, 61], which can be done using the trace of the covariance matrix, $\mathrm{Tr}\{\sigma^2_{y|A}\}$, where A is the set of labeled data points and the candidate query points. Active surveying [22, 41, 72], uses an estimation objective that is the sum of all the labels in the test set. The corresponding myopic active selection criteria is minimizing the sum of all elements in the covariance matrix conditioned on the selected point, $\mathbf{1}^\top \sigma^2_{y|A}\mathbf{1}$. We adopt these last two selection criteria for our active transfer algorithms.

2.3 Combining Transfer and Active Learning

The idea of combining transfer learning and active learning has also been studied recently. Both [62] and [54] perform transfer and active learning in multiple stages. The first work uses the source data without any domain adaptation. The second work performs domain adaptation at the beginning without further refinement. [59] and [13] consider active learning under covariate shift and still assume $P(Y|X)$ stays the same. In Chapter 3, we propose a combined active transfer learning algorithm to handle the general case where $P(Y|X)$ changes smoothly across domains. However, covariate shift algorithms are still needed to solve the problem that $P(X)$ might differ across domains, which follows the assumption covariate shift made on the support of $P(X)$. In Chapter 5, we further propose an algorithm that allows more flexible transformations (parameterized-location-scale[1] transform on both X and Y). Our experiments on real-data shows this additional flexibility pays off in real applications.

2.4 Multi-task Learning

A closely related area to transfer learning is multi-task learning, where people try to learn multiple related tasks jointly. A large part of work on multi-task learning focuses on learning a common feature representation shared by tasks [1, 2, 3, 14, 79], and directly inferring the relatedness of tasks [6, 34, 75, 78]. Also people have achieved multi-task learning by transferring

[1]we use "parameterized" to indicate that the transform is a function w.r.t to X, in order to distinguish from the usual term "location-scale" family which people use for probability distributions.

knowledge of parameters. Both [10, 38] handle multiple-task learning in the context of Gaussian Process by using some GP prior to induce correlations between tasks. [19, 60] use a hierarchical Bayesian framework for multi-task learning in the context of GP and SVM, respectively.

A large part of multi-task learning work is formulated using kernel ridge regression (KRR) with various regularizations on task relations. The ℓ_2 penalty is used on a shared mean function and on the variations specific to each task [19, 20]. In [66] a pairwise ℓ_2 penalty is placed between each pair of tasks. In [84] an ℓ_2 penalty is proposed on each pair of consecutive tasks that controls the temporal smoothness. By regularizing the shared clustering structure among task parameters, task clusters are constructed for different features [83]. A multi-linear multi-task learning algorithm is proposed in [56] by placing a trace norm penalty. However, there are very few literature dealing with the stability of multi-task KRR algorithms. The stability bounds for transductive regression algorithms are analyzed in [15]. In [5], the authors study the stability properties of multi-task KRR by considering each single task separately, thus failing to reveal the advantage of regularizing task relations. In [44], a regularized trace-norm multi-task learning algorithm is studied where the task relations are modeled implicitly, while we study a general family of multi-task learning algorithms where the task relations are modeled explicitly, reflected by a reweighting matrix Λ on task weights. More recently, the algorithmic stability for multi-task algorithms with linear models $f(x) = w^\top x$ (w is the weight vector on x) is studied in [82]. While in this thesis we consider the more challenging **nonlinear** models with feature map $\phi(x)$, where the new multi-task kernel between x_i, x_j is defined as $\phi(x_i)\Lambda^{-1}\phi^\top(x_j)$ (Λ is a reweighting matrix representing the relationship among all the tasks) by absorbing Λ. Our theory is also developed on the more general nonlinear models.

2.5 Distribution Learning

Techniques for distribution regression are also becoming increasingly popular in machine learning. For example, Poczos et al. [52] focuses on distribution to real-value regression, and Oliva et al. [49] studies distribution to distribution regression in a nonparametric setting. Both authors assume that the learner is given multiple sets of input/output samples that one can estimate the distribution from, and the goal is to learn the correspondence from the input distribution to the output distribution. Unlike these approaches, we focus on the transfer learning setting, where we are only given one set of source/target samples, and we recover the target distribution based on the source distribution and a few target labels, assuming that the source and the target are somewhat similar.

One area close to distribution transfer is distribution estimation with priors. Most of these methods focus on the parametric setting, for example, Bayesian posterior estimation. Hjort and Glad [30] develops semiparametric methods for density estimation by using a parametric start. In this thesis, we focus on the nonparametric setting, i.e., we would like to transfer between general distributions, rather than a certain family of distributions that can be represented by some parameters. On the other hand, there has been some work in the area of conditional density estimation [24, 32, 57]. In this thesis, when estimating the conditional density we adopt a similar technique except that the source samples are reweighted according to the distribution transfer objective.

Specifically, we focus on transfer learning on the distribution $P(Y)$ and conditional distribution $P(Y|X = x^*)$ while allowing both distributions to change across domains. This is very different from most existing work on covariate shift [27, 33, 64], which assumes that only the marginal distribution $P(X)$ changes across domains, while the conditional distribution $P(Y|X)$ remains the same.

2.6 Deep Learning Meets Transfer Learning

Recently there also have been a few papers dealing with transfer learning in deep neural networks. The basic idea is to have the lower-layer features fixed while re-training the higher-layer features specific for each task (Figure 2.1). In [77], the authors have shown that deep features transition from general to specific along the deep network, thus the transferability drops for a different task. In the same paper, the authors also show that, as the source and target tasks become more dissimilar, the performance benefits of transferring features decreases. Hence it is necessary to reduce the dataset bias and enhance the transferability across tasks for the higher layers. [40] proposes a Deep Adaptation Network, which adds task-specific layers to match the distribution embeddings of the hidden representations for different domains. [21] proposes a deep architecture that promotes deep features that are discriminative for the source task and also invariant with respect to the shift between the domains. The domain-invariant features are achieved by adding a domain classifier with a gradient reversal layer that ensures the feature distributions over the two domains are made similar, i.e., as indistinguishable as possible for the domain classifier.

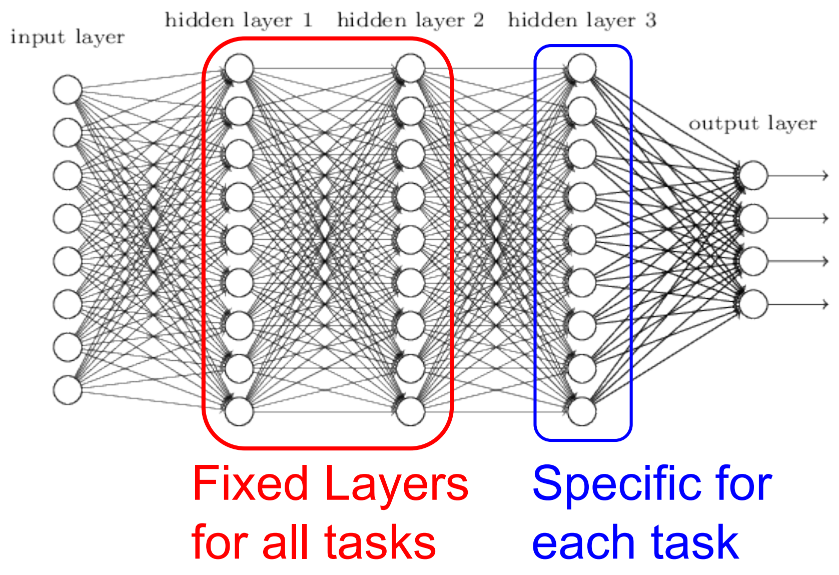


Figure 2.1: Transfer Learning in deep networks: deep features transition from generic to specific

Chapter 3

Active Transfer Learning under Model Shift

3.1 Overview

In this chapter, we propose two transfer learning algorithms under model shift. Here model shift means we allow conditional distributions $P(Y|X)$ to change across domains. We only assume that the change is a smooth function with respect to the features X. The two algorithms are:

- Conditional Distribution Matching:
  we match the conditional distributions between the tasks by using the idea of kernel embedding of conditional distributions
- Modeling the offset/difference between source and target functions by Gaussian Processes: we build Gaussian Processes to model the source task, the target task, and the offset between. This approach also lends itself to an active learning algorithm where we can sequentially select data points for query.

3.2 Problem Formulation

Assume we are given a set of m labeled training data points, (X^s, Y^s), from the source domain where each $X_i^s \in \Re^{d_x}$ and each $Y_i^s \in \Re^{d_y}$. Assume we are also given a set of n test data points, X^t, from the target domain. Some of these will have corresponding labels, Y^{tL}. When necessary we will separately denote the subset of X^t that has labels as X^{tL}, and the subset that does not as X^{tU}.

For *static transfer learning*, the goal is to learn a predictive model using all the given data that minimizes the squared prediction error on the test data, $\Sigma_{i=1}^n(\hat{Y}_i^t - Y_i^t)^2$ where $\hat{Y}_i$ and Y_i are the predicted and true labels for the ith test data point. We will evaluate the transfer learning algorithms by including a subset of labeled test data chosen uniformly at random.

For *active transfer learning* the performance metric is the same. The difference is that the active learning algorithm chooses the test points for labeling rather than being given a randomly chosen set.

The surveying metric is to minimize the error on the sum of the predictions: $(\Sigma_{i=1}^{n}\hat{Y}_i^t - \Sigma_{i=1}^{n}Y_i^t)^2$. Again, this is evaluated using a randomly chosen set of test labels for static transfer surveying or a set chosen by the algorithm for active transfer surveying.

To illustrate the problem, we show a toy example in Figure 3.1. The left figure shows data in the source domain, drawn from a sine function. The right figure shows data in the target domain, drawn from the same sin function adding a positive offset function which is exponentially increasing (the middle figure of Fig. 3.1). The goal is, given the data from the source function (shown as the red stars in Fig. 3.1, left), and a few data points to query (blue dots in Fig. 3.1, right), to recover the target function (the blue line in Fig. 3.1) in the least number of queries.

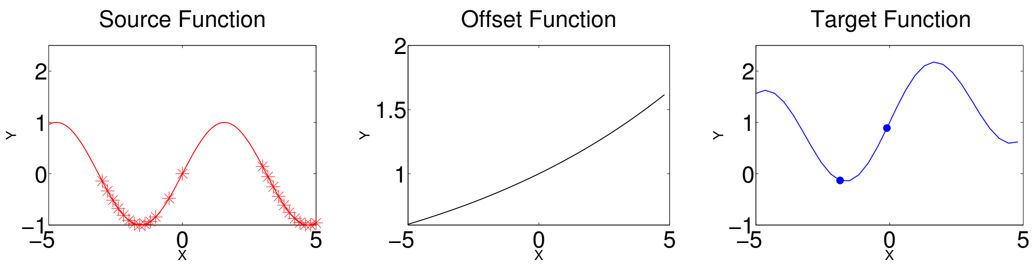


Figure 3.1: Toy example showing the transfer/active learning problem

3.3 Transfer Learning

3.3.1 Conditional Distribution Matching Approach (CDM)

First we propose a distribution matching approach for transfer learning. The basic idea is, we want to

- **Match the conditional distributions**

$$P(Y^{new}|X^s) \text{ and } P(Y^t|X^t),$$

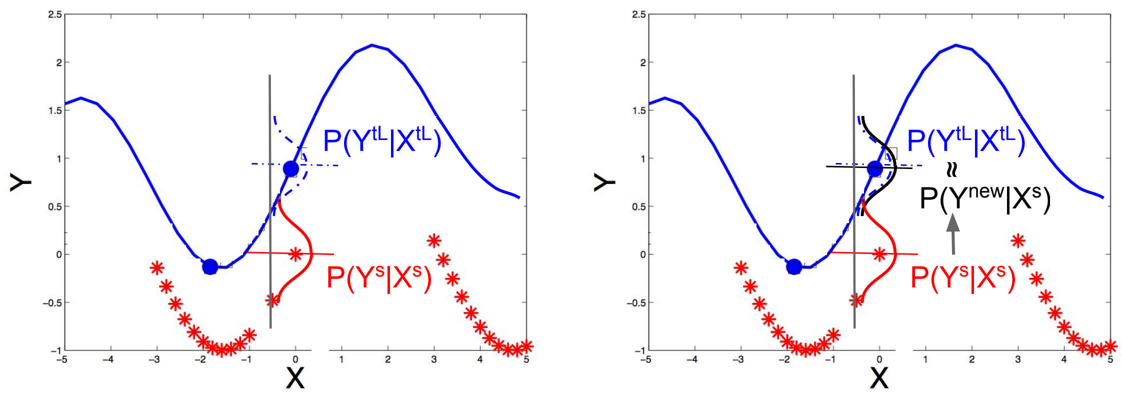


Figure 3.2: Conditional Distribution Matching by transforming the source labels Y^s to Y^{new}

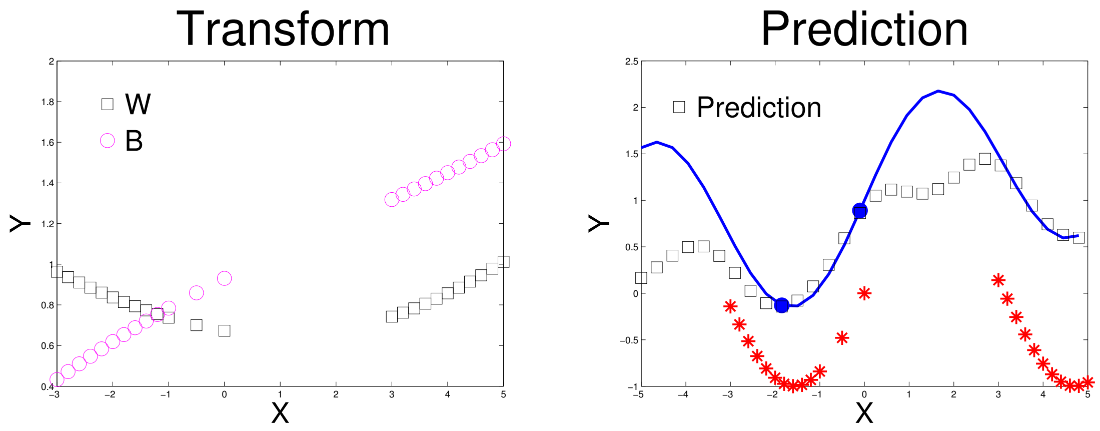


Figure 3.3: Illustration of the smooth transformations $\mathbf{w}, \mathbf{b}$ w.r.t. X (left); Prediction results of CDM after transforming the source labels

as shown by Figure 3.2, where Y^{new} is under a parameterized-location-scale transform of Y^s:

$$Y^{new} = Y^s \odot \mathbf{w}(X^s) + \mathbf{b}(X^s).$$

Here $\mathbf{w}$ is a scale vector on X^s and $\mathbf{b}$ is an offset vector on X^s. The $\odot$ is the elementwise product, which means we allow nonlinear transformations on Y^s. If the conditional distributions are matched with each other, and $P(X^s) = P(X^t)$ (which can be achieved by various methods dealing with covariate shift, hence it is not the focus of this work), then a model learned from the source data will generalize well on the target data because the joint distribution is also matched with each other, i.e., $P(X^s, Y^s) = P(X^t, Y^t)$.

- **The transform function is smooth**, i.e., $\mathbf{w}$ and $\mathbf{b}$ should be smooth w.r.t X (Figure 3.3, left). Figure 3.3 (right) shows an example of the prediction result by using CDM.

To achieve the first goal, similar to the kernel mean matching idea, we can directly minimize the discrepancy (Hilbert-Schmidt norm) between the conditional embedding operators [67] of the two distributions with a regularization term:

$$\min_{\mathbf{w},\mathbf{b}} L + L_{reg}, \text{where } L = ||\hat{\mathcal{U}}[P_{Y^{new}|X^s}] - \hat{\mathcal{U}}[P_{Y^t|X^t}]||^2_{HS}, L_{reg} = \lambda_{reg}(||\mathbf{w} - \mathbf{1}||^2 + ||\mathbf{b}||^2). \tag{3.1}$$

Here $\mathcal{U}$ is the conditional embedding operator [67], where its empirical estimate $\hat{\mathcal{U}}$ is given by:

$$\hat{\mathcal{U}}[P_{Y|X}] = \Psi(Y)(K_{XX} + \lambda I)^{-1}\Phi^\top(X).$$

To obtain the derivative of L w.r.t. $\mathbf{w}, \mathbf{b}$, we simplify L by:

$$\begin{aligned}L &= ||\psi(Y^{new})(K_{X^sX^s} + \lambda I)^{-1}\phi^\top(X^s) - \psi(Y^t)(K_{X^tX^t} + \lambda I)^{-1}\phi^\top(X^t)||^2_F \\ &= C + \mathrm{Tr}\{\phi(X^s)(L^s + \lambda I)^{-1}\tilde{K}(L^s + \lambda I)^{-1}\phi^\top(X^s)\} \\ &\quad - 2\,\mathrm{Tr}\{\phi(X^s)(L^s + \lambda I)^{-1}\tilde{K}^c(L^t + \lambda I)^{-1}\phi^\top(X^t)\} \\ &= C + \mathrm{Tr}\{(L^s + \lambda I)^{-1}\tilde{K}(L^s + \lambda I)^{-1}L^s\} - 2\,\mathrm{Tr}\{(L^s + \lambda I)^{-1}\tilde{K}^c(L^t + \lambda I)^{-1}K_{X^tX^s}\},\end{aligned}$$

where C is some constant, $\tilde{K} = K_{Y^{new}Y^{new}}, \tilde{K}^c = K_{Y^{new}Y^t}, L^s = K_{X^sX^s}, L^t = K_{X^tX^t}$. $K_{XX'}$ is the Gram matrix defined as $K_{XX'}(i,j) = k(x_i, x'_j)$ with kernel k. λ is the regularization parameter to ensure the kernel matrix is invertible.

Hence the derivative of L w.r.t $\mathbf{w}, \mathbf{b}$ is given by:

$$\begin{aligned}
\frac{\partial L}{\partial \tilde{K}} &= (K_{XX} + \lambda I)^{-1} K_{XX}^{\top} (K_{XX} + \lambda I)^{-1},\\
\frac{\partial L}{\partial \tilde{K}_c} &= 2(K_{XX} + \lambda I)^{-1} K_{X^{tL}X}^{\top} (K_{XX}^{tL} + \lambda I)^{-1},\\
\frac{\partial L}{\partial \mathbf{w}_p} &= \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}})^{\top}(\frac{\partial \tilde{K}}{\partial \mathbf{w}_p})] - \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}_c})^{\top}(\frac{\partial \tilde{K}_c}{\partial \mathbf{w}_p})]\\
&= \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}})^{\top}(\tilde{K} \odot D_p)] - \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}_c})^{\top}(\tilde{K}_c \odot E_p)],
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{b}_p} &= \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}})^{\top}(\frac{\partial \tilde{K}}{\partial \mathbf{b}_p})] - \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}_c})^{\top}(\frac{\partial \tilde{K}_c}{\partial \mathbf{b}_p})]\\
&= \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}})^{\top}(\tilde{K} \odot \tilde{D}_p)] - \mathrm{Tr}[(\frac{\partial L}{\partial \tilde{K}_c})^{\top}(\tilde{K}_c \odot \tilde{E}_p)],
\end{aligned}$$

where

$$\begin{aligned}
[D_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{new})(Y_i^s I(i=p) - Y_j^s I(j=p)),\\
[E_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{tL}) Y_i^s I(i=p),\\
[\tilde{D}_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{new})(I(i=p) - I(j=p)),\\
[\tilde{E}_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{tL}) I(i=p).
\end{aligned}$$

To make the transformation smooth w.r.t. X, we parameterized $\mathbf{w}, \mathbf{b}$ in this way [80]: $\mathbf{w} = R\mathbf{g}, \mathbf{b} = R\mathbf{h}$, where $R = L^s(L^s + \lambda I)^{-1}$. Then the new parametrization results in:

$$\begin{aligned}
[D_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{new})(Y_i^s R_{ip} - Y_j^s R_{jp}),\\
[E_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{tL}) Y_i^s R_{ip},\\
[\tilde{D}_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{new})(R_{ip} - R_{jp}),\\
[\tilde{E}_p]_{ij} &= -\frac{1}{\sigma^2}(Y_i^{new} - Y_j^{tL}) R_{ip}.
\end{aligned}$$

Similarly, the regularization term is:

$$\lambda_{reg}(||R\mathbf{g} - \mathbf{1}||^2 + ||R\mathbf{h}||^2).$$

Then we apply gradient descent to minimize the objective function.

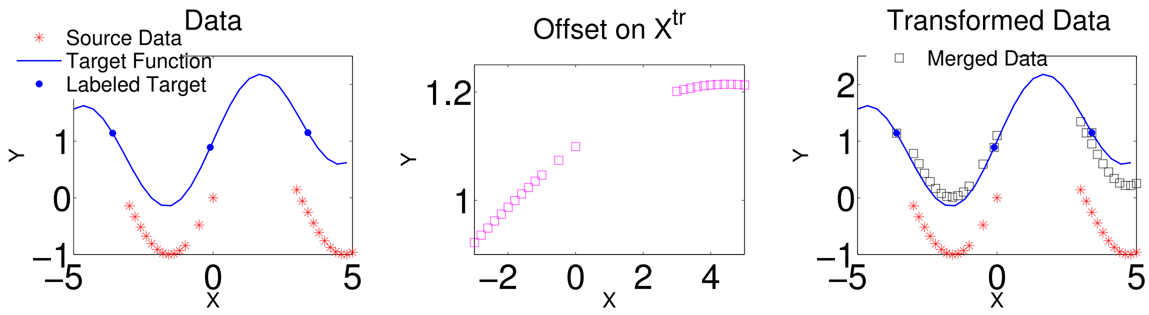


Figure 3.4: Illustration of the Offset Approach

When matching the conditional distributions, if we only use X^{tL}, Y^{tL} in the empirical version of the conditional operator $\hat{\mathcal{U}}[P_{Y^t|X^t}]$, it will be unstable due to the small size of the observed labeled test points, especially in the early stage of active learning. However, using both X^{tL}, Y^{tL} and X^{tU}, Y^{tU} would require knowing the values Y^{tU}, which are not obtained before querying. We replace Y^{tU} with the prediction $\hat{Y}^{tU}$ based on $\{X^s, Y^{new}\} \cup \{X^{tL}, Y^{tL}\}$, where Y^{new} are under transformation using current $\mathbf{w}, \mathbf{b}$, while $\{X^{tL}, Y^{tL}\}$ are the labeled test data selected up to the present. After obtaining $\hat{Y}^{tU}$ we minimize the objective function Eq 3.1. We iterate over the two steps until convergence.

3.3.2 Offset Approach

In the second proposed method, we use Gaussian Processes to model the source task, the target task, and the offset between, described as follows (K in the following equations stands for the Gaussian kernel, and λ is the regularization parameter to ensure the kernel matrix is invertible):

- We build a GP from the source domain and predict on X^{tL}, then compute the offset Z between the prediction and the true labels Y^{tL}:

$$\hat{Z}^{tL} = Y^{tL} - \hat{Y}^{tL}.$$

  Using the property of GP it follows:

$$P(\hat{Z}^{tL}|X^s, Y^s, X^{tL}, Y^{tL}) \sim \mathcal{N}(\mu_s, \Sigma_s),$$

  where

$$\mu_s = Y^{tL} - K_{X^{tL}X^s}(K_{X^sX^s} + \lambda I)^{-1}Y^s,$$
$$\Sigma_s = K_{X^{tL}X^{tL}} - K_{X^{tL}X^s}(K_{X^sX^s} + \lambda I)^{-1}K_{X^sX^{tL}}.$$

- We transform Y^s to Y^{new} by

$$Y^{new} = Y^s + \hat{Z}^s,$$

  where $\hat{Z}^s$ is the predicted mean of the offset on X^s using the GP built from $\{X^{tL}, \hat{Z}^{tL}\}$ (Figure 3.4, middle), i.e.,

$$P(\hat{Z}^s|\hat{Z}^{tL}, X^s, X^{tL}) \sim \mathcal{N}(\mu_0, \Sigma_0),$$

where

$$\mu_0 = K_{X^s X^{tL}}(K_{X^{tL}X^{tL}} + \lambda I)^{-1}\hat{Z}^{tL},$$
$$\Sigma_0 = K_{X^s X^s} - K_{X^s X^{tL}}(K_{X^{tL}X^{tL}} + \lambda I)^{-1}K_{X^{tL}X^s}.$$

- Train a model on the merged data $\{X^s, Y^{new}\} \cup \{X^{tL}, Y^{tL}\}$ (Figure 3.4, right), and we use the model to make predictions on X^{tU}.

Computational Complexity. In the offset approach, building a GP on the source data with m samples has a complexity of $O(m^3)$ because of the kernel matrix inverse. The complexity can be reduced to $O(d^3 + d^2 m)$ by using random Fourier features [53] with d-dimension features.

3.4 Active Learning

We consider two active learning goals and apply a myopic selection criterion to each:

- **Active learning** which tries to reduce the total predictive variance [35, 61]. The goal is to minimize the risk (here $\hat{y}$ denotes the estimation on y):

$$\mathbb{E}\sum_{i \in tU}(y_i - \hat{y}_i)^2 = \mathbb{E}[\mathbb{E}[\sum_{i \in tU}(y_i - \hat{y}_i)^2 | Y^s, Y^{tL}]]$$

An optimal myopic selection is achieved by choosing the point which minimizes the trace of the predictive covariance matrix conditioned on that selection: $\mathrm{Tr}\{\sigma^2_{y|A}\}$.

- **Active surveying** which tries to predict $\sum_i Y^t_i$. The goal is to minimize the risk:

$$\mathbb{E}(\sum_{i \in tU} y_i - \sum_{i \in tU} \hat{y}_i)^2 = \mathbb{E}[\mathbb{E}[(\mathbf{1}^\top \mathbf{y}^{tU} - \mathbf{1}^\top \hat{\mathbf{y}}^{tU})^2 | Y^s, Y^{tL}]]$$

An optimal myopic selection is achieved by choosing the point which minimizes the sum over all elements of the covariance matrix conditioned on that selection [22], which is also denoted Σ-optimality in [41]: $\mathbf{1}^\top \sigma^2_{y|A}\mathbf{1}$.

Note that the predictive covariances for a Gaussian process are computed without using the observed labels. This means that conditioning on hypothetical point selections can be done quickly without needing to marginalize out the unknown label. All that is needed to create an integrated active transfer algorithm using the offset approach from the previous section is to determine the corresponding predictive covariance matrices so the active selection criteria can be applied. We now derive these.

3.4.1 Uncertainty for Offset

Note that the transformation on the training labels is given by

$$Y^{new} = Y^s + \hat{Z}^s,$$

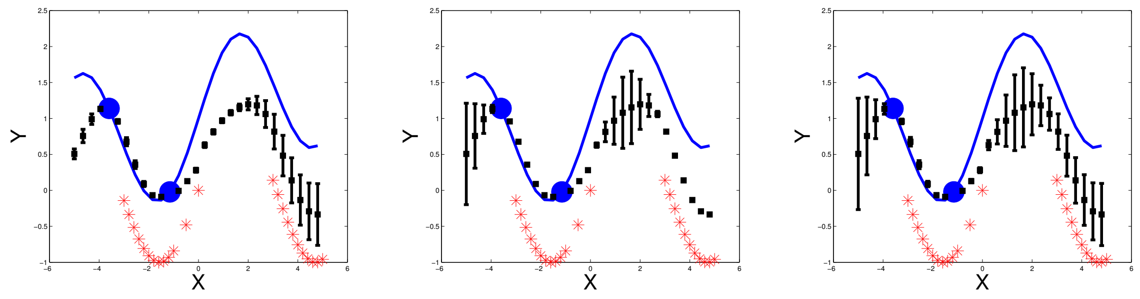


Figure 3.5: Illustration of the uncertainty computed at each step: Uncertainty for offset (left), Uncertainty for prediction given the true offset (middle), The combined uncertainty (right)

in this part we first estimate the distribution of $\hat{Z}^s$, i.e., $P(\hat{Z}^s|X^s, Y^s, X^{tL}, Y^{tL})$. Given

$$P(\hat{Z}^{tL}|X^s, Y^s, X^{tL}, Y^{tL}) \sim \mathcal{N}(\mu_s, \Sigma_s),$$
$$P(\hat{Z}^s|\hat{Z}^{tL}, X^s, X^{tL}) \sim \mathcal{N}(\mu_0, \Sigma_0),$$

from the Offset approach, we can estimate $P(\hat{Z}^s|X^s, Y^s, X^{tL}, Y^{tL})$ by integrating over $\hat{Z}^{tL}$, i.e.,

$$P(\hat{Z}^s|X^s, Y^s, X^{tL}, Y^{tL}) = \int_{\hat{Z}^{tL}} P(\hat{Z}^s, \hat{Z}^{tL}|X^s, Y^s, X^{tL}, Y^{tL}) d(\hat{Z}^{tL})$$
$$= \int_{\hat{Z}^{tL}} P(\hat{Z}^s|\hat{Z}^{tL}, X^s, X^{tL}) P(\hat{Z}^{tL}|X^s, Y^s, X^{tL}, Y^{tL}) d(\hat{Z}^{tL}).$$

Denote $K_1 = K_{X^sX^{tL}}(K_{X^{tL}X^{tL}} + \lambda I)^{-1}$, we can derive that $P(\hat{Z}^s|X^s, Y^s, X^{tL}, Y^{tL}) \sim \mathcal{N}(\mu_1, \Sigma_1)$, where

$$\mu_1 = \Sigma_1 \Sigma_0^{-1} K_1 (K_1^\top \Sigma_0^{-1} K_1 + \Sigma_s^{-1})^{-1} \Sigma_s^{-1} \mu_s,$$
$$\Sigma_1 = \Sigma_0 + K_1 \Sigma_s K_1^\top.$$

In Figure 3.5 (left), we show an example of the offset uncertainty, and we can see that in the places where we do not have any target labels, we have high uncertainty for the offset prediction.

3.4.2 Uncertainty for target label prediction given the true offset

The prediction on X^{tU} is based on the Gaussian Process built from the merged data $\{X^s, Y^{new}\} \cup \{X^{tL}, Y^{tL}\}$, hence it also follows a Gaussian distribution:

$$P(\hat{Y}^{tU}|X^{tU}, X^s, Y^{new}, X^{tL}, Y^{tL}) \sim \mathcal{N}(\mu, \Sigma),$$

where

$$\mu = K_{X^{tU}X}(K_{XX} + \lambda I)^{-1} Y = [\Omega_1 \; \Omega_2][Y^{new} \; Y^{tL}]^\top,$$
$$\Sigma = K_{X^{tU}X^{tU}} - K_{X^{tU}X}(K_{XX} + \lambda I)^{-1} K_{XX^{tU}}.$$

Here X, Y represent the merged data, i.e., $X = X^s \cup X^{tL}, Y = Y^{new} \cup Y^{tL}$. Ω_1 is the matrix consisting of the first m columns of $K_{X^{tU}X}(K_{XX} + \lambda I)^{-1}$, where m is the number of training data points. Ω_2 consists of the remaining l columns, where l is the size of labeled test points.

In Figure 3.5 (middle), we show an example of the prediction uncertainty, and we can see that in the places where we do not have any source data or target labels, we have high uncertainty for the label prediction.

3.4.3 The combined uncertainty for final prediction

Due to the uncertainty for the transformed labels Y^{new}, to model the uncertainty for the final prediction again we need to integrate over Y^{new} , i.e.:

$$\begin{aligned}
&P(Y^{tU}|X^{tU}, X^s, Y^s, X^{tL}, Y^{tL})\\
=&\int_{Y^{new}} P(Y^{tU}, Y^{new}|X^{tU}, X^s, Y^s, X^{tL}, Y^{tL})dY^{new}\\
=&\int_{Y^{new}} P(Y^{tU}|X^{tU}, X^s, Y^{new}, X^{tL}, Y^{tL})P(Y^{new}|X^s, Y^s, X^{tL}, Y^{tL})dY^{new}\\
=&C\int_{Y^{new}} \exp\{-\frac{1}{2}(Y^{tU} - \mu)^\top \Sigma^{-1}(Y^{tU} - \mu)\}\\
&\quad \exp\{-\frac{1}{2}(Y^{new} - Y^s - \mu_1)^\top \Sigma_1^{-1}(Y^{new} - Y^s - \mu_1)\}dY^{new}\\
=&C\int_{Y^{new}} \exp\{-\frac{1}{2}(Y_* - \Omega_1 Y^{new})^\top \Sigma^{-1}(Y_* - \Omega_1 Y^{new})\}\\
&\quad \exp\{-\frac{1}{2}(Y^{new} - Y^s - \mu_1)^\top \Sigma_1^{-1}(Y^{new} - Y^s - \mu_1)\}dY^{new}\\
=&C'\int_{Y^{new}} \exp\{-\frac{1}{2}(Y_* - \Omega_1 Y^{new})^\top \Sigma^{-1}(Y_* - \Omega_1 Y^{new})\}\\
&\quad \exp\{-\frac{1}{2}(Y^{new} - \mu_1)^\top \Sigma_1^{-1}(Y^{new} - \mu_1)\}dY^{new}
\end{aligned}$$

where $Y_* = Y^{tU} - \Omega_2 Y^{tL}$.
Combining the terms related to Y^{new}, we would have

$$\begin{aligned}
&\int_{Y^{new}} \exp\{-\frac{1}{2}[(Y^{new})^\top(\Omega_1^\top \Sigma^{-1}\Omega_1 + \Sigma_1^{-1})Y^{new} - 2(Y_*^\top \Sigma^{-1}\Omega_1 + \mu_1^\top \Sigma_1^{-1})Y^{new}]\}dY^{new}\\
&= C' \exp\{\frac{1}{2}(Y_*^\top \Sigma^{-1}\Omega_1 + \mu_1^\top \Sigma_1^{-1})(\Omega_1^\top \Sigma^{-1}\Omega_1 + \Sigma_1^{-1})^{-1}(Y_*^\top \Sigma^{-1}\Omega_1 + \mu_1^\top \Sigma_1^{-1})^\top\}
\end{aligned}$$

Combining the term related to Y_*, we have Y_* also following a Gaussian distribution with the quadratic term as the following:

$$Y_*^\top(\Sigma^{-1} - \Sigma^{-1}\Omega_1(\Omega_1^\top \Sigma^{-1}\Omega_1 + \Sigma_1^{-1})^{-1}\Omega_1^\top \Sigma^{-1})Y_*$$

which implies that Y_*, or Y^{tU} has a covariance matrix of:

$$\begin{aligned}\Sigma_2 &= (\Sigma^{-1} - \Sigma^{-1}\Omega_1(\Omega_1^\top\Sigma^{-1}\Omega_1 + \Sigma_1^{-1})^{-1}\Omega_1^\top\Sigma^{-1})^{-1} \\ &= \Sigma + \Omega_1\Sigma_1\Omega_1^\top \qquad \text{(matrix inversion lemma)} \\ &= \Sigma + \Omega_1(\Sigma_0 + K_1\Sigma_s K_1^\top)\Omega_1^\top\end{aligned}$$

In conclusion, we have $P(\hat{Y}^{tU}|X^{tU}, X^s, Y^s, X^{tL}, Y^{tL}) \sim \mathcal{N}(\mu_2, \Sigma_2)$, where

$$\begin{aligned}\mu_2 &= \Sigma_2\Sigma^{-1}\Omega_1(\Omega_1^\top\Sigma^{-1}\Omega_1 + \Sigma_1^{-1})^{-1}\Sigma_1^{-1}(\mu_1 + Y^s), \\ \Sigma_2 &= \Sigma + \Omega_1\Sigma_1\Omega_1^\top = \Sigma + \Omega_1(\Sigma_0 + K_1\Sigma_s K_1^\top)\Omega_1^\top.\end{aligned}$$

Hence we get $\mu(\hat{Y}^{tU}) = \Omega_2 Y^{tL} + \Sigma_2\Sigma^{-1}\Omega_1(\Omega_1^\top\Sigma^{-1}\Omega_1 + \Sigma_1^{-1})^{-1}\Sigma_1^{-1}(\mu_1 + Y^s)$.

In Figure 3.5 (right), we show an example of the final prediction uncertainty, and we can see that it is a combination of the previous two kinds of uncertainty we computed.

3.5 Experiments

3.5.1 Synthetic Experiments

Data Description

We generate two synthetic datasets. The first one has a constant shift between the labels Y^s and Y^t. The second one has a shift in both the data points X^s, X^t and their labels Y^s and Y^t. Illustrations for the two datasets are shown as in Figure 3.6.

- Synthetic Dataset 1 (using matlab notation):
  Source: X^s = [-3:0.2:-1 -0.5:0.5:0 3:0.2:5]; $Y^s = \sin(X^s)$;
  Target: X^t = [-5:0.35:5]; $Y^t = \sin(X^t) + 1$.
- Synthetic Dataset 2 (using matlab notation):
  Source: X^s = [-5:0.2:-1 -0.5:0.5:0.5 1:0.2:5]; $Y^s = \sin(X^s)$;
  Target: X^t = [-5:0.35:5]; $Y^t = \sin(X^t + 1)$.

Transfer Learning on Synthetic Dataset

We compare the following methods:

- **Distribution Matching approach**, described in section 3.2.1.
- **Offset approach**, described in section 3.2.2.
- **Use only test x**. GP Prediction using only labeled test points (i.e. no transfer learning).
- **Use both x**. GP Prediction using both training points and labeled test points, without any transfer learning.
- **KMM** for covariate shift [33].
- **Target/Conditional shift**, proposed by [80], code is from `http://people.tuebingen.mpg.de/kzhang/Code-TarS.zip`.

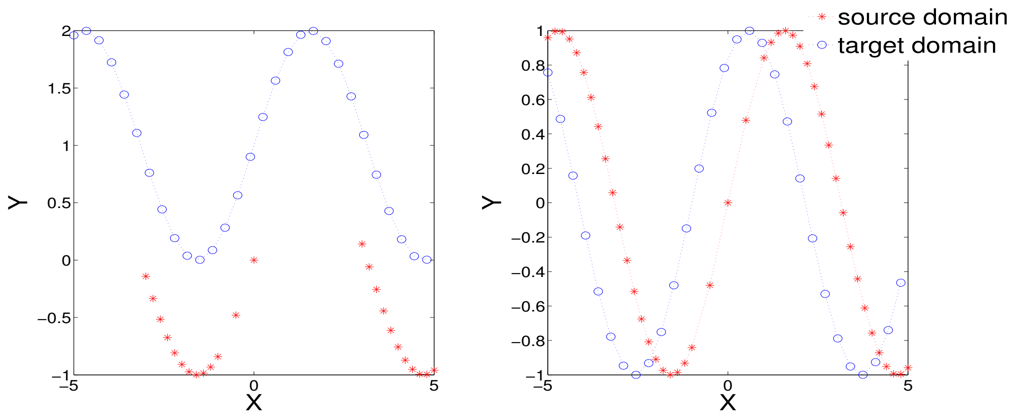


Figure 3.6: Illustration of two synthetic datasets

The evaluation metric is the mean squared error of predictions on the unlabeled test points with different numbers of observed test points with labels. The results are averaged over 10 experiments. Parameters (kernel width, regularization term, etc.) are set using cross validation. In the test domain initially there is not much data for tuning parameters using cross validation, we assume the same smoothness constraint (same kernel width and λ) as in the source domain. The selection of which test points to label is done uniformly at random. Results for the synthetic datasets are shown in Figures 3.7. From the results we can see that for observed test points with labels fewer than 10, our proposed methods can greatly reduce the prediction error by transferring the model learned from the source domain. With more points the errors tend to converge to the error of prediction using only X^{tL}, Y^{tL} because the number of labeled points in the test domain is large enough for learning a good model by itself. KMM and Target/Conditional shift methods do not utilize the possible label information Y^{tL}, hence the error is much larger compared to other methods which use a few Y^t's.

Active Learning/Surveying on Synthetic Dataset

We consider two active learning goals:

- *Active Learning* which reduces the total predictive variance (shortened to *Active Learning*, or AL in the following description)
- *Active Surveying* (AS in the following description)

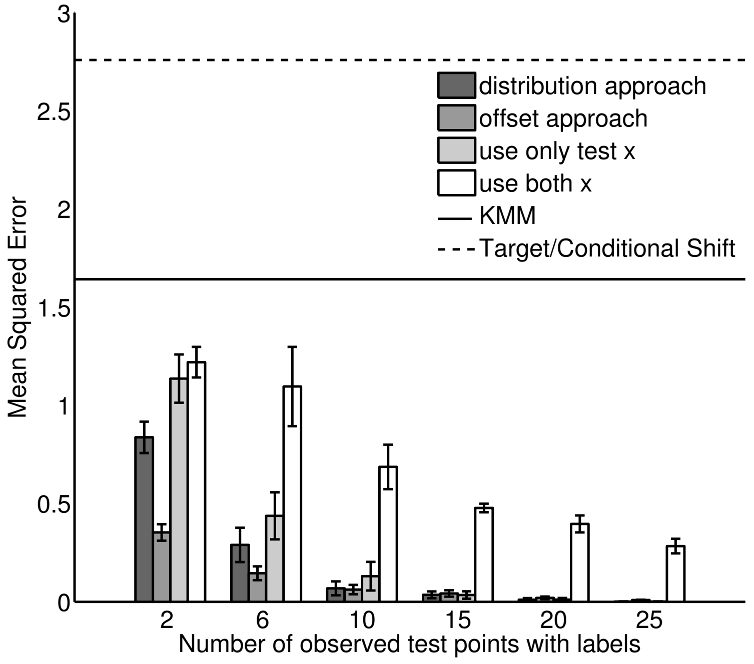
We compare the following uncertainty measures for each goal:

- **combined**. AL/AS using the combined covariance matrix (Σ_2 in section 3.4.3).
- **source**. AL/AS using the covariance matrix based only on the source domain, i.e.,

$$K_{X^{tU}X^{tU}} - K_{X^{tU}X^s}(K_{X^sX^s} + \lambda I)^{-1}K_{X^sX^{tU}}.$$

- **target**. AL/AS using the covariance matrix based only on the target domain, i.e.,

$$K_{X^{tU}X^{tU}} - K_{X^{tU}X^{tL}}(K_{X^{tL}X^{tL}} + \lambda I)^{-1}K_{X^{tL}X^{tU}}.$$



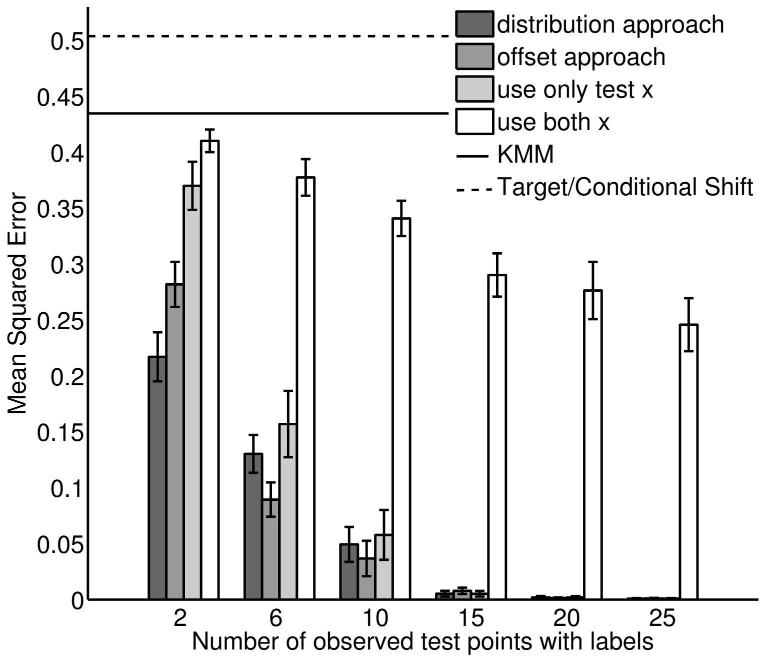


Figure 3.7: MSE for transfer learning on synthetic dataset 1 and 2

- **both**. AL/AS using the covariance matrix based on both source and target domain, i.e.,

$$K_{X^{tU}X^{tU}} - K_{X^{tU}\tilde{X}}(K_{\tilde{X}\tilde{X}} + \lambda I)^{-1}K_{\tilde{X}X^{tU}}, \text{ where } \tilde{X} = X^s \cup X^{tL}.$$

- **random**. Points selected uniformly at random.

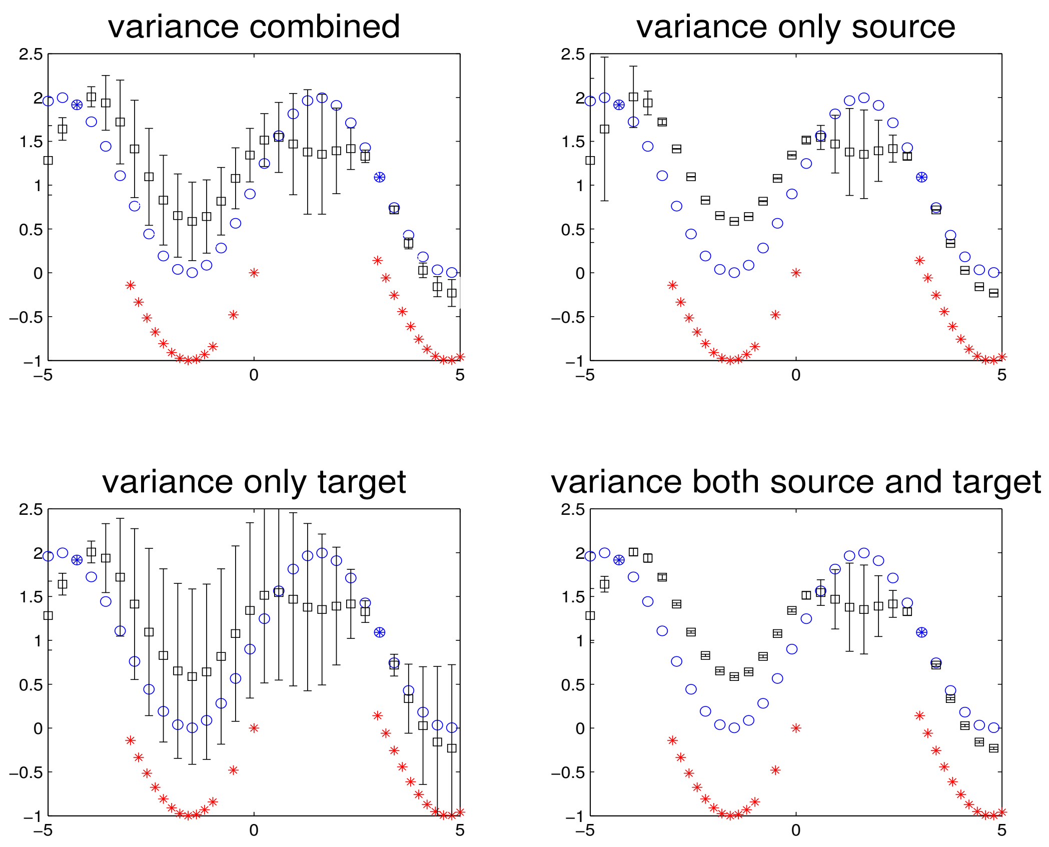


Figure 3.8: The comparison of different covariance matrices.

To better illustrate how the combined covariance matrix compares to other covariance matrices, we show a comparison by plotting the diagonal elements of each covariance matrix, as the uncertainty for prediction on the unlabeled points (with two points labeled) in the test domain, as shown in Figure 3.8. The red stars show the data from the source domain, and the blue circles show the data from the target domain. The black bars show the error bar/uncertainty (diagonal elements of the covariance matrix) on the prediction of unlabeled test points. The two labeled test points are shown in filled blue circles ($x_1 = -4.3, x_2 = 3.05$). Based on what covariance matrix is used for active learning, the most likely selection for the unlabeled test points are:

- **source**: points far away from the source data;
- **target**: points far away from the labeled test points;
- **both**: points far away from both the source data and the labeled test points;
- **combined**: the uncertainty of unlabeled test points will be approximately ranked as (from

highest to lowest), (1) points far away from both the source data and the labeled test points, (2) points far away from the labeled test points but close to the source data, and points far away from the source data but close to the labeled test points, (3) points close to both the source data and the labeled test points.

We consider the mean squared error (for *Active Learning*) and absolute error (for *Active Surveying*) with respect to different number of observed test points with labels (in the order being selected by the corresponding active selection criteria). We averaged the results over 29 experiments, each one initiated with a test point chosen uniformly at random. On Synthetic Dataset 1, *Active Learning* results are shown in Figure 3.9 (left), and *Active Surveying* Results are shown in Figure 3.9 (right). On Synthetic Dataset 2, *Active Learning* results are shown in Figure 3.10 (left), and *Active Surveying* Results are shown in Figure 3.10 (right). From the results we can see that, on Synthetic Dataset 1, for both *Active Learning* and *Active Surveying* our proposed combined covariance matrix (Σ_2 in section 3.3) clearly outperforms all other baselines. On Synthetic Dataset 2, our gain of using combined covariance matrix is smaller because Y^t differs from Y^s at almost every location of X. Hence choosing a point corresponding to a larger transfer learning gain becomes very similar to choosing the point uniformly, which is the selection strategy of using covariance matrix merely based on the target domain.

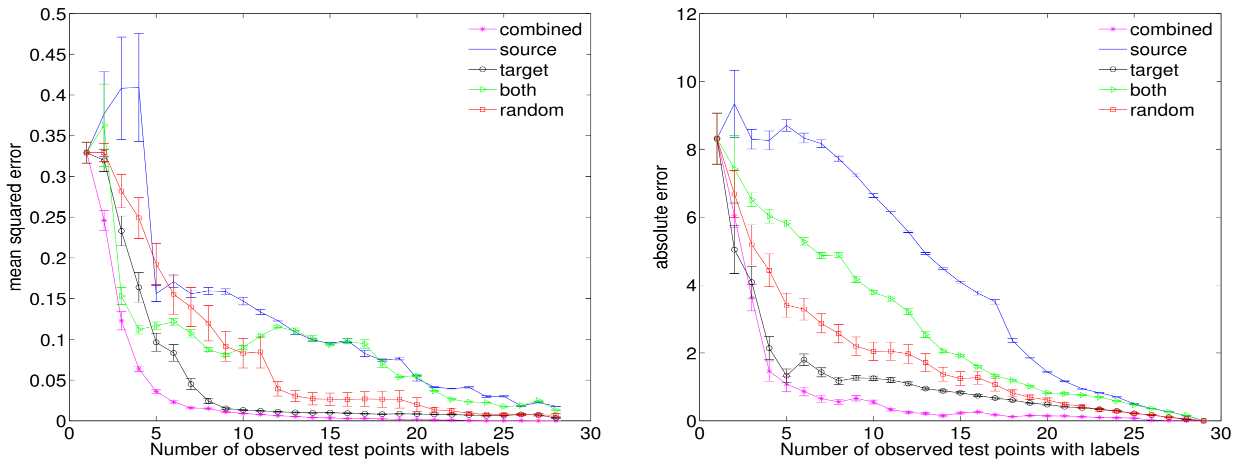
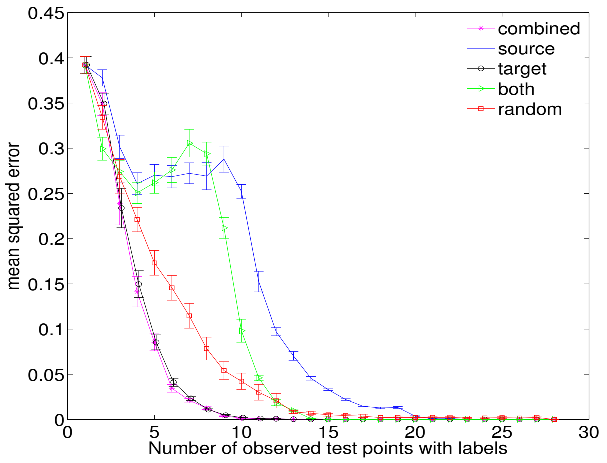


Figure 3.9: MSE for *Active Learning*/*Active Surveying* on Synthetic Dataset 1

3.5.2 Real-world Experiments

Grape Yield Estimation

We have two datasets with grape images and the number of grapes on them as labels, one is riesling (128 labeled images), another is traminette (96 labeled images), as shown in Figure 1.1. The goal is to transfer the model learned from one grape dataset to another grape dataset. The total number of grapes for these two datasets are $19,253$ and $30,360$, respectively.



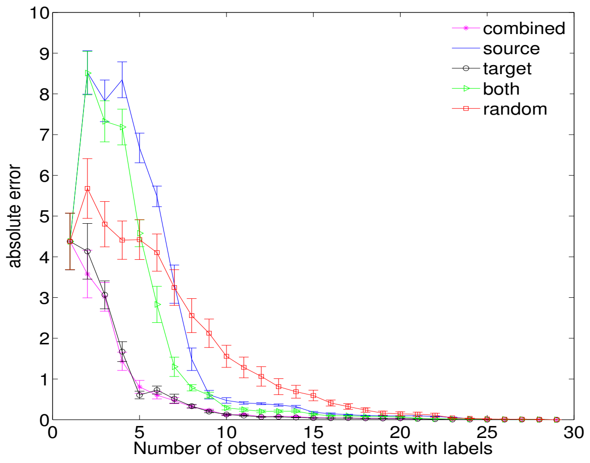


Figure 3.10: MSE for *Active Learning/Active Surveying* on Synthetic Dataset 2

We extract raw-pixel features from the images, and use the methods in [50] to get the coefficients as feature vectors, resulting in 2177 features. We compare to the same baselines for both transfer learning and active learning goals as in the synthetic experiments. For transfer learning the results are shown in Figure 3.11, averaged over 10 experiments. We can see with labeled test points fewer than 25, our proposed approaches (both distribution matching approach and the offset approach) can reduce the error by transferring the model learned from the source domain. The *Active Learning* result is shown in Figure 3.12 (left), and the *Active Surveying* result is shown in Figure 3.12 (right). From the results we can see that our proposed method can well achieve both goals.

AQI Prediction

The dataset is the Air Quality Index (AQI) dataset [45]. We extract bag-of-words vectors (feature X with dimension $d = 100,395$) from social media posts to predict the AQI (label Y) across cities. Fig. 3.13 shows the prediction error using the proposed transfer method, compared with state-of-the-art baselines. The transfer method benefits from modeling a smoother offset across domains compared to optDA [12] with single-source, and it also outperforms KMM [33] by allowing changes in $P(Y|X)$.

3.6 Conclusion

In this chapter we propose algorithms for transfer learning under the model shift assumption. Unlike most existing work that only handles marginal distribution change, our proposed algorithms allow the conditional distributions to change across domains, which is a more challenging case. In addition we propose active learning algorithms under this transfer learning framework, which significantly improves the performance at a small cost of additional label requests. We demon-

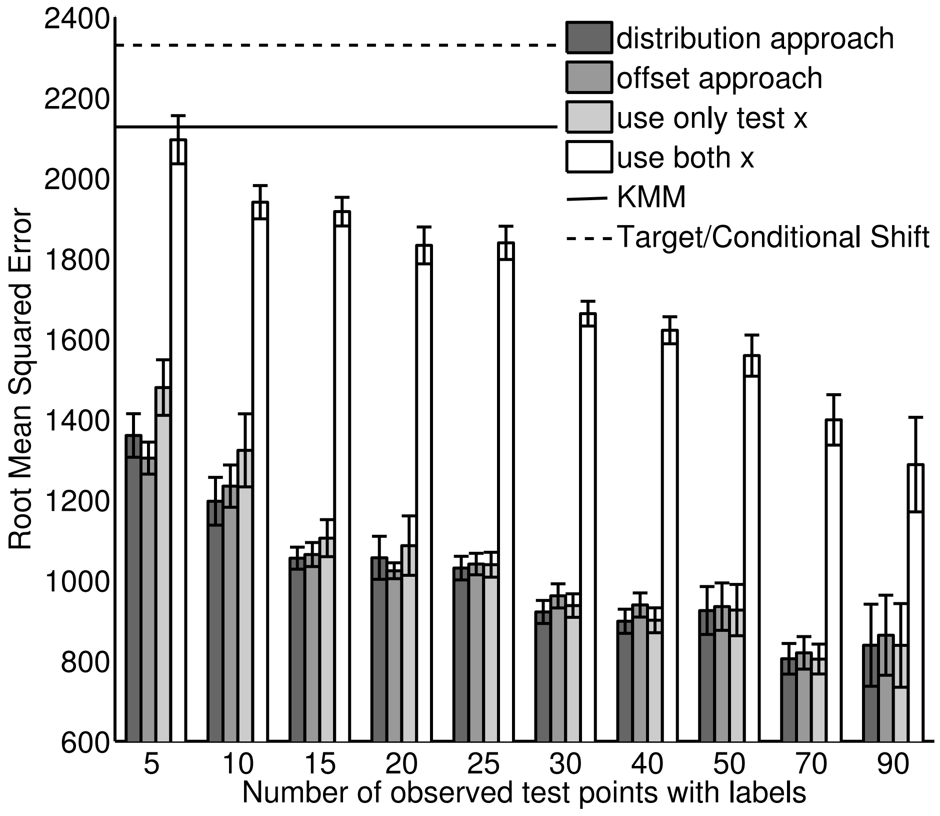


Figure 3.11: RMSE for transfer learning on the real grape data

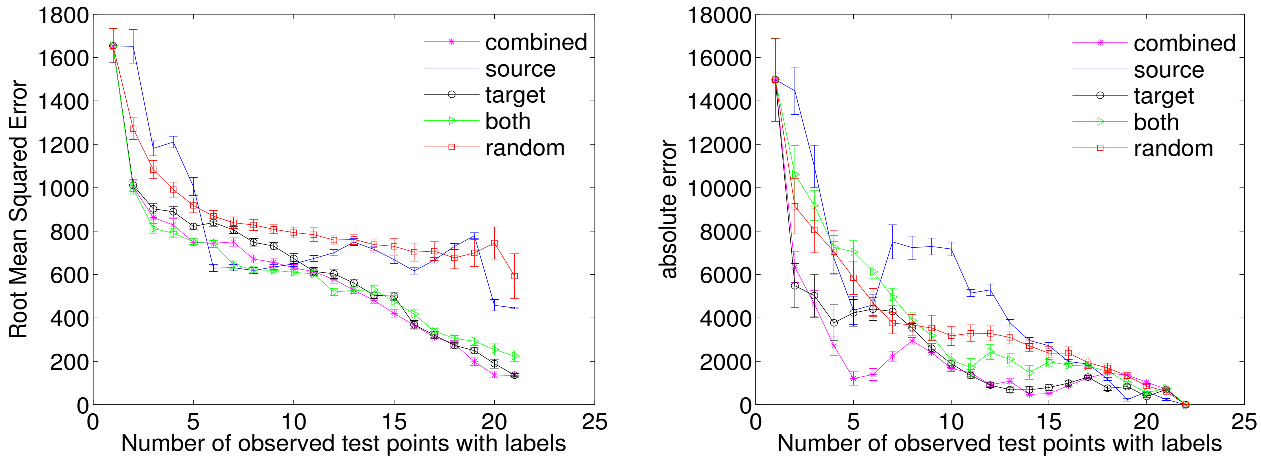


Figure 3.12: RMSE for *Active Learning/Active Surveying* on the real data

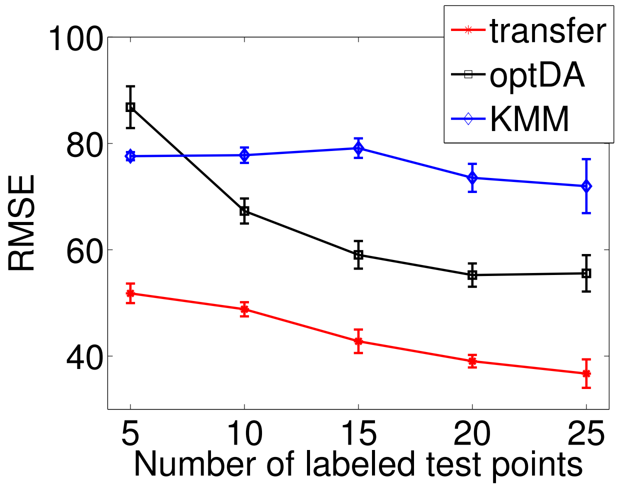


Figure 3.13: Results of transfer learning on the AQI data

strate the effectiveness of our active transfer learning algorithms on both synthetic examples and a real-world grape yield prediction application.

Chapter 4

Generalization Bounds for Transfer Learning under Model Shift

4.1 Overview

As mentioned in the introduction, most transfer learning work is focused on the problem of covariate shift [26, 33, 64]. Theoretical analyses on covariate shift have also been developed in the past, e.g., for sample size m in the source domain and sample size n in the target domain, the analysis of [43] achieves a rate of $O(m^{-1/2} + n^{-1/2})$ for the difference between empirical risk and true risk, and [33] achieves a rate of $O((1/m + 1/n)^{1/2})$ for the convergence of reweighted means in the feature space.

In this chapter, we develop theoretical analysis for transfer learning algorithms under the model shift assumption (an example is shown in Figure 4.1). Specifically, we analyze the **Conditional Distribution Matching** approach (both scale and location shift) and the **Offset** approach (location shift) proposed in Section 3.

Our analysis shows that even when the conditional distributions are allowed to change across domains, we are still able to obtain a generalization bound of $O(\frac{1}{\lambda_* \sqrt{n_l}})$ with respect to the labeled target sample size n_l, modified by the smoothness of the transformation parameters (λ_*) across domains. Our analysis also sheds light on conditions when transfer learning works better than no-transfer learning. We show that under certain smoothness assumptions it is possible to obtain a favorable convergence rate with transfer learning compared to no transfer at all. Furthermore, using the generalization bounds we derived in this section, we are able to extend the transfer learning algorithm from a single source to multiple sources, where each source is assigned a weight that indicates how helpful it is for transferring to the target.

We illustrate our theoretical results by empirical comparisons on both synthetic data and real-world data. Our results demonstrate cases where we obtain the same rate as no-transfer learning, and cases where we obtain a favorable rate with transfer learning under certain smoothness assumptions, which coincide with our theoretical analysis. In addition, experiments on the real data show that our algorithm for reweighting multiple sources yields better results than existing state-of-the-art algorithms.

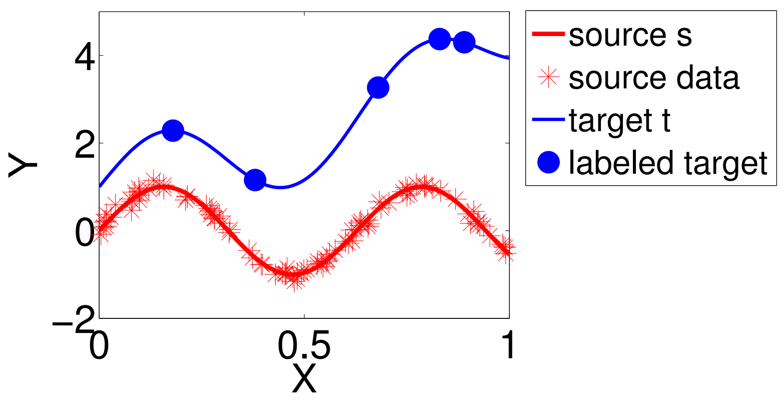


Figure 4.1: Transfer learning example: m source data points $\{X^s, Y^s\}$ (red), n target data points $\{X^t, Y^t\}$ (blue), and n_l labeled target points (solid blue circles). Here X denotes the input features and Y denotes the output labels.

4.2 Transfer Learning under Model Shift: Notations and A Brief Review of the Algorithms

Notation: Let $\mathcal{X} \in \mathcal{R}^d$ and $\mathcal{Y} \in \mathcal{R}$ be the input and output space for both the source and the target domain. We are given a set of m labeled data points, $(x_i^s, y_i^s) \in (X^s, Y^s), i = 1, \ldots, m$, from the source domain. We are also given a set of n target data points, X^t, from the target domain. Among these we have n_l labeled target data points, denoted as (X^{tL}, Y^{tL}). The unlabeled part of X^t is denoted as X^{tU}, with unknown labels Y^{tU}. For simplicity let $z \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ denote the pair of (x, y), and we use z^s, z^t, z^{tL} for the source, target, and labeled target, correspondingly. We assume X^s, X^t are drawn from the same $P(X)$ throughout the section since we focus more on $P(Y|X)$[1]. If necessary $P(X)$ can be easily matched by various methods dealing with covariate shift (e.g. Kernel Mean Matching) without the use of Y.

Let $\mathcal{H}$ be a reproducing kernel Hilbert space with kernel K such that $K(x, x) \leq \kappa^2 < \infty$ for all $x \in X$. Let $||.||_k$ denote the corresponding RKHS norm. Let ϕ denote the feature mapping on x associated with kernel K, and $\Phi(X)$ denote the matrix where the i-th column is $\phi(x_i)$. Denote $K_{XX'}$ as the kernel computed between matrix X and X', i.e., $K_{ij} = k(x_i, x'_j)$. When necessary, we use ψ to denote the feature map on y, and the corresponding matrix as $\Psi(Y)$. For a hypothesis $h \in \mathcal{H}$, assume that $|h(x)| \leq M$ for some $M > 0$. Also assume bounded label set $|y| \leq M$. We use $\ell 2$ loss as the loss function $l(h(x), y)$ throughout this section, which is σ-admissible, i.e.,

$$\forall x, y, \forall h, h', |l(h(x), y) - l(h'(x), y)| \leq \sigma |h(x) - h'(x)|. \tag{4.1}$$

It is easy to see that $\sigma = 4M$ for bounded $h(x)$ and y. Note the loss function is also bounded, $l(h(x), y) \leq 4M^2$.

[1]This assumption is only required in our analysis for simplicity. It can be relaxed when applying the algorithms.

Next we will briefly summarize the two algorithms introduced in the previous section that handle transfer learning under model shift: the first is conditional distribution matching, and the second is two-stage offset estimation.

4.2.1 Conditional Distribution Matching (CDM)

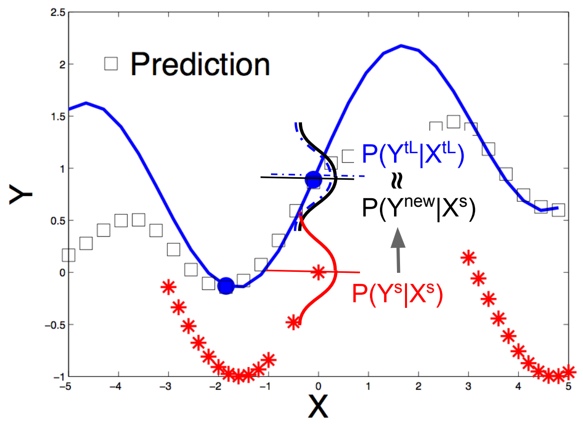


Figure 4.2: Illustration of the conditional distribution matching: red (source), blue (target).

The basic idea of CDM is to match the conditional distributions $P(Y|X)$ for the source and the target domain. Since there is a difference in $P(Y|X)$ across domains, these two conditional distributions cannot be matched directly. Therefore, we propose to make a parameterized-location-scale transform on the source labels Y^s:

$$Y^{new} = Y^s \odot \mathbf{w}(X^s) + \mathbf{b}(X^s),$$

where $\mathbf{w}$ denotes the scale transform, $\mathbf{b}$ denotes the location transform, and $\odot$ denotes the Hadamard (elementwise) product. $\mathbf{w}$ and $\mathbf{b}$ are non-linear functions of X which allows a non-linear transform from Y^s to Y^{new}.

The objective is to use the transformed conditional distribution in the source domain $P(Y^{new}|X^s)$, to match the conditional distribution in the target domain, $P(Y^{tL}|X^{tL})$, such that the transformation parameter $\mathbf{w}$ and $\mathbf{b}$ can be learned through optimization. The matching on $P(Y|X)$ is achieved by minimizing the discrepancy of the conditional embedding operator for $P(Y|X)$ with a regularization term:

$$\begin{aligned} &\min_{\mathbf{w},\mathbf{b}} L + L_{reg}, \text{ where} \\ &\quad L = ||\hat{\mathcal{U}}[P_{Y^{new}|X^s}] - \hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]||_k^2, \\ &\quad L_{reg} = \lambda_{reg}(||\mathbf{w} - \mathbf{1}||^2 + ||\mathbf{b}||^2), \end{aligned} \tag{4.2}$$

where $\mathcal{U}[P_{Y|X}]$ is the conditional embedding operator for $P(Y|X)$ [67], and $\hat{\mathcal{U}}[P_{Y|X}]$ is the empirical estimation of $\mathcal{U}[P_{Y|X}]$ based on samples X, Y. Further we make a smoothness assumption on the transformation, i.e., $\mathbf{w}, \mathbf{b}$ are parameterized using:

$\mathbf{w} = R\mathbf{g}, \mathbf{b} = R\mathbf{h}$, where $R = K_{X^sX^s}(K_{X^sX^s} + \lambda_R I)^{-1}$, and $\mathbf{g}, \mathbf{h} \in \mathbb{R}^{m\times 1}$ are the new parameters to optimize in the objective. After obtaining $\mathbf{g}, \mathbf{h}$ (or equivalently $\mathbf{w}, \mathbf{b}$), Y^{new} is computed based on the transformation. Finally the prediction on X^{tU} is based on the merged data: $(X^s, Y^{new}) \cup (X^{tL}, Y^{tL})$.

Fig 4.2 shows an illustration of the conditional distribution matching algorithm. As we can see from the figure, Y^s is transformed to Y^{new} such that $P(Y^{new}|X^s)$ and $P(Y^{tL}|X^{tL})$ can be approximately matched together.

Remark. Here we analyze what happens when the smoothness assumption is relaxed. It is easy to derive that, when setting $\mathbf{w} = \mathbf{1}, \mathbf{b} = \mathbf{0}$, we can directly solve for Y^{new} by taking the derivative of L with respect to Y^{new}, and we get:

$$K_{X^sX^s}(K_{X^sX^s} + \lambda I)^{-1}Y^{new} = K_{X^sX^{tL}}(K_{X^{tL}X^{tL}} + \lambda I)^{-1}Y^{tL}, \tag{4.3}$$

where λ is some regularization parameter to make sure the kernel matrix is invertible. In other words, the smoothed Y^{new} is given by the prediction on the source using only labeled target data. Hence Y^{new} provides no extra information for prediction on the target, compared with using the labeled target data alone.

4.2.2 Two-stage Offset Estimation (Offset)

The idea of Offset is to model the target function f^t using the source function f^s and an offset, $f^o = f^t - f^s$, while assuming that the offset function is smoother than the target function. Specifically, using kernel ridge regression (KRR) to estimate all three functions, the algorithm works as follows:
(1) Model the source function using the source data, i.e.,

$$f^s(x) = K_{xX^s}(K_{X^sX^s} + \lambda I)^{-1}Y^s.$$

(2) Model the offset function by the difference between the true target labels and the predicted target labels, i.e.,

$$f^o(X^{tL}) = Y^{tL} - f^s(X^{tL}).$$

(3) Transform Y^s to Y^{new} by adding the offset, i.e.,

$$Y^{new} = Y^s + f^o(X^s),$$

where

$$f^o(X^s) = K_{X^sX^{tL}}(K_{X^{tL}X^{tL}} + \lambda I)^{-1}f^o(X^{tL}).$$

(4) Train a model on $\{X^s, Y^{new}\} \cup \{X^{tL}, Y^{tL}\}$, and use the model to make predictions on X^{tU}.

We would like to answer: under what conditions these transfer learning algorithms will work better than no-transfer learning, and how the smoothness assumption affects the generalization bounds for these algorithms.

4.3 Analysis of Conditional Distribution Matching

In this section, we analyze the generalization bound for the conditional distribution matching (**CDM**) approach.

4.3.1 Risk Estimates for CDM

We use stability analysis on the algorithm to estimate the generalization error. First we have:
Theorem 4.3.1. *([11], Theorem 12 and Example 3) Consider a training set*

$$S = \{z_1 = (x_1, y_1), ..., z_m = (x_m, y_m)\}$$

drawn i.i.d. from an unknown distribution D. Let l be the $\ell 2$ loss function which is σ-admissible with respect to $\mathcal{H}$, and $l \leq 4M^2$. The Kernel Ridge Regression algorithm defined by:

$$A_S = \arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} l(h, z_i) + \lambda ||h||_k^2$$

has uniform stability β with respect to l with

$$\beta \leq \frac{\sigma^2 \kappa^2}{2\lambda m}.$$

In addition, let $R = \mathbb{E}_z[l(A_S, z)]$ be the generalization error, and $R_{emp} = \frac{1}{m}\sum_{i=1}^{m} l(A_S, z_i)$ be the empirical error, then the following holds with probability at least $1 - \delta$,

$$R \leq R_{emp} + \frac{\sigma^2 \kappa^2}{\lambda m} + (\frac{2\sigma^2\kappa^2}{\lambda} + 4M^2)\sqrt{\frac{\ln(1/\delta)}{2m}}.$$

In CDM, the prediction on the unlabeled target data points is given by merging the transformed source data and the labeled target data, i.e., $(X^s, Y^{new}) \cup (X^{tL}, Y^{tL})$. Hence we need to bound the difference between the empirical error on the merged data and the generalization error (risk) in the target domain.

Denote $\tilde{z}_i = (\tilde{x}_i, \tilde{y}_i) \in (\tilde{X}, \tilde{Y})$, where $\tilde{X}, \tilde{Y}$ represents the merged data: $\tilde{X} = X^s \cup X^{tL}, \tilde{Y} = Y^{new} \cup Y^{tL}$. Let $h^* \in H$ be the minimizer on the merged data, i.e.,

$$h^* = \arg\min_{h \in \mathcal{H}} \frac{1}{m + n_l} \sum_{i=1}^{m+n_l} l(h, \tilde{z}_i) + \lambda ||h||_k^2.$$

Then the following theorem holds:
Theorem 4.3.2. *Assume the conditions in Theorem 4.3.1 hold. Also assume $||\hat{\mathcal{U}}[P_{Y^{new}|X^s}] - \hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]||_k \leq \epsilon$ after we optimize objective Eq. 4.2. The following holds with probability at least $1 - \delta$:*

$$\begin{aligned}
&|\frac{1}{m + n_l} \sum_{i=1}^{m+n_l} l(h^*, \tilde{z}_i) - E_{z^t}[l(h^*, z^t)]| \\
&\leq 4M(\epsilon\kappa + C(\lambda_c^{1/2} + (n_l\lambda_c)^{-1/2})) + \\
&\quad \frac{\sigma^2\kappa^2}{\lambda_t(m + n_l)} + (\frac{2\sigma^2\kappa^2}{\lambda_t} + 4M^2)\sqrt{\frac{\ln(1/\delta)}{2(m + n_l)}},
\end{aligned}$$

where λ_c is the regularization parameter used in estimating $\hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}] = \Psi(Y^{tL})(K_{X^{tL}X^{tL}} + \lambda_c n_l I)^{-1}\Phi^\top(X^{tL})$, and λ_t is the regularization parameter when estimating the target function. $C > 0$ is some constant.

Proof. Let $\bar{z}_i = (\bar{x}_i, \bar{y}_i) \in (\bar{X}, \bar{Y})$, where $\bar{X}, \bar{Y}$ are the auxiliary samples with $\bar{X} = X^s \cup X^{tL}, \bar{Y} = \bar{Y}_s^t \cup Y^{tL}$, where $\bar{Y}_s^t$ are pseudo labels in the target domain for the source data points X^s. Using triangle inequality we can decompose the LHS by:

$$\begin{aligned}
&|\frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \tilde{z}_i) - E_{z^t}[l(h^*, z^t)]| \\
&\leq |\frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \tilde{z}_i) - \frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \bar{z}_i)| \\
&+ |\frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \bar{z}_i) - E_{z^t}[l(h^*, z^t)]|
\end{aligned}$$

The second term is easy to bound since it is simply the difference between the empirical error and the generalization error in the target domain with effective sample size $n_l + m$, thus using Theorem 4.3.1, we have

$$\begin{aligned}
&|\frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \bar{z}_i) - E_{z^t}[l(h, z^t)]| \\
&\leq \frac{\sigma^2\kappa^2}{\lambda_t(m+n_l)} + (\frac{2\sigma^2\kappa^2}{\lambda_t} + 4M^2)\sqrt{\frac{\ln(1/\delta)}{2(m+n_l)}}.
\end{aligned} \tag{4.4}$$

To bound the first term, we have

$$\begin{aligned}
&|\frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \tilde{z}_i) - \frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \bar{z}_i)| \\
&\leq \frac{1}{m+n_l}\sum_{i=1}^{m+n_l} |l(h^*, \tilde{z}_i) - l(h^*, \bar{z}_i)| \\
&\leq \frac{1}{m+n_l}\sum_{i=1}^{m} 4M|y_i^{new} - \mathcal{U}[P_{Y^t|X^t}]\phi(x_i^s)| \\
&\leq \frac{4M}{m+n_l}\sum_{i=1}^{m}(|\hat{\mathcal{U}}[P_{Y^{new}|X^s}]\phi(x_i^s) - \hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]\phi(x_i^s)| \\
&\qquad + |\hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]\phi(x_i^s) - \mathcal{U}[P_{Y^t|X^t}]\phi(x_i^s)|) \\
&\leq \frac{4M}{m+n_l}\sum_{i=1}^{m}(||\hat{\mathcal{U}}[P_{Y^{new}|X^s}] - \hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]||_k\sqrt{k(x,x)} \\
&\qquad + |\hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]\phi(x_i^s) - \mathcal{U}[P_{Y^t|X^t}]\phi(x_i^s)|) \\
&\leq 4M(\epsilon\kappa + C(\lambda_c^{1/2} + (n_l\lambda_c)^{-1/2})),
\end{aligned} \tag{4.5}$$

where in the last inequality, the second term is bounded using Theorem 6, [67].

Now combining Eq. 4.5 and Eq. 4.4 concludes the proof. □

4.3.2 Tighter Bounds under Smooth Parameterization

Theorem 4.3.2 suggests that using CDM, the empirical risk converges to the expected risk at a rate of

$$O(\lambda_c^{1/2} + (n_l\lambda_c)^{-1/2} + \lambda_t^{-1}(m+n_l)^{-1/2}). \tag{4.6}$$

In the following, we show how the smoothness parameterization in CDM helps us obtain faster convergence rates.

Under the smoothness assumption on the transformation, $\mathbf{w}, \mathbf{b}$ are parameterized using: $\mathbf{w} = R\mathbf{g}, \mathbf{b} = R\mathbf{h}$, where $R = K_{X^sX^s}(K_{X^sX^s} + \lambda_R I)^{-1}$. For simplicity we assume the same λ_R for both $\mathbf{w}$ and $\mathbf{b}$. Similar to the derivation in Eq. 4.5, we have

$$\begin{aligned}
&|y_i^{new} - \mathcal{U}[P_{Y^t|X^t}]\phi(x_i^s)| \\
&= |\hat{\mathcal{U}}[P_{Y^{new}|X^s}]\phi(x_i^s) - \hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]\phi(x_i^s)| + |\hat{\mathcal{U}}[P_{Y^{tL}|X^{tL}}]\phi(x_i^s) - \mathcal{U}[P_{Y^t|X^t}]\phi(x_i^s)|) \\
&\le \epsilon\kappa + |\hat{\mathcal{U}}[P_{w^{tL}|X^{tL}}]\phi(x_i^s) - \mathcal{U}[P_{w^t|X^t}]\phi(x_i^s)| \cdot |y_i^s| + |\hat{\mathcal{U}}[P_{b^{tL}|X^{tL}}]\phi(x_i^s) - \mathcal{U}[P_{b^t|X^t}]\phi(x_i^s)| \\
&\le \epsilon\kappa + C_1(\lambda_R^{1/2} + (n_l\lambda_R)^{-1/2})M + C_2(\lambda_R^{1/2} + (n_l\lambda_R)^{-1/2}) \\
&\le \epsilon\kappa + C'(\lambda_R^{1/2} + (n_l\lambda_R)^{-1/2}).
\end{aligned} \tag{4.7}$$

Hence we can update the bound in Eq. 4.5 by:

$$|\frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \tilde{z}_i) - \frac{1}{m+n_l}\sum_{i=1}^{m+n_l} l(h^*, \bar{z}_i)| \le 4M(\epsilon\kappa + C'(\lambda_R^{1/2} + (n_l\lambda_R)^{-1/2})). \tag{4.8}$$

It is easy to see that Eq. 4.4 remains the same. Hence, the rate for CDM under the smooth parametrization is:

$$O(\lambda_R^{1/2} + (n_l\lambda_R)^{-1/2} + \lambda_t^{-1}(m+n_l)^{-1/2}). \tag{4.9}$$

In transfer learning we usually assume the number of source data is sufficient, i.e., $m \to \infty$. Comparing Eq. 4.9 with Eq. 4.6 we can see that, when the number of labeled points n_l is small, the term $(n_l\lambda_c)^{-1/2}$ in Eq. 4.6 and the term $(n_l\lambda_R)^{-1/2}$ in Eq. 4.9 take over. If we further assume that the transformation $\mathbf{w}$ and $\mathbf{b}$ are smoother functions with respect to X than the target function with respect to X, i.e., $\lambda_R > \lambda_c$, then Eq. 4.9 is more favorable. On the other hand, when the number of labeled target points n_l is large enough for the first term $\lambda_c^{1/2}$ in Eq. 4.6 and the first term $\lambda_R^{1/2}$ in Eq. 4.9 to take over, then it is reasonable to use a λ_R closer to λ_c to get a similar convergence rate as in Eq. 4.6. Intuitively, when the number of labeled target points is large enough, it is not very helpful to transfer from the source for target prediction.

Remark. Note that in Eq. 4.6 and Eq. 4.9, an ideal choice of λ close to $1/\sqrt{n_l}$ can minimize $\lambda^{1/2} + (n_l\lambda)^{-1/2}$. However, note that the generalization bound is the difference between the expected risk R and the empirical risk R_{emp}, and a λ that minimizes the generalization bound does not necessarily minimize the expected risk R, since the empirical risk R_{emp} (which is also affected by λ) can still be large. To obtain a relatively small empirical risk, λ should be determined by the smoothness of the offset/target function, since it is the regularization parameter when estimating the offset/target. In practice λ is chosen by cross validation on the labeled data, and is not necessarily close to $1/\sqrt{n_l}$. For example, on real data we find that λ is usually chosen to be in the range of $1e-2$ to $1e-4$ to accommodate a fairly wide range of functions, which makes the second term $1/\sqrt{n_l \lambda}$ dominate the risk if n_l is much smaller than $1e4$.

Connection with Domain Adaptation Learning Bounds

In [43], the authors provided several bounds on the pointwise difference of the loss for two different hypothesis (Theorem 11, 12 and 13). It is worth noting that in order to bound the pointwise loss, the authors make the following assumptions when the labeling function f_S (source) and f_T (target) are potentially different:

$$\delta^2 = L_{\hat{S}}(f_S(x), f_T(x)) \ll 1,$$

where $L_{\hat{S}}(f_S(x), f_T(x)) = \mathbb{E}_{\hat{S}(x)} l(f_S(x), f_T(x))$. This condition is easily violated under the model shift assumption, where the two labeling functions can differ by a large margin. However, with our transformation from Y^s to Y^{new}, we can translate the above assumption to the following equivalent condition:

$$\begin{aligned}\delta^2 = L_{\hat{S}}(Y^{new}, f_T(x)) &= \frac{1}{m}\sum_{i=1}^{m}(y_i^{new} - \mathcal{U}[P_{Y^t|X^t}]\phi(x_i^s))^2 \\ &\le (\epsilon\kappa + C'(\lambda_R^{1/2} + (n_l\lambda_R)^{-1/2}))^2,\end{aligned}$$

using the results in Eq. 4.7. Hence we can bound δ^2 to be small under reasonable assumptions on n_l and λ_R.

Comparing with No-transfer Learning

Without transfer, which means we predict on the unlabeled target set based merely on the labeled target set, the generalization error bound is simply: $|\frac{1}{n_l}\sum_{i=1}^{n_l} l(h^{tL}, z_i^{tL}) - E_{z^t}[l(h^{tL}, z^t)]| \le \frac{\sigma^2\kappa^2}{\lambda_t n_l} + (\frac{2\sigma^2\kappa^2}{\lambda_t} + 4M^2)\sqrt{\frac{\ln(1/\delta)}{2n_l}}$, where h^{tL} is the KRR minimizer on $\{X^{tL}, Y^{tL}\}$. Then

$$E_{z^t}[l(h^{tL}, z^t)] - \frac{1}{n_l}\sum_{i=1}^{n_l} l(h^{tL}, z_i^{tL}) = O(\frac{1}{\lambda_t\sqrt{n_l}}). \tag{4.10}$$

We can see that with transfer learning, first we obtain a faster rate $O(\lambda_t^{-1}(m+n_l)^{-1/2})$ in Eq. 4.9 with effective sample size $n_l + m$ than $O(\lambda_t^{-1}n_l^{-1/2})$ in Eq. 4.10 with effective sample size n_l. However, the transfer-rate Eq. 4.9 comes with a penalty term $O(\lambda_R^{1/2} + (n_l\lambda_R)^{-1/2})$ which captures the estimation error between the transformed labels and the true target labels. Again, in transfer learning usually we assume $m \to \infty$, and n_l is relatively small, then the transfer-rate becomes $O((n_l\lambda_R)^{-1/2})$. Further if we assume that the smoothness parameter λ_R for the transformation is larger than the smoothness parameter λ_t for the target function ($\lambda_R > \lambda_t$ will be sufficient if $\lambda_R < 1$, otherwise we need to set $\lambda_R > \lambda_t^2$ if $\lambda_R \geq 1$), then we obtain a faster convergence rate with transfer than no-transfer. We will further illustrate the results by empirical comparisons on synthetic and real data in the experimental section.

4.4 Analysis of the Offset Method

In this section, we analyze the generalization error on the two-stage offset estimation (**Offset**) approach. Interestingly, our analysis shows that the generalization bounds for offset and CDM have the same dependency on n_l.

4.4.1 Risk Estimates for Offset

First, we learn a model from the source domain by minimizing the squared loss on the source data, i.e.,

$$h^s = \arg\min_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} l(h, z_i^s) + \lambda_s ||h||_k^2.$$

Using Theorem 4.3.1, we have with probability at least $1-\delta$,

$$R^s \leq R^s_{emp} + \frac{\sigma^2\kappa^2}{\lambda_s m} + (\frac{2\sigma^2\kappa^2}{\lambda_s} + 4M^2)\sqrt{\frac{\ln(1/\delta)}{2m}},$$

where $R^s = E_{z^s}[l(h^s, z^s)]$, $R^s_{emp} = \frac{1}{m}\sum_{i=1}^{m} l(h^s, z_i^s)$. Hence

$$R^s - R^s_{emp} = O(\frac{1}{\lambda_s\sqrt{m}}), \tag{4.11}$$

Second, we learn the offset by KRR on $\{X^{tL}, \hat{y}^o\}$, where $\hat{y}^o = Y^{tL} - f^s(X^{tL})$, i.e., $\hat{y}^o$ is the estimated offset on labeled target points X^{tL}, and $f^s(X^{tL})$ is the prediction on X^{tL} using source data.

Denote $\hat{h}^o$ as the minimizer on $\hat{z}^o = \{X^{tL}, \hat{y}^o\}$, i.e.,

$$\begin{aligned}\hat{h}^o &= \arg\min_{h\in\mathcal{H}} \frac{1}{n_l}\sum_{i=1}^{n_l} l(h, \hat{z}_i^o) + \lambda_o ||h||_k^2 \\ &= \arg\min_{h\in\mathcal{H}} R(h) + N(h).\end{aligned} \tag{4.12}$$

Denote h^o as the minimizer on $z^o = \{X^{tL}, y^o\}$, where y^o is the unknown true offset:

$$\begin{aligned}h^o &= \arg\min_{h\in\mathcal{H}} \frac{1}{n_l}\sum_{i=1}^{n_l} l(h, z_i^o) + \lambda_o ||h||_k^2 \\ &= \arg\min_{h\in\mathcal{H}} R'(h) + N(h),\end{aligned} \tag{4.13}$$

Using Theorem 4.3.1, we have with probability at least $1 - \delta$,

$$R^o \leq R^o_{emp} + \frac{\sigma^2 \kappa^2}{\lambda^o n_l} + (\frac{2\sigma^2\kappa^2}{\lambda^o} + 4M^2)\sqrt{\frac{\ln(1/\delta)}{2n_l}} \tag{4.14}$$

where $R^o = E_{z^o}[l(h^o, z^o)], R^o_{emp} = \frac{1}{n_l}\sum_{i=1}^{n_l} l(h^o, z^o_i)$. In our estimation we use $\hat{y}^o$ instead of y^o, hence we need to account for this estimation error.

Lemma 4.4.1. *The generalization error R^o is bounded by:*

$$R^o = \bar{R}^o_{emp} + O(\frac{1}{\lambda_o \sqrt{n_l}}), \tag{4.15}$$

as $m \to \infty$. Here $\bar{R}^o_{emp} = \frac{1}{n_l}\sum_{i=1}^{n_l} l(\hat{h}^o, \hat{z}^o_i)$ is the empirical error of our estimator $\hat{h}^o$ on $\{X^{tL}, \hat{y}^o\}$.

Proof. Define the Bregman Divergence associated to F of f to g by $B_F(f||g) = F(f) - F(g) - < f - g, \nabla F(g) >$.

Let $F(h) = R(h) + N(h), F'(h) = R'(h) + N(h)$. Since $h^o, \hat{h}^o$ are the minimizers, we have

$$\begin{aligned} &B_{F'}(\hat{h}^o||h^o) + B_F(h^o||\hat{h}^o) \\ =&F'(\hat{h}^o) - F'(h^o) + F(h^o) - F(\hat{h}^o) \\ =&R'(\hat{h}^o) - R'(h^o) + R(h^o) - R(\hat{h}^o). \end{aligned}$$

In addition, using the nonnegativity of B and

$$B_F = B_R + B_N, B_{F'} = B_{R'} + B_N,$$

we have

$$B_N(\hat{h}^o||h^o) + B_N(h^o||\hat{h}^o) \leq B_F(h^o||\hat{h}^o) + B_{F'}(\hat{h}^o||h^o).$$

Combining the two we have

$$\begin{aligned} &B_N(\hat{h}^o||h^o) + B_N(h^o||\hat{h}^o) \leq R'(\hat{h}^o) - R'(h^o) + R(h^o) - R(\hat{h}^o) \\ =&\frac{1}{n_l}\sum_{i=1}^{n_l} l(\hat{h}^o, z^o_i) - \frac{1}{n_l}\sum_{i=1}^{n_l} l(h^o, z^o_i) + \frac{1}{n_l}\sum_{i=1}^{n_l} l(h^o, \hat{z}^o_i) - \frac{1}{n_l}\sum_{i=1}^{n_l} l(\hat{h}^o, \hat{z}^o_i) \\ \leq&\frac{2}{n_l}\sum_{i=1}^{n_l} \sigma|y^o_i - \hat{y}^o_i|, \end{aligned}$$

using $|l(\hat{h}^o, z^o_i) - l(\hat{h}^o, \hat{z}^o_i)| \leq |2\hat{h}^o(x_i) - y^o_i - \hat{y}^o_i| \cdot |y^o_i - \hat{y}^o_i| \leq \sigma|y^o_i - \hat{y}^o_i|$, $\sigma = 4M$.

Since for RKHS norm $B_N(f||g) = ||f - g||^2_k$, we have

$$B_N(\hat{h}^o||h^o) + B_N(h^o||\hat{h}^o) = 2||h^o - \hat{h}^o||^2_k.$$

Combined with the above inequality, we have $2||h^o - \hat{h}^o||_k^2 \leq \frac{2}{n_l}\sum_{i=1}^{n_l} \sigma|y_i^o - \hat{y}_i^o|$.
Then we have

$$|l(h^o, z_i^o) - l(\hat{h}^o, z_i^o)| \leq \sigma|h^o(x_i) - \hat{h}^o(x_i)| \leq \sigma||h^o - \hat{h}^o||_k\kappa \leq \sigma\kappa\sqrt{\frac{1}{n_l}\sum_{i=1}^{n_l}\sigma|y_i^o - \hat{y}_i^o|}.$$

Hence

$$\begin{aligned}
&|\frac{1}{n_l}\sum_{i=1}^{n_l} l(h^o, z_i^o) - \frac{1}{n_l}\sum_{i=1}^{n_l} l(\hat{h}^o, \hat{z}_i^o)| \\
\leq &\frac{1}{n_l}\sum_{i=1}^{n_l}[|l(h^o, z_i^o) - l(\hat{h}^o, z_i^o)| + |l(\hat{h}^o, z_i^o) - l(\hat{h}^o, \hat{z}_i^o)|] \\
\leq &\sigma\kappa\sqrt{\frac{1}{n_l}\sum_{i=1}^{n_l}\sigma|y_i^o - \hat{y}_i^o|} + \frac{1}{n_l}\sum_{i=1}^{n_l}\sigma|y_i^o - \hat{y}_i^o|.
\end{aligned}$$

Now we can conclude that

$$R_{emp}^o = \frac{1}{n_l}\sum_{i=1}^{n_l} l(h^o, z_i^o) \leq \bar{R}_{emp}^o + P, \tag{4.16}$$

where $\bar{R}_{emp}^o = \frac{1}{n_l}\sum_{i=1}^{n_l} l(\hat{h}^o, \hat{z}_i^o)$, and $P = \sigma\kappa\sqrt{\frac{1}{n_l}\sum_{i=1}^{n_l}\sigma|y_i^o - \hat{y}_i^o|} + \frac{1}{n_l}\sum_{i=1}^{n_l}\sigma|y_i^o - \hat{y}_i^o|$.
To bound P, first we have

$$\begin{aligned}
\frac{1}{n_l}\sum_{i=1}^{n_l}|y_i^o - \hat{y}_i^o| =&\frac{1}{n_l}\sum_{i=1}^{n_l}|(y_i^{tL} - y_i^s) - (y_i^{tL} - \hat{y}_i^s)| \\
=&\frac{1}{n_l}\sum_{i=1}^{n_l}|y_i^s - \hat{y}_i^s| \\
\leq&\sqrt{\frac{1}{n_l}\sum_{i=1}^{n_l}(y_i^s - \hat{y}_i^s)^2}.
\end{aligned}$$

Using Eq. 4.11, $\frac{1}{n_l}\sum_{i=1}^{n_l}(y_i^s - \hat{y}_i^s)^2$ is bounded by $R_{emp}^s + O(\frac{1}{\lambda_s\sqrt{m}})$. We can see that the penalty term P diminishes as $m \to \infty$. Plugging Eq. 4.16 into Eq. 4.14 concludes the proof. $\square$

Now we analyze the generalization error in the target domain. Using the assumption that the target labels y^t can also be decomposed by $y^o + y^s$, we have:

$$\begin{aligned}
&\mathbb{E}_{z^t}[l(h, z^t)] = \mathbb{E}_{z^t}[(h(x^t) - y^t)^2] \\
&= \mathbb{E}[(h^o(x^t) + h^s(x^t) - y^o - y^s)^2] \\
&\leq 2\mathbb{E}(h^o(x^t) - y^o)^2 + 2\mathbb{E}(h^s(x^t) - y^s)^2.
\end{aligned} \tag{4.17}$$

Plugging in Eq. 4.11 and Eq. 4.15, we have

$$R^t = \mathbb{E}_{z^t}[l(h, z^t)] = 2R^s_{emp} + 2\bar{R}^o_{emp} + O(\frac{1}{\lambda_o\sqrt{n_l}} + \frac{1}{\lambda_s\sqrt{m}})$$

In transfer learning usually we assume that the number of source data is sufficient, i.e., $m \to \infty$, hence

$$R^t - 2(R^s_{emp} + \bar{R}^o_{emp}) = O(\frac{1}{\lambda_o\sqrt{n_l}}). \tag{4.18}$$

Comparing with No-transfer Learning

As with the no-transfer-rate in Sec. 4.3.2, we have

$$R^t - R^{tL}_{emp} = O(\frac{1}{\lambda_t\sqrt{n_l}}), \tag{4.19}$$

where λ_t is the regularization parameter when estimating the target function. Comparing this rate with Eq. 4.18, and using our assumption that we have a smoother offset than the target function, i.e., $\lambda_o > \lambda_t$, we can see that we obtain a faster convergence rate with transfer than no-transfer.

4.5 Multi-Source Transfer Learning

In this section, we show that we can easily adapt the transfer learning algorithm from a single source to transfer learning with multiple-sources, by utilizing the generalization bounds we derived in earlier sections. Transfer learning with multiple sources is similar to multi-task learning, where we learn the target and multiple sources jointly.

A closer look at Eq. 4.9 for CDM, and Eq. 4.18 for Offset reveals that, when n_l is small and $m \to \infty$, we have a convergence rate of $O(\frac{1}{\lambda_*\sqrt{n_l}})$ for both algorithms, where λ_* is some parameter that controls the smoothness of the source-to-target transfer (for Eq. 4.9 we can set $\lambda_R = \lambda_*^2$). This observation motivates our reweighting scheme on the source hypotheses to achieve transfer learning under multiple sources, described as the following.

Assume we have S sources and a target. First, we apply the transfer learning algorithm from a single source to obtain a model M_s from each source s to target t, where the parameter λ_*^s is determined by cross-validation, $s = 1, ..., S$. Second, we compute the weight for each source s by:

$$w_s = p(D|M_s)p(M_s), \text{ where}$$
$$p(D|M_s) = \exp\{-\sum_{i=1}^{m_s}(y_i^{tL} - \hat{f}^s(x_i^{tL}))^2\},$$
$$p(M_s) \propto \exp\{-\alpha\frac{1}{\lambda_*^s}\},$$

where $\hat{f}^s(x_i^{tL})$ is the prediction given by M_s.

The idea is similar to Bayesian Model Averaging [31], where the first term $p(D|M_s)$ serves as the data likelihood of the predictive model M_s from source s, and the second term $p(M_s)$ is the prior probability on model M_s. In our case, $p(M_s)$ is chosen to indicate how similar each source to the target is, where the similarity is measured by how smooth the change is from source s to target t. It is easy to see that, the weights coincide with our analysis of the generalization bounds for transfer learning, and the choice of α should be in the order of $O(1/\sqrt{n_l})$. Intuitively, when the number of labeled target points n_l is small, $p(M_s)$ has a larger effect on w_s, which means we prefer the source that has a smoother change (larger λ_*^s) for the transfer. On the other hand, when n_l is large, then $p(D|M_s)$ takes over, i.e., we prefer the source that results in a larger data likelihood (smaller prediction errors). Finally, we combine the predictions by:

$$\hat{f}(x_i^{tU}) = \sum_{s=1}^{S} \frac{w_s}{\sum_{s=1}^{S} w_s} \hat{f}^s(x_i^{tU})$$

This weighted combination of source hypotheses gives us the following generalization bound in the target domain:

$$\begin{aligned}
&\mathbb{E}_{z^t}[l(h, z^t)] = \mathbb{E}_{z^t}[(\sum_s \frac{w_s}{\sum_{s=1}^{S} w_s} h_s(x^t) - y^t)^2] \\
&= \mathbb{E}_{z^t}[(\sum_s \frac{w_s}{\sum_{s=1}^{S} w_s} (h_s(x^t) - y^t))^2] \\
&\leq \sum_s \frac{w_s}{\sum_{s=1}^{S} w_s} \mathbb{E}_{z^t}[(h_s(x^t) - y^t)^2] \\
&= \sum_s \frac{w_s}{\sum_{s=1}^{S} w_s} [\tilde{R}^s_{emp} + O(\frac{1}{\lambda_s \sqrt{n_l}})],
\end{aligned}$$

where the third inequality uses Jensen's inequality, and the last equality uses the bounds we derived. Here $\tilde{R}^s_{emp}$ refers to the empirical error for source s when transferring from s to t (Thm. 4.3.2 for CDM and Eq. 4.18 for Offset).

4.6 Experiments

4.6.1 Synthetic Experiments

In this section, we empirically compare the generalization error of transfer learning algorithms to that of no-transfer learning (learning by labeled target data only), on synthetic datasets simulating different conditions.

We generate the synthetic dataset in this way: X^s, X^t are drawn uniformly at random from $[0, 4]$, $Y^s = \sin(2X^s) + \sin(3X^s)$ with additive Gaussian noise. Y^t is the same function with a smoother location-scale transformation/offset. In each of the following comparisons, we plot the mean squared error (MSE) on the unlabeled target points (as an estimation of the generalization error) with respect to different number of labeled target points. The labeled target points are chosen uniformly at random, and we average the error over 10 experiments. The parameters are chosen using cross validation.

In Fig. 4.3, we compare transfer learning using CDM with no-transfer learning. The results show that with the additional smoothness assumption, we are able to achieve a much lower generalization error for transfer learning than no-transfer learning. In Fig. 4.4 and 4.5, we compare transfer learning using the Offset approach with no-transfer learning. The two figures show different generalization error curves when the smoothness of the offset is different. We can see that with a smoother offset (Fig. 4.4) we are able to achieve a much lower generalization error than no-transfer learning. With a less smooth offset (Fig. 4.5) we can still achieve a lower generalization error than no-transfer learning, but the rate is slower compared to Fig. 4.4. Further we analyze the case when the smoothness assumption does not hold, by setting the source function to be $\sin(X^s) + \epsilon$ such that the target changes faster than the source. In this case, transfer learning with CDM/Offset yield almost the same generalization error as no-transfer learning (Fig. 4.6), i.e., the source data does not help in learning the target.

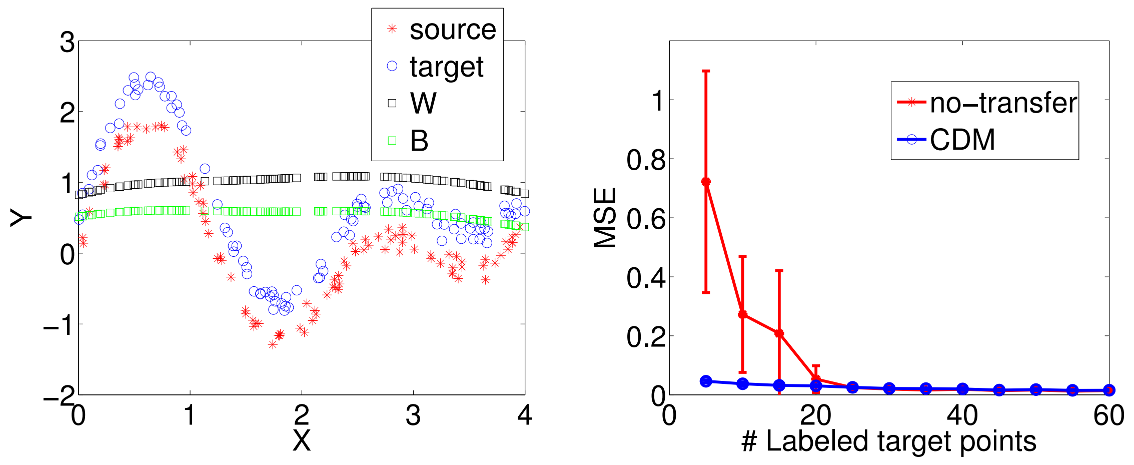


Figure 4.3: No-transfer learning vs. transfer learning (CDM)

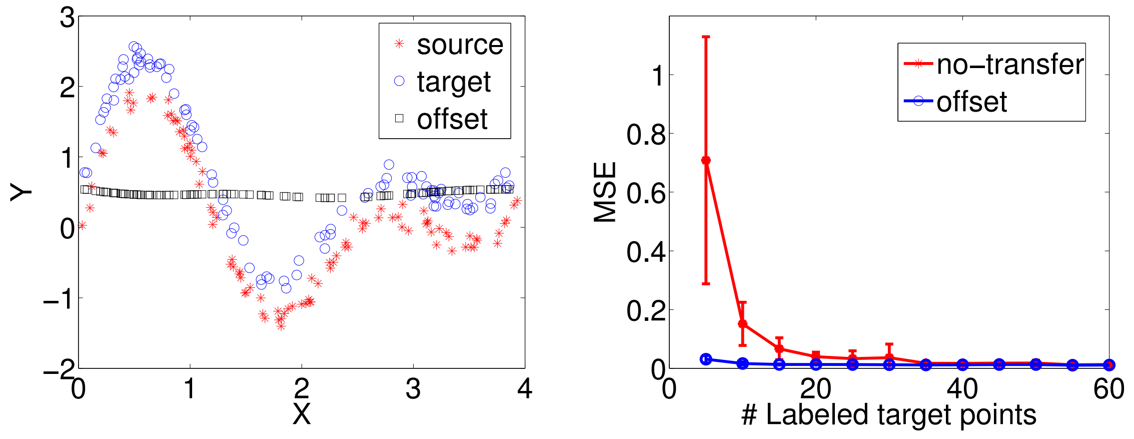


Figure 4.4: No-transfer learning vs. transfer learning using the Offset approach (smoother offset, $\lambda_R = 0.1$)

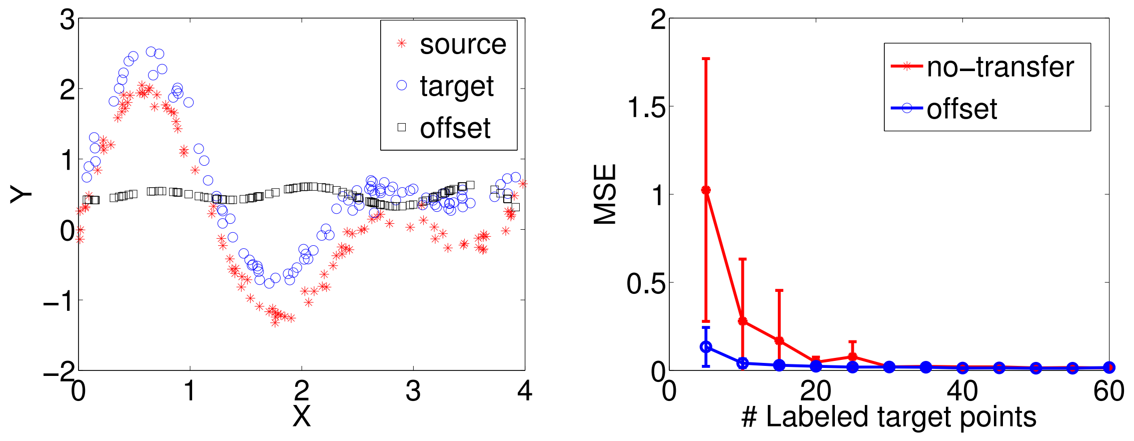


Figure 4.5: No-transfer learning vs. transfer learning using the Offset approach (less smooth offset, $\lambda_R = 0.001$)

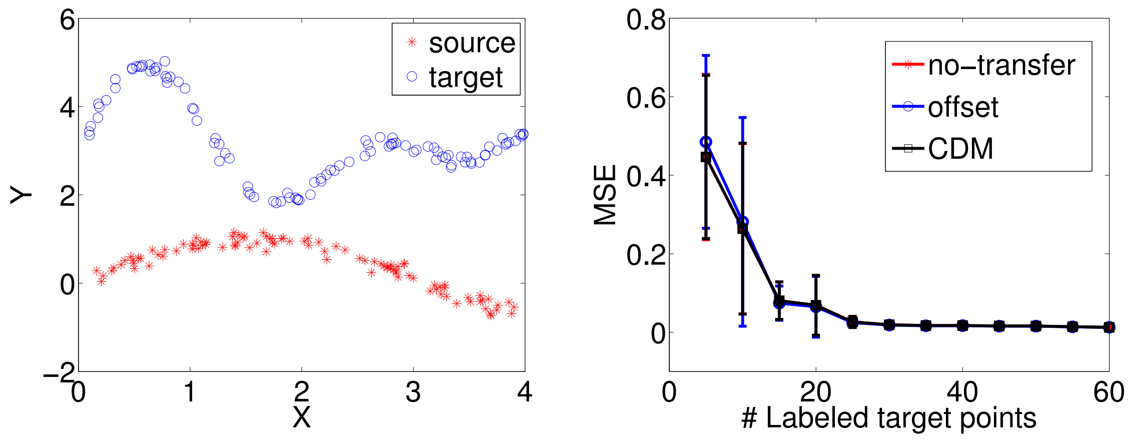


Figure 4.6: No-transfer learning vs. transfer learning, when the smoothness assumption does not hold

4.6.2 Experiments on the Real Data

Comparing Transfer Learning to No-transfer Learning, Using Different Sources

The real-world dataset is an Air Quality Index (AQI) dataset [45] during a 31-day period from Chinese cities. For each city, the input feature x_i is a bag-of-words vector extracted from Weibo posts of each day, with $100,395$ dimensions as the dictionary size. The output label y_i is a real number which is the AQI of that day.

Fig. 4.7 shows a comparison of MSE on the unlabeled target points, with respect to different number of labeled target points, when transferring from a nearby city (Ningbo) and a faraway city (Xi'an), to a target city (Hangzhou). The data is shown in the left figure of Fig. 4.7, where the x-axis is each day. The results are averaged over 20 experiments with uniformly randomly chosen labeled target points. First we observe that we obtain a lower generalization error by transferring from other cities than learning by the target city data alone (no-transfer). In addition, the generalization error are much lower if we transfer from nearby cities where the difference between source and target is smoother.

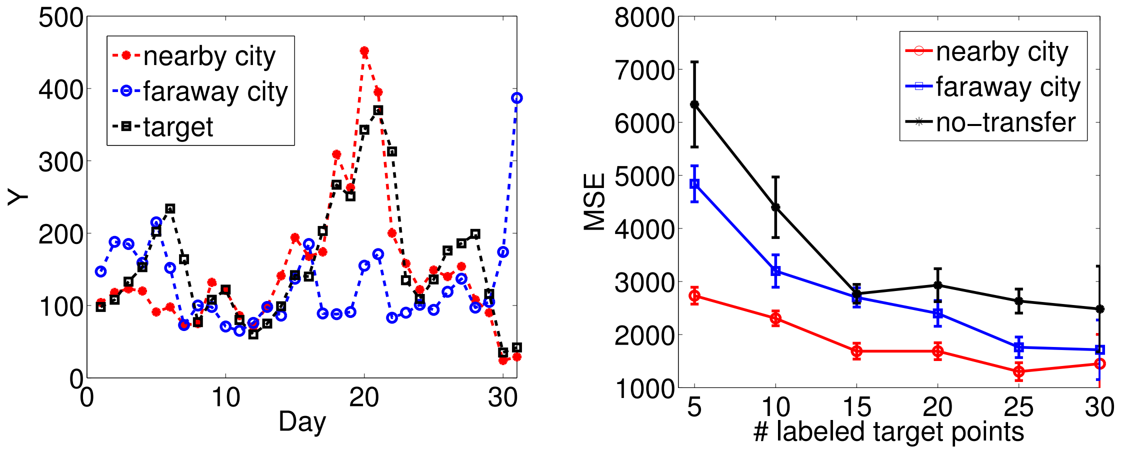


Figure 4.7: Comparison of MSE on unlabeled target points

Transfer Learning with Multiple Sources

The results in Sec. 4.6.2 indicate that, when transferring from multiple sources to a target, it is important to choose which source to transfer, in order to obtain a larger gain. In this section, we show the results on the same air quality index data by reweighting different sources (Sec. 4.5).

Fig. 4.8 shows a comparison of MSE on the unlabeled target data (data shown in the left figure) with respect to different number of labeled target points ($n_l \in \{2, 5, 10, 15, 20\}$), where the prediction is based on each source independently (labeled as **source** i, $i \in \{1, 2, 3\}$), and based on multiple sources (labeled as **posterior**). Since CDM and Offset give similar bounds, we use two-stage offset estimation as the prediction algorithm from each source s to target t. The weighting on the sources is as described in Sec. 4.5. As can be seen from the results, using posterior reweighting on different sources, we obtain results that are very close to the results using the best source.

Further in Figure. 4.9, we show a comparison of MSE on the unlabeled target data between the proposed approach and two baselines, with respect to different number of labeled target points. The results are averaged over 20 experiments. The first baseline **wDA** is a weighted multi-source domain adaptation approach proposed in [42], where the distribution $D_i(x)$ for source i on a target point x is estimated using kernel density estimation with a Gaussian kernel. Note that the original algorithm proposed in [42] does not assume the existence of a few labeled target points, thus the hypothesis $h_i(x)$ from each source i is computed by using the source data only. To ensure a fair comparison, we augment $h_i(x)$ by using the prediction of the Offset approach given n_l labeled target points. The second baseline **optDA** is a multi-source domain adaptation algorithm under an optimization framework, as proposed in [12], where the parameters γ_A, γ_I are set as described in the section, and θ is chosen using cross-validation on the set $\{0.1, 0.2, ..., 0.9\}$

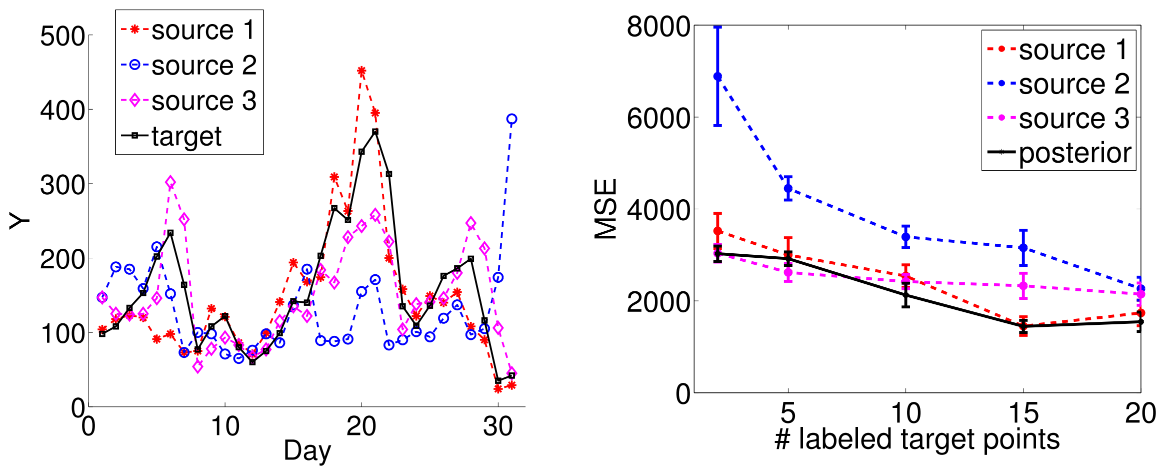


Figure 4.8: Comparison of MSE on unlabeled target points, with multiple sources

(the final choice of θ is 0.1). Note that our proposed algorithm gives the best performance. In addition, our algorithm does not require density estimation as in **wDA**, which can be difficult in real-world applications with high-dimensional features. Further note **posterior** considers the change in $P(Y|X)$ while **wDA** focuses on the change of $P(X)$. A potential improvement can be achieved by combining these two in the reweighting scheme, which should be an interesting future direction.

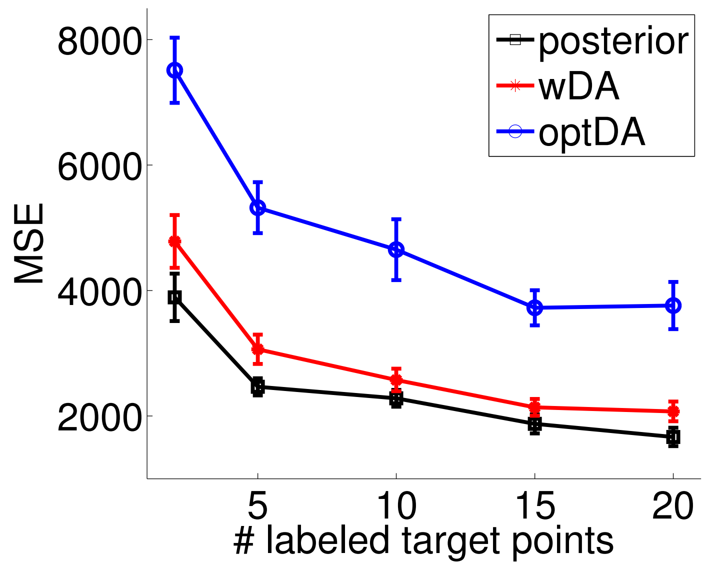


Figure 4.9: Multi-source transfer learning: comparison of MSE on the proposed approach (**posterior**) and baselines

4.7 Conclusion and Discussion

In this chapter, we provide theoretical analysis for algorithms proposed for transfer learning under the model shift assumption. Unlike previous work on covariate shift, the model shift poses a harder problem for transfer learning, and our analysis shows that we are still able to achieve a similar rate as in covariate shift/domain adaptation, modified by the smoothness of the transformation parameters. We also show conditions when transfer learning works better than no-transfer learning. Finally we extend the algorithms to transfer learning with multiple sources.

The choice of λ_* is determined by cross validation on the training data. One might think we are able to choose the best λ_* by minimizing the generalization bound, but the problem is that λ_* also affects the empirical error, hence the best λ_* that minimizes the generalization bound $R - R_{emp}$ might not minimize the generalization error R.

Note that although CDM has both location and scale shift and Offset has only location shift, the two methods are comparable because the scale shift can be translated into location shift. On the other hand, since CDM has both shifts one might think we need more samples to learn the transformations due to the increase of the number of parameters, compared to the Offset approach. Indeed CDM takes relatively longer time to optimize than Offset in our real-data experiments, but the results of these two approaches are comparable with the same number of samples. This is because we introduced the smoothness constraints for CDM, thus reducing the effective number of parameters for both location and scale shift.

Chapter 5

Flexible Transfer Learning under Support and Model Shift

5.1 Overview

In this chapter, we propose a flexible algorithm that allows both the support on X and Y, and the model $P(Y|X)$ to change across the source and target domains. We assume only that the change is smooth as a function of X. In this way, more flexible transformations are allowed than mean-centering and variance-scaling.

Specifically, we build a Gaussian Process to model the prediction on the transformed X, then the prediction is matched with a few observed labels Y (also properly transformed) available in the target domain such that both transformations on X and on Y can be learned.

5.2 Transfer Learning

5.2.1 Proposed Algorithm

Our strategy is to simultaneously learn a nonlinear mapping $X^t \to X^{new}$ and $Y^t \to Y^*$. This allows flexible transformations on both X and Y, and our smoothness assumption using GP prior makes the estimation stable. We call this method Support and Model Shift (SMS).

We apply the following steps (K in the following represents the Gaussian kernel, and K_{XY} represents the kernel between matrices X and Y, λ ensures invertible kernel matrix):

- Transform X^{tL} to $X^{new(L)}$ by a parameterized-location-scale shift (Fig. 5.1, left):

  $$X^{new(L)} = \mathbf{W}^{tL} \odot X^{tL} + \mathbf{B}^{tL},$$

  such that the support of $P(X^{new(L)})$ is contained in the support of $P(X^s)$. Here $\odot$ is the elementwise product to allow nonlinear transformations.
- Build a Gaussian Process on (X^s, Y^s) and predict on $X^{new(L)}$ to get $Y^{new(L)}$;
- Transform Y^{tL} to Y^* by a parameterized-location-scale shift (Fig. 5.1, right):

  $$Y^* = \mathbf{w}^{tL} \odot Y^{tL} + \mathbf{b}^{tL},$$

then we optimize the following empirical loss:

$$\arg \min_{\mathbf{W}^{tL},\mathbf{B}^{tL},\mathbf{w}^{tL},\mathbf{b}^{tL},\mathbf{w}^{t}} ||Y^{*} - Y^{new(L)}||^2 + \lambda_{reg}||\mathbf{w}^t - \mathbf{1}||^2, \tag{5.1}$$

where $\mathbf{W}^{tL}$, $\mathbf{B}^{tL}$ are matrices with the same size as X^{tL}, $\mathbf{w}^{tL}$, $\mathbf{b}^{tL}$ are vectors with the same size as Y^{tL}, and $\mathbf{w}^t$ is the scale vector on all Y^t. λ_{reg} is a regularization parameter.

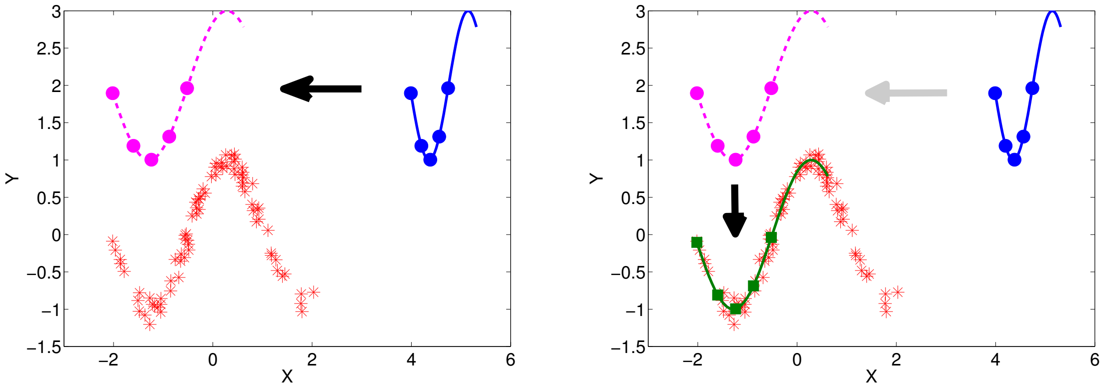


Figure 5.1: Illustration of the SMS method

Smoothness Constraint. To ensure the smoothness of the transformation w.r.t. X, we parameterize $\mathbf{W}^{tL}, \mathbf{B}^{tL}, \mathbf{w}^{tL}, \mathbf{b}^{tL}$ using:

$$\mathbf{W}^{tL} = R^{tL}\mathbf{G}, \mathbf{B}^{tL} = R^{tL}\mathbf{H}, \mathbf{w}^{tL} = R^{tL}\mathbf{g}, \mathbf{b}^{tL} = R^{tL}\mathbf{h},$$

where $R^{tL} = L^{tL}(L^{tL} + \lambda I)^{-1}, L^{tL} = K_{X^{tL}X^{tL}}$.

Following the same smoothness constraint we also have:

$$\mathbf{w}^t = R^t\mathbf{g},$$

where $R^t = K_{X^{te}X^{tL}}(L^{tL} + \lambda I)^{-1}$.

This parametrization results in the new objective function:

$$\arg \min_{G,H,g,h} ||(R^{tL}\mathbf{g} \odot Y^{tL} + R^{tL}\mathbf{h}) - Y^{new(L)}||^2 + \lambda_{reg}||R^t\mathbf{g} - \mathbf{1}||^2. \tag{5.2}$$

Note in the objective function, although we minimize the discrepancy between the transformed labels and the predicted labels for only the labeled points in the test domain, we put a regularization term on the transformation for all X^t to ensure an overall smoothness in the test domain.

5.2.2 Optimization Method

We use a Metropolis-Hasting algorithm to optimize the objective function (Eq. 5.2) which is multi-modal due to the use of the Gaussian kernel. The proposal distribution is given by

$$\theta^t \sim \mathcal{N}(\theta^{t-1}, \Sigma),$$

where Σ is a diagonal matrix with diagonal elements determined by the magnitude of $\theta \in \{\mathbf{G}, \mathbf{H}, \mathbf{g}, \mathbf{h}\}$.

In addition, the transformation on X requires that the support of $P(X^{new})$ is contained in the support of $P(X^s)$, which might be hard to achieve on real data, especially when X has a high-dimensional feature space. To ensure that the training data can be better utilized, we relax the support-containing condition by enforcing an overlapping ratio between the transformed X^{new} and X^s, i.e., we reject those proposal distributions which do not lead to a transformation that exceeds this ratio.

5.2.3 Recover the Prediction

After obtaining $\mathbf{G}, \mathbf{H}, \mathbf{g}, \mathbf{h}$, we make predictions on X^{tU} by:

- Transform X^{tU} to $X^{new(U)}$ with the optimized $\mathbf{G}, \mathbf{H}$:

$$X^{new(U)} = \mathbf{W}^{tU} \odot X^{tU} + \mathbf{B}^{tU} = R^{tU}\mathbf{G} \odot X^{tU} + R^{tU}\mathbf{H};$$

- Build a Gaussian Process on (X^s, Y^s) and predict on $X^{new(U)}$ to get $Y^{new(U)}$;
- Predict using optimized $\mathbf{g}, \mathbf{h}$:

$$\hat{Y}^{tU} = (Y^{new(U)} - \mathbf{b}^{tU})./\mathbf{w}^{tU} = (Y^{new(U)} - R^{tU}\mathbf{h})./R^{tU}\mathbf{g},$$

where $R^{tU} = K_{X^{tU}X^{tL}}(L^{tL} + \lambda I)^{-1}$.

With the use of $\mathbf{W} = R\mathbf{G}, \mathbf{B} = R\mathbf{H}, \mathbf{w} = R\mathbf{g}, \mathbf{b} = R\mathbf{h}$, we allow more flexible transformations than mean-centering and variance-scaling while assuming that the transformations are smooth w.r.t X. We will illustrate the advantage of the proposed method in the experimental section.

5.3 Active Learning

We consider two active learning goals and apply a myopic selection criteria to each:

- Active Learning which reduces total predictive covariance [35, 61]. An optimal myopic selection is achieved by choosing the point which minimizes the trace of the predictive covariance matrix conditioned on that selection.
- Active Surveying [22, 41, 72] which uses an estimation objective that is the sum of all the labels in the test set. An optimal myopic selection is achieved by choosing the point which minimizes the sum over all elements of the predictive covariance conditioned on that selection.

Now we derive the predictive covariance of the SMS approach. Note the transformation between $\hat{Y}^{tU}$ and $Y^{new(U)}$ is given by:

$$\hat{Y}^{tU} = (Y^{new(U)} - \mathbf{b}^{tU})./\mathbf{w}^{tU}.$$

Hence we have

$$Cov[\hat{Y}^{tU}] = \text{diag}\{1./\mathbf{w}^{tU}\} \cdot Cov(Y^{new(U)}) \cdot \text{diag}\{1./\mathbf{w}^{tU}\}.$$

As for $Y^{new(U)}$, since we build on Gaussian Processes for the prediction from $X^{new(U)}$ to $Y^{new(U)}$, it follows:

$$Y^{new(U)}|X^{new(U)} \sim \mathcal{N}(\mu, \Sigma),$$

where

$$\begin{aligned}\mu &= K_{X^{new(U)}X^s}(K_{X^{tr}X^s} + \lambda I)^{-1}Y^s,\\ \Sigma &= K_{X^{new(U)}X^{new(U)}} - K_{X^{new(U)}X^s}(K_{X^{tr}X^s} + \lambda I)^{-1}K_{X^{tr}X^{new(U)}}.\end{aligned}$$

Note the transformation between $X^{new(U)}$ and X^{tU} is given by: $X^{new(U)} = \mathbf{W}^{tU} \odot X^{tU} + \mathbf{B}^{tU}$. Integrating over $X^{new(U)}$, i.e.,

$$P(\hat{Y}^{new(U)}|X^{tU}, D) = \int_{X^{new(U)}} P(\hat{Y}^{new(U)}|X^{new(U)}, D)P(X^{new(U)}|X^{tU})dX^{new(U)},$$

with $D = \{X^s, Y^s, X^{tL}, Y^{tL}\}$.

Using the empirical form of $P(X^{new(U)}|X^{tU})$ which has probability $1/|X^{tU}|$ for each sample, we get:

$$Cov[\hat{Y}^{new(U)}|X^{tU}, X^s, Y^s, X^{tL}, Y^{tL}] = \Sigma.$$

Plugging the covariance of $Y^{new(U)}$ into $Cov[\hat{Y}^{tU}]$ we can get the final predictive covariance:

$$Cov(\hat{Y}^{tU}) = \text{diag}\{1./\mathbf{w}^{tU}\} \cdot \Sigma \cdot \text{diag}\{1./\mathbf{w}^{tU}\} \tag{5.3}$$

5.4 Experiments

5.4.1 Synthetic Dataset

Data Description

We generate the synthetic data with (using matlab notation):
$X^s = randn(80, 1); Y^s = sin(\alpha X^s + 1) + 0.1 * randn(80, 1);$
$X^t = [w * \min(X^s) + b : 0.03 : w * \max(X^s)/3 + b]; Y^t = \sin(\alpha(rev_w * X^t + rev_b) + 1) + 2;$
In words, X^s is drawn from a standard normal distribution, and Y^s is a sine function with Gaussian noise. X^t is drawn from a uniform distribution with a parameterized-location-scale transform on a subset of X^s. Y^t is the same sine function plus a constant offset of 2. w, b can be tuned to determine how X^t is transformed from X^s. rev_w, rev_b are the reverse transformation from X^t to X^s. α can be tuned to determine how complicated the function is. The synthetic dataset used is with $\alpha = 2; w = 0.5; b = 5; rev_w = 2; rev_b = -10$, as shown in Fig. 1.8.

Results

We compare the SMS approach with the following approaches:

- **Only test x**: prediction using labeled test data only;
- **Both x**: prediction using both training and labeled test data without transformation;
- **Offset**: the offset approach [73];
- **DM**: the distribution matching approach [73];
- **KMM**: Kernel mean matching [33];
- **T/C shift**: Target/Conditional shift [80], code is from `http://people.tuebingen.mpg.de/kzhang/Code-TarS.zip`.

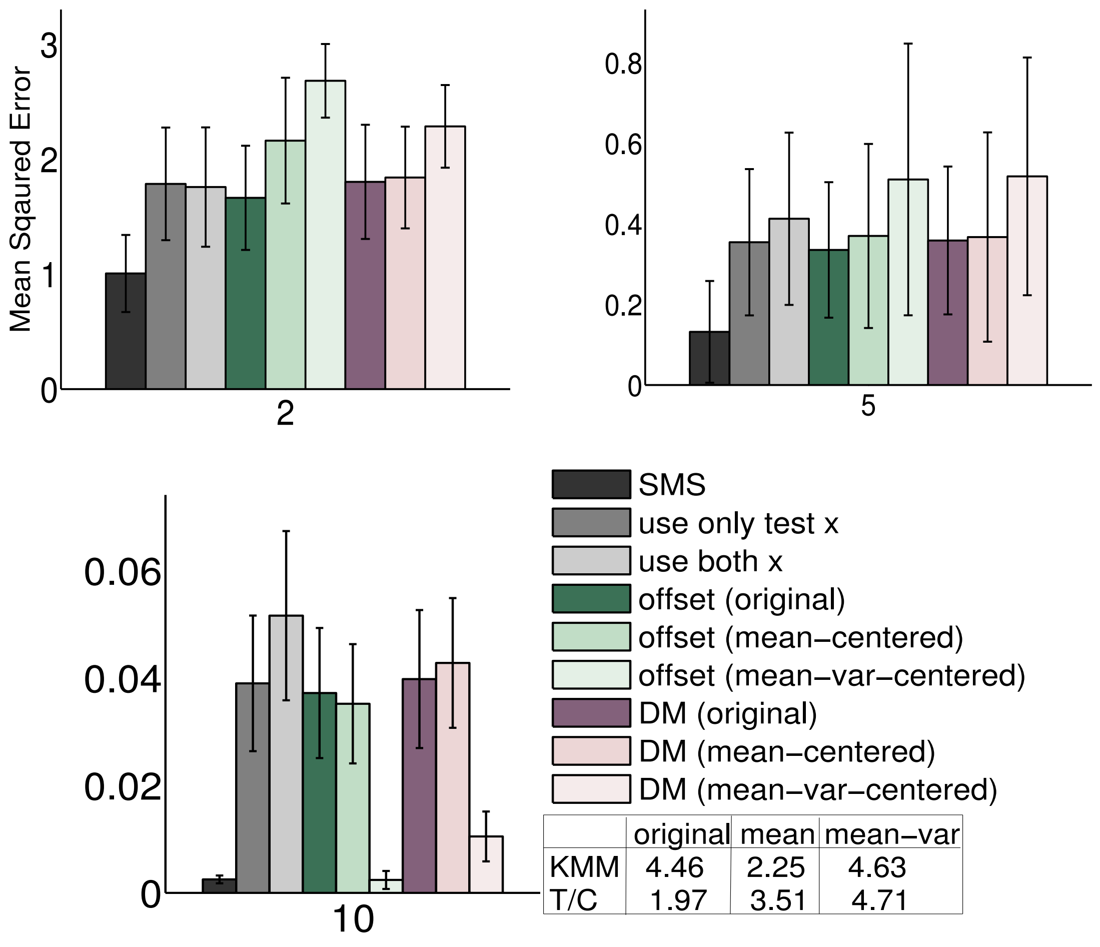
To ensure the fairness of comparison, we apply (3) to (6) using: the original data, the mean-centered data, and the mean-centered + variance-scaled data.

A detailed comparison with different number of observed test points are shown in Fig. 5.2, averaged over 10 experiments. The selection of which test points to label is done uniformly at random for each experiment. The parameters are chosen by cross-validation. Since KMM and Target/Conditional shift do not utilize the labeled test points, the Mean Squared Error of these two approaches are constants as shown in the text box. As we can see from the results, our proposed approach performs better than all other approaches.

As an example, the results for transfer learning with 5 labeled test points on the synthetic dataset are shown in Fig. 5.3. The 5 labeled test points are shown as filled blue circles. First, our proposed model, SMS, can successfully learn both the transformation on X and the transformation on Y, thus resulting in almost a perfect fit on unlabeled test points. Using only labeled test points results in a poor fit towards the right part of the function because there are no observed test labels in that part. Using both training and labeled test points results in a similar fit as using the labeled test points only, because the support of training and test domain do not overlap. The offset approach with mean-centered+variance-scaled data, also results in a poor fit because the training model is not true any more. It would have performed well if the variances are similar across domains. The support of the test data we generated, however, only consists of part of the support of the training data and hence simple variance-scaling does not yield a good match on $P(Y|X)$. The distribution matching approach suffers the same problem. The KMM approach, as mentioned before, applies the same conditional model $P(Y|X)$ across domains, hence it does not perform well. The Target/Conditional Shift approach does not perform well either since it does not utilize any of the labeled test points. Its predicted support of $P(Y^t)$, is constrained in the support of $P(Y^s)$, which results in a poor prediction of Y^t once there exists an offset between the Y's.

5.4.2 Real-world Dataset

The data and features are the same as described in section 3.5.2. The results for transfer learning are shown in Table 5.1. We compare the SMS approach with the same baselines as in the synthetic experiments. For $\{$DM, offset, KMM, T/C shift$\}$, we only show their best results after applying them on the original data, the mean-centered data, and the mean-centered+variance-scaled data. In each row the result in bold indicates the result with the best RMSE. The result



	original	mean	mean−var
KMM	4.46	2.25	4.63
T/C	1.97	3.51	4.71

Figure 5.2: Comparison of MSE on the synthetic dataset with $\{2, 5, 10\}$ labeled test points

with a star mark indicates that the best result is statistically significant at a $p = 0.05$ level with unpaired t-tests. From the results we can see that our proposed algorithm yields better results under most cases, especially when the number of labeled test points is small. This means our proposed algorithm can better utilize the source data and will be particularly useful in the early stage of learning model transfer, when only a small number of labels in the target domain is available/required.

The Active Learning/Active Surveying results are as shown in Fig. 5.4. We compare the **SMS** approach (covariance matrix in Eq. 5.3 for test point selection, and SMS for prediction) with:

- **combined+SMS**: combined covariance [73] for selection, and SMS for prediction;
- **random+SMS**: random selection, and SMS for prediction;
- **combined+offset**: the Active Learning/Surveying algorithm proposed in [73], using combined covariance for selection, and the corresponding offset approach for prediction.

From the results we can see that SMS is the best model overall. SMS is better than the Active Learning/Surveying approach proposed in [73] (combined+offset), especially in the Active

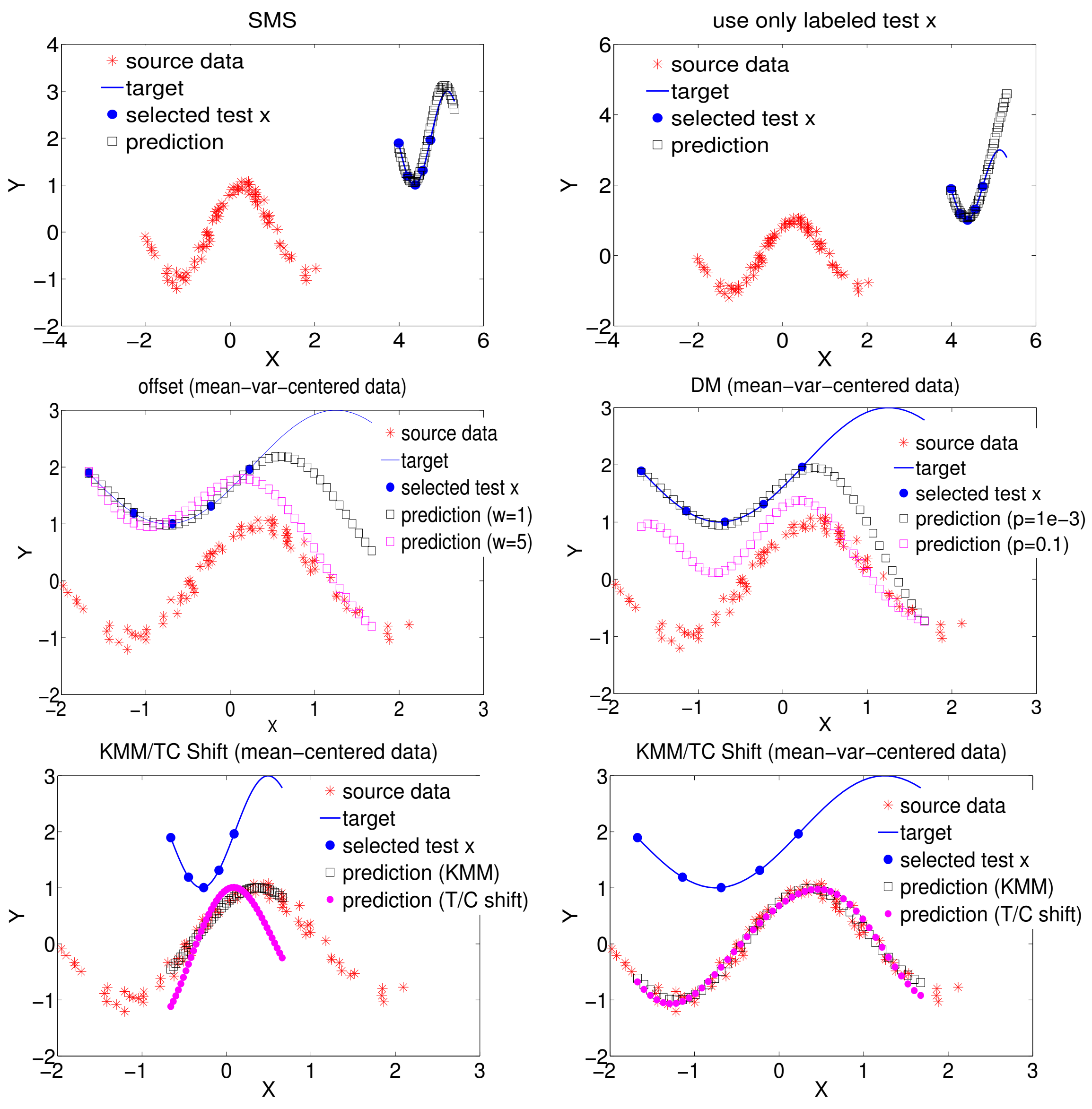


Figure 5.3: Comparison of results on the synthetic dataset: An example

Surveying result. Moreover, the combined+SMS result is better than combined+offset, which also indicates that the SMS model is better for prediction than the offset approach in [73]. Also, given the better model that SMS has, there is not much difference in which active learning algorithm we use. However, SMS with active selection is better than SMS with random selection, especially in the Active Learning result.

Table 5.1: RMSE for transfer learning on real data

# X^{tL}	SMS	DM	Offset	Only test x	Both x	KMM	T/C Shift
5	**1197±23***	1359±54	1303±39	1479±69	2094±60	2127	2330
10	**1046±35***	1196±59	1234±53	1323±91	1939±41	2127	2330
15	**993±28**	1055±27	1063±30	1104±46	1916±36	2127	2330
20	**985±13**	1056±54	1024±20	1086±74	1832±46	2127	2330
25	**982±14**	1030±29	1040 ±27	1039±31	1839±41	2127	2330
30	960±19	**921±29**	961±30	937±29	1663±31	2127	2330
40	**890±26**	898±30	938±30	901±31	1621±34	2127	2330
50	**893±16**	925±59	935±59	926±64	1558±51	2127	2330
70	860±40	805±38	819±40	**804±37**	1399±63	2127	2330
90	**791±98**	838±102	863±99	838±104	1288±117	2127	2330

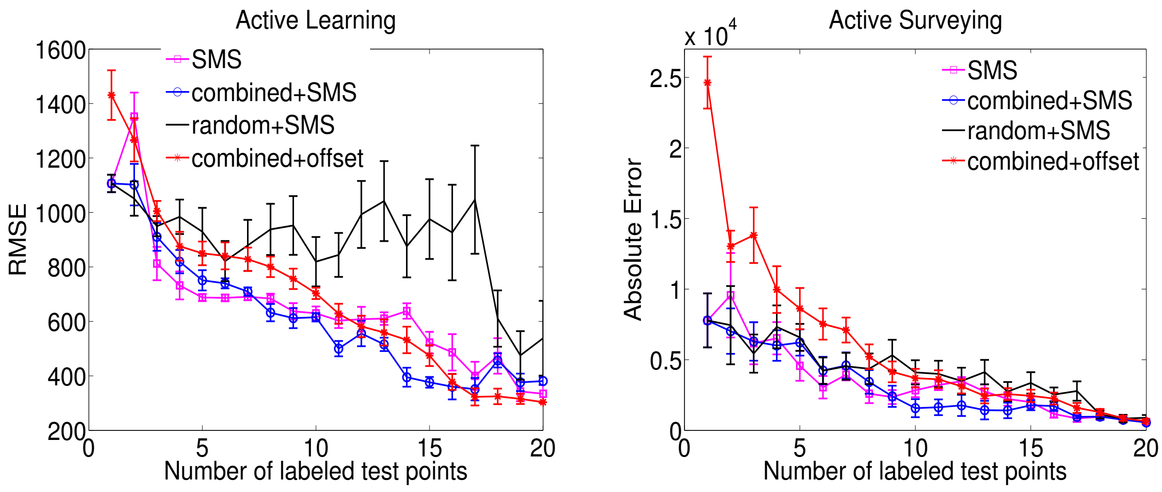


Figure 5.4: Active Learning/Surveying results on the real dataset (legend: selection+prediction).

5.5 Conclusion

In this chapter we proposed transfer learning algorithms under a more general case where both the support and the model change across domains. We show that by allowing smooth transformations for both features and labels across domains we are able to obtain better results than simple mean-centering and variance scaling, which is a common technique people use today. In addition we proposed active learning algorithms under this transfer learning framework, and we show that the performance is further improved after a few queries on a real-world grape yield prediction application.

Chapter 6

Nonparametric Stability Analysis for Multi-Task Learning Problems

6.1 Overview

A closely related problem of transfer learning is multi-task learning, where multiple domains are coupled together in the learning process to improve the overall performance. Similar to transfer learning where smoothness relationship between the source domain and target domain is enforced, multi-task learning imposes some kind of smooth relationship among the tasks, e.g., by assuming the function for each task consists of a shared central function and an independent offset function, or by assuming pairwise smoothness between each pair of tasks.

In this chapter, we propose a general framework for a family of multi-task learning algorithms that use kernel ridge regression, by placing a reweighting matrix on the task weights to capture the relationship among tasks. We study the stability bounds under this framework and show that the stability bounds mainly depend on the diagonal blocks of the inversed reweighting matrix. This analysis provide insight on how much we can gain from regularizing the relationship among tasks under different smoothness assumptions.

6.2 Stability Analysis on Multi-Task Kernel Ridge Regression

In this section, we formulate the problem of multi-task kernel ridge regression (MT-KRR) and then we analyze the stability bounds for the MT-KRR algorithm. Our analysis shows that, MT-KRR achieves tighter stability bounds than independent task learning by regularizing task relations. In addition, different regularization techniques yield different stability bounds that are closely related to the diagonal blocks of the inversed reweighting matrix.

Remark. For multi-task learning, the usual assumption is that we have T moderately labeled tasks (i.e., comparable number of samples for each task). Hence this section focuses on how the risk bound is affected by the reweighting matrix on task weights (i.e., different smoothness parameters for each task and among tasks), which is less affected by the number of samples for each task. This is different from the analysis for transfer learning, where the usual assumption is that we have a sufficiently labeled source task and a very limited labeled target task, and the risk bound is affected by the smoothness parameter for the offset, the sample size for the source task and the sample size for the target task.

6.2.1 Multi-task KRR Algorithm: Formulation and Objective

Notations. Assume we have T tasks, each task t has data matrix $X_t \in \mathcal{R}^{n_t \times d}, Y_t \in \mathcal{R}^{n_t}$, where $x_{t,i} \in \mathcal{X}$ is the i-th row of X_t, and $y_{t,i} \in \mathcal{Y}$ is the i-th scalar of Y_t. n_t is the number of data points for each task, and d is the dimension of features. Denote the total number of data points as $m = \sum_{t=1}^{T} n_t$.

Let ϕ be the feature mapping on x associated to kernel k with dimension q, and $\Phi(X_t)$ denote the matrix in $\mathcal{R}^{n_t \times q}$ whose rows are the vectors $\phi(x_{t,i})$. Let $\Phi(X) \in \mathcal{R}^{m \times Tq}$ represent the diagonalized data matrix $\Phi(X) = \text{diag}[\Phi(X_1)\ \Phi(X_2)\ \cdots\ \Phi(X_T)]$ for all tasks as following, $Y \in \mathcal{R}^{m \times 1}$ be the stacked label vector, and $w \in \mathcal{R}^{Tq \times 1}$ be the stacked weight vector for all tasks:

$$\Phi(X) = \begin{pmatrix} \Phi(X_1) & 0 & \cdots & 0 \\ 0 & \Phi(X_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Phi(X_T) \end{pmatrix},$$

$$Y = [Y_1\, Y_2\, \ldots\, Y_T]^\top, w = [w_1\, w_2\, \ldots\, w_T]^\top.$$

Throughout the section we use ℓ_2 loss as the loss function for a hypothesis h, i.e., $l(h(x), y) = (h(x) - y)^2$. Note that $l(h(x), y)$ is a σ-admissible loss function, i.e.,

$$\forall x, y, \forall h, h', |l(h(x), y) - l(h'(x), y)| \leq \sigma |h(x) - h'(x)|.$$

For ℓ_2 loss $\sigma = 4B$, assuming $|h(x)| \leq B, |y| \leq B$ for some $B > 0$.

Objective. Define the MT-KRR objective as:

$$\min_w \frac{1}{m} ||Y - \Phi(X)w||_F^2 + w^\top \Lambda w,$$

where Λ is a $Tq \times Tq$ reweighting matrix on task weights w.

Let $\tilde{\phi}(x_{t,j}) = [0\ \cdots\ 0\ \phi(x_{t,j})\ 0\ \cdots\ 0]$ be a row of $\Phi(X)$ for task t. Let $\mathcal{H}$ be a reproducing kernel Hilbert space with kernel

$$k_{\Lambda^{-1}}(x_{s,i}, x_{t,j}) = \tilde{\phi}(x_{s,i}) \Lambda^{-1} \tilde{\phi}^\top(x_{t,j})$$

where s, t are indices for tasks, the objective becomes:

$$\min_{g \in \mathcal{H}} \frac{1}{m} \sum_{t=1}^{T} \sum_{j=1}^{n_t} (y_{t,j} - g(x_{t,j}))^2 + ||g||_{k_{\Lambda^{-1}}}^2 \tag{6.1}$$

where $g(x) = \langle g, k_{\Lambda^{-1}}(x, .) \rangle_{\mathcal{H}}$, and $||.||_{K_{\Lambda^{-1}}}$ is the norm in $\mathcal{H}$. This generalizes to the case where $q = \infty$. The solution to MT-KRR is (assuming nonsingular Λ):

$$w = \Lambda^{-1} \Phi^\top(X) [\Phi(X) \Lambda^{-1} \Phi^\top(X) + mI]^{-1} Y.$$

Note in multi-task learning setting, we assume $\Lambda = \Omega \otimes \mathbf{I_q}$ (for some $\Omega \in \mathcal{R}^{T\times T}$), where $\mathbf{I_q}$ is the $q \times q$ identity matrix and $\otimes$ is the Kronecker product. By the property of the inverse of a Kronecker product, $\Lambda^{-1} = M \otimes \mathbf{I_q}$ where $M = \Omega^{-1}$, and it can be easily shown that $k_{\Lambda^{-1}}(x_{s,i}, x_{t,j}) = M_{s,t}k(x_{s,i}, x_{t,j})$.

Remark. Eq. 6.1 assumes same weight $1/m$ on the loss for $(x_{t,j}, y_{t,j})$ for all tasks. Alternatively, we can put different weights on the loss for different tasks, i.e,

$$\min_w \sum_{t=1}^{T} \frac{1}{n_t} \sum_{j=1}^{n_t} (\phi(x_{t,j})w_t - y_{t,j})^2 + w^\top \Lambda w.$$

The solution becomes

$$w = \Lambda^{-1}\Phi^\top(X)(\Phi(X)\Lambda^{-1}\Phi^\top(X) + C^{-1}I)^{-1}Y,$$

where C is the loss-reweighting matrix with $1/n_t$'s as the diagonal elements. As C is the same under different Λ's, it is not the focus of this section. A study on the effect of C can be found in [15].

6.2.2 Examples

Much existing multi-task algorithms can be cast into the above framework, with different Λ's for the penalty. In the following, we show a few examples (see Table 6.1 for a summary), where $P = w^\top \Lambda w$ is the penalty term.

Independent tasks

If we consider each task independently, the algorithm is the same as regular KRR. Hence

$$P = \lambda \sum_{t=1}^{T} ||w_t||^2, \text{ and } \Lambda = \lambda \mathbf{I_T} \otimes \mathbf{I_q}.$$

Hence for $\Lambda = \Omega \otimes \mathbf{I_q}$, we have

$$\Omega = \begin{pmatrix} \lambda_s & 0 & \cdots & 0 \\ 0 & \lambda_s & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_s \end{pmatrix}.$$

Central/mean function+offset

This algorithm is described in [19]. The basic assumption is that, for each task the hypothesis is given by $h(x_t) = x_t(w_c + w_t)$, and

$$P = \lambda_c ||w_c||^2 + \lambda_o \sum_{t=1}^{T} ||w_t||^2,$$

where λ_c is regularizing the central/mean function, and λ_o is regularizing the offset function. In the same paper the authors proved an equivalent version of the penalty term:

$$P = \lambda_s \sum_{t=1}^{T} ||w_t||^2 + \lambda_p \sum_{t=1}^{T} ||w_t - \frac{1}{T} \sum_{s=1}^{T} w_s||^2,$$

hence for $\Lambda = \Omega \otimes \mathbf{I_q}$, we have

$$\Omega = \begin{pmatrix} \lambda_s + \lambda_p(1 - \frac{1}{T}) & -\frac{\lambda_p}{T} & \cdots & -\frac{\lambda_p}{T} \\ -\frac{\lambda_p}{T} & \lambda_s + \lambda_p(1 - \frac{1}{T}) & \cdots & -\frac{\lambda_p}{T} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{\lambda_p}{T} & -\frac{\lambda_p}{T} & \cdots & \lambda_s + \lambda_p(1 - \frac{1}{T}) \end{pmatrix}.$$

Pairwise penalty

This algorithm is mentioned in [66]. Let λ_p be the regularization parameter for the pairwise penalty and λ_s be the regularization parameter for each single task.

$$P = \lambda_p \sum_{s,t \in \{1 \ldots T\}, s \neq t} ||w_s - w_t||^2 + \lambda_s \sum_{t=1}^{T} ||w_t||^2,$$

for $\Lambda = \Omega \otimes \mathbf{I_q}$, we have

$$\Omega = \begin{pmatrix} \lambda_p(T-1) + \lambda_s & -\lambda_p & \cdots & -\lambda_p \\ -\lambda_p & \lambda_p(T-1) + \lambda_s & \cdots & -\lambda_p \\ \vdots & \vdots & \ddots & \vdots \\ -\lambda_p & -\lambda_p & \cdots & \lambda_p(T-1) + \lambda_s \end{pmatrix}.$$

Remark. Note the equivalence of regularizing using a pairwise penalty and using central+offset, if we set λ_p/T in central+offset to be λ_p in pairwise penalty.

Temporal penalty

The algorithm is described in [84], where there is a penalty on parameters for each pair of consecutive tasks, and an extra penalty on the parameter for each task.

$$P = \lambda_p \sum_{t=1,\ldots,T-1} ||w_t - w_{t+1}||^2 + \lambda_s \sum_{t=1}^{T} ||w_t||^2,$$

for $\Lambda = \Omega \otimes \mathbf{I_q}$, we have

$$\Omega = \begin{pmatrix} \lambda_p + \lambda_s & -\lambda_p & 0 & \cdots & 0 \\ -\lambda_p & 2\lambda_p + \lambda_s & -\lambda_p & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \vdots & -\lambda_p & 2\lambda_p + \lambda_s & -\lambda_p \\ 0 & 0 & \cdots & -\lambda_p & \lambda_p + \lambda_s \end{pmatrix}.$$

Methods	Penalty $P = w^\top \Lambda w$	$\Lambda = \Omega \otimes \mathbf{I_q}$
Independent tasks	$\lambda_s \sum_{t=1}^{T} \lvert\lvert w_t \rvert\rvert^2$	$\Omega = \lambda_s \mathbf{I_T}$
Central+offset [19]	$\lambda_s \sum_{t=1}^{T} \lvert\lvert w_t \rvert\rvert^2 + \lambda_p \sum_{t=1}^{T} \lvert\lvert w_t - \frac{1}{T}\sum_{s=1}^{T} w_s \rvert\rvert^2$	$\begin{cases} \Omega_{t,t} = \lambda_s + \lambda_p(1-\frac{1}{T}) \\ \Omega_{s,t} = -\frac{\lambda_p}{T}, s \neq t \end{cases}$
Pairwise [66]	$\lambda_p \sum_{s \neq t} \lvert\lvert w_s - w_t \rvert\rvert^2 + \lambda_s \sum_{t=1}^{T} \lvert\lvert w_t \rvert\rvert^2$	$\begin{cases} \Omega_{t,t} = \lambda_p(T-1) + \lambda_s \\ \Omega_{s,t} = -\lambda_p, s \neq t \end{cases}$
Temporal [84]	$\lambda_p \sum_{t=1}^{T-1} \lvert\lvert w_t - w_{t+1} \rvert\rvert^2 + \lambda_s \sum_{t=1}^{T} \lvert\lvert w_t \rvert\rvert^2$	$\begin{cases} \Omega_{t,t} = 2\lambda_p + \lambda_s, t = 2, \dots, T-1; \\ \Omega_{t,t+1} = \Omega_{t+1,t} = -\lambda_p, t = 1, \dots, T-1; \\ \Omega_{1,1} = \Omega_{T,T} = \lambda_p + \lambda_s;\ \text{zero otherwise.} \end{cases}$

Table 6.1: Examples of multi-task learning algorithms with different Λ's as penalty

6.2.3 Uniform Stability for MT-KRR

We study the uniform stability [11], which is usually used to bound true risk in terms of empirical risk, for the MT-KRR algorithm.

Definition 6.2.1. ([11]). *The uniform stability β for an algorithm A w.r.t. the loss function l is defined as: $\forall S \in \mathcal{Z}^m, \forall i \in \{1, ..., m\}$,*

$$||l(A_S, .) - l(A_{S\setminus i}, .)||_\infty \leq \beta,$$

where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ drawn i.i.d from an unknown distribution D, and $S \setminus i$ is formed by removing the i-th element.

Definition 6.2.2. (Uniform stability w.r.t a task t). *Let i be a data index for task t. The uniform stability β_t of a learning algorithm A w.r.t a task t, w.r.t. loss l is: $\forall S \in \mathcal{Z}^m, \forall i \in \{1, ..., n_t\}$,*

$$||l(A_S, .) - l(A_{S\setminus i}, .)||_\infty \leq \beta_t.$$

Let the risk or generalization error be defined as $R(A, S) = \mathbb{E}_z[l(A_S, z)], z \in \mathcal{Z}$, and the empirical error be defined as $R_{emp} = \frac{1}{m}\sum_{i=1}^{m} l(A_S, z_i), z_i \in \mathcal{Z}^m$. Then we have the following generalization error bound (Theorem 12, [11]) with probability at least $1 - \delta$:

$$R \leq R_{emp} + 2\beta + (4m\beta + 4B^2)\sqrt{\frac{\ln 1/\delta}{2m}}.$$

This theorem gives tight bounds when the stability β scales as $1/m$. For the MT-KRR algorithm, we have the following theorem hold with respect to the uniform stability:

Theorem 6.2.3. *Denote $\Lambda^{-1} = M \otimes \mathbf{I_q}$, and $M_1, \dots, M_T$ are the diagonal elements of M. Assuming the kernel values are bounded: $\forall x \in \mathcal{X}, k(x, x) \leq \kappa^2 < \infty$. The learning algorithm defined by the minimizer of Eq. 6.1 has uniform stability β w.r.t. σ-admissible loss l with:*

$$\beta \leq \frac{\sigma^2 \kappa^2}{2m} \max_t M_t.$$

Proof. Denote $h \in \mathcal{H}$ as the minimizer of Eq. 6.1, i.e.,

$$F(g) = \frac{1}{m}\sum_{s=1}^{T}\sum_{j=1}^{n_s}(y_{s,j} - g(x_{s,j}))^2 + ||g||^2_{k_{\Lambda^{-1}}} = R(g) + N(g),$$

and $h' \in \mathcal{H}$ as the minimizer of Eq. 6.1 on the same data except with the element $x_{t,i}, y_{t,i}$ removed:

$$F'(g) = \frac{1}{m}\sum_{s=1}^{T}\sum_{j\neq i \text{ if } s=t}(y_{s,j} - g(x_{s,j}))^2 + ||g||_{k_{\Lambda^{-1}}}^2 = R'(g) + N(g).$$

Define the Bregman Divergence associated to F of f to g by

$$B_F(f||g) = F(f) - F(g) - < f - g, \nabla F(g) > .$$

Since h, h' are the minimizers, we have

$$\begin{aligned} B_{F'}(h||h') + B_F(h'||h) &= F'(h) - F'(h') + F(h') - F(h) \\ &= R'(h) - R'(h') + R(h') - R(h) \end{aligned} \tag{6.2}$$

Using the nonnegativity of B and $B_F = B_R + B_N$, $B_{F'} = B_{R'} + B_N$, we have

$$B_N(h||h') + B_N(h'||h) \leq B_F(h'||h) + B_{F'}(h||h') \tag{6.3}$$

Combining Eq. 6.2 and Eq. 6.3 we have

$$\begin{aligned} &B_N(h||h') + B_N(h'||h) \\ &\leq R'(h) - R'(h') + R(h') - R(h) \\ &= \frac{1}{m}[\sum_{s=1}^{T}\sum_{j\neq i \text{ if } s=t} l(h(x_{s,j}), y_{s,j}) - \sum_{s=1}^{T}\sum_{j\neq i \text{ if } s=t} l(h'(x_{s,j}), y_{s,j}) \\ &\quad + \sum_{s=1}^{T}\sum_{j=1}^{n_s} l(h'(x_{s,j}), y_{s,j}) - \sum_{s=1}^{T}\sum_{j=1}^{n_s} l(h(x_{s,j}), y_{s,j})] \\ &= \frac{1}{m}[l(h'(x_{t,i}), y_{t,i}) - l(h(x_{t,i}), y_{t,i})] \\ &\leq \frac{1}{m}\sigma||\Delta h||_{k_{\Lambda^{-1}}}\sqrt{\tilde{\phi}(x_{t,i})\Lambda^{-1}\tilde{\phi}^\top(x_{t,i})}, \end{aligned} \tag{6.4}$$

using the fact that l is a σ-admissible loss function.
Since for RKHS norm $B_N(f||g) = ||f - g||_{k_{\Lambda^{-1}}}^2$, we have $B_N(h||h') + B_N(h'||h) = 2||h - h'||_{k_{\Lambda^{-1}}}^2$. Using Eq. 6.4, we have

$$\begin{aligned} 2||h - h'||_{k_{\Lambda^{-1}}}^2 &= 2||\Delta h||_{k_{\Lambda^{-1}}}^2 \\ &\leq \frac{1}{m}\sigma||\Delta h||_{k_{\Lambda^{-1}}}\sqrt{\tilde{\phi}(x_{t,i})\Lambda^{-1}\tilde{\phi}^\top(x_{t,i})}. \end{aligned}$$

We get

$$||\Delta h||_{k_{\Lambda^{-1}}} = ||h - h'||_{k_{\Lambda^{-1}}} \leq \frac{\sigma\sqrt{\tilde{\phi}(x_{t,i})\Lambda^{-1}\tilde{\phi}^\top(x_{t,i})}}{2m}. \tag{6.5}$$

Note $\tilde{\phi}(x_{t,i})$ is in the form of $[0 \ \cdots \ 0 \ \phi(x_{t,i}) \ 0 \ \cdots \ 0]$. By the standard bounds for Rayleigh quotient, we have

$$\tilde{\phi}(x_{t,i})\Lambda^{-1}\tilde{\phi}^\top(x_{t,i}) \leq \kappa^2\lambda_{max}(M_t\mathbf{I_q}) = \kappa^2 M_t. \tag{6.6}$$

Combining Eq. 6.5 and Eq. 6.6, and generalizing to $\forall t$ we have

$$||\Delta h||_{k_{\Lambda^{-1}}} \leq \frac{\sigma\sqrt{\kappa^2 \max_t M_t}}{2m} \tag{6.7}$$

Now we obtain the uniform stability by $\forall x, y$,

$$\begin{aligned}|l(h(x),y) - l(h'(x),y)| &\leq \sigma||\Delta h||_{k_{\Lambda^{-1}}}[\tilde{\phi}(x)\Lambda^{-1}\tilde{\phi}^\top(x)] \\ &\leq \sigma||\Delta h||_{k_{\Lambda^{-1}}}\sqrt{\kappa^2 \max_t M_t} \\ &\leq \frac{\sigma^2\kappa^2}{2m}\max_t M_t,\end{aligned}$$

using Eq. 6.6 and Eq. 6.7. □

Remark. The above theorem provides a more direct stability bound by taking the maximum over the diagonal elements of M, instead of computing the largest eigenvalue as in [82]. Also, for a specific task t, if $M_t < \max_s\{M_s\}$, then it is possible to obtain tighter stability β_t using only M_t, which yields tighter bounds than the one in [82] where they consider the worst case for all tasks.

Lemma 6.2.4. *The learning algorithm defined by the minimizer of Eq. 6.1 has uniform stability β_t w.r.t a task t, w.r.t. σ-admissible loss l with:*

$$\beta_t \leq \frac{\sigma^2\kappa^2}{2n_t}M_t.$$

The proof is a straightforward adaptation of proof of Thm 6.2.3, with x constrained to be $x_{t,i}$. The reason we care about β_t is that, it leads to a tighter generalization error bound for a task t, given $\beta_t < \beta$. We have, with probability at least $1-\delta$,

$$R^t \leq R^t_{emp} + 2\beta_t + (4n_t\beta_t + 4B^2)\sqrt{\frac{\ln 1/\delta}{2n_t}},$$

where $R^t = \mathbb{E}_z[l(A_S(x_{t,i}), y_{t,i})]$, $R^t_{emp} = \frac{1}{n_t}\sum_{i=1}^{n_t} l(A_S(x_{t,i}), y_{t,i})$.
In the following section, we study the stability bounds under a few special cases (Table 6.1), where it can be shown that we have tighter stability bounds for MT-KRR than learning each task independently.

6.2.4 Stability Bounds under Different Penalties

(a) Independent tasks. It is easy to derive that $\forall t, M_t = 1/\lambda_s$, and

$$\beta_{ind} \leq \frac{\sigma^2\kappa^2}{2\lambda_s m}.$$

Remark. In [5], the stability of multi-task KRR is analyzed by considering each task separately, which corresponds to the above analysis. In the following, we will show that different regularizations on task relations help tighten the stability bounds of MTL algorithms.

(b) Central function+offset. Applying blockwise matrix inversion we have

$$\forall t, M_t = \frac{\lambda_p/T + \lambda_s}{\lambda_s(\lambda_p + \lambda_s)}.$$

Hence we achieve tighter stability bounds than β_{ind} for $T \geq 2$ and $\lambda_p > 0$:

$$\max_t M_t = \frac{\lambda_p/T + \lambda_s}{\lambda_s(\lambda_p + \lambda_s)} < \frac{1}{\lambda_s}. \tag{6.8}$$

(c) Pairwise penalty. Similarly to (b), we can derive that

$$M_t = \frac{\lambda_p + \lambda_s}{\lambda_s(\lambda_p T + \lambda_s)}.$$

For $T \geq 2$ and $\lambda_p > 0$, again we obtain tighter bounds than β_{ind}:

$$\max_t M_t = \frac{\lambda_p + \lambda_s}{\lambda_s(\lambda_p T + \lambda_s)} < \frac{1}{\lambda_s}. \tag{6.9}$$

(d) Temporal penalty. We have the following lemma:

Lemma 6.2.5. *Let Λ be defined as in Table 6.1 under temporal penalty and M be defined as in theorem 6.2.3. Let $M_{t_{mid}}$ be the middle element(s) of $M_1, M_2, \ldots, M_T$, i.e., $t_{mid} = T/2, T/2+1$ if T is even, and $t_{mid} = (T+1)/2$ if T is odd. Then the following hold:*

$$\begin{aligned} &M_t < M_{t-1}, t = 2, \ldots, t_{mid}; M_t < M_{t+1}, t = t_{mid}, \ldots, T; \\ &\max_t M_t = M_1 = M_T < \frac{1}{\lambda_s}; \min_t M_t = M_{t_{mid}} \geq \frac{\lambda_p + \lambda_s}{\lambda_s(\lambda_p T + \lambda_s)}. \end{aligned}$$

Proof. Using blockwise matrix inversion we can obtain the following recurrence equations for the diagonal elements of $M = \Lambda^{-1}$:

$$\begin{aligned} &M_1 = 1/\lambda_1, \\ &M_t = 1/\lambda_t + \lambda_p^2/\lambda_t^2 M_{t-1}, t = 2, \ldots, T \end{aligned} \tag{6.10}$$

where

$$\begin{cases} \lambda_t = (2\lambda_p + \lambda_s) - \lambda_p^2/\lambda_{t+1} & t = 2, \ldots, T-1 \\ \lambda_t = (\lambda_p + \lambda_s) - \lambda_p^2/\lambda_{t+1} & t = 1 \end{cases}$$

with initial condition $\lambda_T = \lambda_p + \lambda_s$.
First using induction we have $\lambda_t \geq \lambda_p + \lambda_s, t = 2, \ldots, T$ and $\lambda_t > \lambda_{t+1}, t = 2, \ldots, T-1$. Hence

$$\lambda_t - \lambda_{t+1} \leq \frac{\lambda_p \lambda_s}{\lambda_p + \lambda_s}$$

Applying the initial condition we get

$$\lambda_t \leq (T-t)\frac{\lambda_p \lambda_s}{\lambda_p + \lambda_s} + (\lambda_p + \lambda_s), t \geq 2 \tag{6.11}$$

Second, it can be shown that $M_t = M_{T+1-t}, t = 1, \ldots, T$ by applying blockwise inversion Lemma in two ways, one is starting from the top-left element of the matrix, another is starting from the right-bottom element of the matrix.
Third, rewriting the recurrence equation for M_t we get

$$M_t - \frac{\lambda_t}{\lambda_t^2 - \lambda_p^2} = \frac{\lambda_p^2}{\lambda_t^2}(M_{t-1} - \frac{\lambda_t}{\lambda_t^2 - \lambda_p^2})$$

Since $\lambda_t \geq \lambda_p + \lambda_s$, we have

$$\begin{cases} M_t < M_{t-1}, & \text{if } M_t > \frac{\lambda_t}{\lambda_t^2 - \lambda_p^2} \\ M_t > M_{t-1}, & \text{if } M_t < \frac{\lambda_t}{\lambda_t^2 - \lambda_p^2} \end{cases}$$

It is easy to see that $\frac{\lambda_t}{\lambda_t^2 - \lambda_p^2} = \frac{1}{\lambda_t + \lambda_{t-1} - (2\lambda_p + \lambda_s)}$ is a monotonically increasing sequence. Hence we have $M_t < M_{t-1}$ for $t = 2, \ldots, t_{mid}$, and $M_t < M_{t+1}$ for $t = t_{mid}, \ldots, T$, where t_{mid} is(are) the middle element(s) of $1, \ldots, T$.
Now it is easy to see the $\min_t M_t = M_{t_{mid}}, \max_t M_t = M_1 = M_T$.
If T is even, then $t_{mid} = T/2$ or $T/2 + 1$, and $M_{T/2} = M_{T/2+1}$. We have the following equation hold:

$$1/\lambda_{T/2} + \lambda_p^2/\lambda_{T/2}^2 M_{T/2} = M_{T/2+1} = M_{T/2}$$

We can solve for $M_{T/2}$ and get

$$M_{T/2} = \frac{1}{\lambda_{T/2+1} - \lambda_p^2/\lambda_{T/2+1}^2} = \frac{1}{\lambda_{T/2+1} + \lambda_{T/2} - (2\lambda_p + \lambda_s)} \geq \frac{\lambda_p + \lambda_s}{T\lambda_p\lambda_s + \lambda_s^2}$$

by applying inequality 6.11.
If T is odd, then $t_{mid} = (T+1)/2$, and $M_{(T-1)/2} = M_{(T+3)/2}$. Again we can solve for $M_{(T+1)/2}$ and get

$$M_{(T+1)/2} = \frac{1}{2\lambda_t - (2\lambda_p + \lambda_s)} \geq \frac{\lambda_p + \lambda_s}{T\lambda_p\lambda_s + \lambda_s^2}$$

again by applying inequality 6.11. Moreover, using induction it can be shown that when $T = 3$, the bound is tight, i.e., $M_{t_{mid}} = \frac{\lambda_p + \lambda_s}{T\lambda_p\lambda_s + \lambda_s^2}$; and for $T > 3$, $M_{t_{mid}} > \frac{\lambda_p + \lambda_s}{T\lambda_p\lambda_s + \lambda_s^2}$. Finally,

$$M_1 = M_T = 1/\lambda_1 \leq \frac{\lambda_p + \lambda_s}{\lambda_s(2\lambda_p + \lambda_s)} < \frac{1}{\lambda_s}.$$

□

Combining Lemma 6.2.5 and Lemma 6.2.4 we can see that, with temporal penalty we have tightest stability bounds β_t for $t = t_{mid}$. Also, we achieve tighter stability bounds β_t for the tth task than the $t-1$th task, if $t < t_{mid}$; and tighter β_t for the tth task than the $t+1$th task, if $t > t_{mid}$. However, since

$$M_{t_{mid}} \geq (\lambda_p + \lambda_s)/(\lambda_s(\lambda_p T + \lambda_s)),$$

we achieve a looser bound with temporal penalty than Eq. 6.8 or Eq. 6.9. It indicates that we might lose some algorithmic stability due to the relatively restricted temporal smoothness assumption, compared to assuming pairwise smoothness. Nonetheless, the stability bound with temporal penalty is tighter than learning each task independently: $\max_t M_t < 1/\lambda_s$, for $T \geq 2$.

6.3 Extension to Classification

For a multi-task classification problem, we are given T tasks, each task t has data matrix X_t with dimension $n_t \times d$, and Y_t with dimension $n_t \times 1$, where $x_{t,i} \in \mathcal{R}$ is the i-th row of X_t, and $y_{t,i} \in \{0, 1\}$ is the i-th entry of Y_t.

To extend the previous multi-task regression algorithm to classification, we assume a logistic function from a latent continuous variable $z_{t,i} \in \mathcal{R}$ to the binary label $y_{t,i} \in \{0, 1\}$:

$$p(y_{t,i}|z_{t,i}) = \frac{1}{1 + e^{-z_{t,i}y_{t,i}}}$$

Let Z_t be an $n_t \times 1$ vector with ith entry being $z_{t,i}$, and Z be the stacked vector:

$$Z = [Z_1\ Z_2\ \ldots\ Z_T]^\top.$$

The goal is to maximize the likelihood with respect to $\tilde{X}$, which is the transformed X under the multi-task assumption with task correlation matrix Λ:

$$\begin{aligned} p(Z|Y, \tilde{X}) &\propto p(Z|\tilde{X})P(Y|Z) \\ &\propto p(Z|\tilde{X}) \prod_{t,i} p(y_{t,i}|z_{t,i}) \\ &\propto \exp\{-\frac{1}{2} Z^\top K^{-1}_{X\Lambda^{-1}X} Z\} \prod_{t,i} \frac{1}{1 + e^{-z_{t,i}y_{t,i}}}, \end{aligned}$$

where $K_{X\Lambda^{-1}X} = \Phi(X)\Lambda^{-1}\Phi^\top(X)$.
Equivalently we maximize:

$$\log p(Z|Y, X) = -\frac{1}{2} Z^\top K^{-1}_{X\Lambda^{-1}X} Z - \sum_{t,i} \log(1 + e^{-z_{t,i}y_{t,i}}) + C,$$

where C is some constant. Hence the objective is:

$$\min_Z \frac{1}{2} Z^\top K^{-1}_{X\Lambda^{-1}X} Z + \sum_{t,i} \log(1 + e^{-z_{t,i}y_{t,i}}).$$

After optimization we obtain Z, which is continuous. For a new data matrix X^*, since

$$\begin{bmatrix} Z \\ Z^* \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} K_{X\Lambda^{-1}X} & K_{X\Lambda^{-1}X^*} \\ K_{X^*\Lambda^{-1}X} & K_{X^*\Lambda^{-1}X^*} \end{bmatrix}),$$

we get

$$Z^*|Z \sim \mathcal{N}(K_{X^*\Lambda^{-1}X}K_{X\Lambda^{-1}X}^{-1}Z, K_{X^*\Lambda^{-1}X^*} - K_{X^*\Lambda^{-1}X}K_{X\Lambda^{-1}X}K_{X\Lambda^{-1}X^*})$$

For each scalar $z^* \in Z^*$, we can predict the corresponding y^* using

$$p(y^*|z^*) = \frac{1}{1 + e^{-z^*y^*}},$$

By thresholding we predict $y^* = 1$ if $p(y^*|z^*) \geq 0.5$, and $y^* = 0$ otherwise.

6.4 Estimating the Penalty Matrix Λ

In many real applications the task-relation is unknown to us beforehand, thus we need to estimate the penalty matrix Λ to correctly identify the relationship among all the tasks.

The Bayesian interpretation of estimating the task weight w using MT-KRR is by maximizing the posterior distribution of w:

$$\max_{w} p(w|X, Y) \propto p(X, Y|w) * p(w),$$

where $Y = Xw + \epsilon$ with Gaussian noise ϵ, and we assume a Gaussian prior distribution on w:

$$w \sim \mathcal{N}(0, \Lambda^{-1}),\ p(w) \propto \exp\{-w^\top \Lambda w\}.$$

If we maximize the log likelihood of the posterior distribution with kernelization on feature X, we have the MT-KRR objective:

$$\min_{w} \frac{1}{m}||Y - \Phi(X)w||_F^2 + w^\top \Lambda w.$$

Hence, the inversed penalty matrix Λ^{-1} acts as a covariance matrix for the weight vector w. After estimating w using the initial value of Λ^{-1}, we can use the sample covariance matrix of w to re-estimate Λ^{-1}.

Denote the $q \times T$ matrix $W = [\mathbf{w}_1 \mathbf{w}_2 \ldots \mathbf{w}_T]$, the sample covariance matrix of W is estimated as:

$$\text{Cov}(W) = \frac{1}{q-1}(W - \bar{W})^\top (W - \bar{W}),$$

where $\bar{W}$ is $1/q$ of the sum of all the rows in W. Note this is the MLE estimate the of the covariance matrix. When the number of tasks T is large, we can have an MAP estimate of the covariance matrix by adding regularizations on Λ^{-1} (e.g., nuclear norm or $\ell 1$ norm).

For a feature map $x \to \phi(x)$ and $\phi(x) \in \mathcal{R}^q$ where q is a finite number, we can directly estimate the sample covariance matrix of w. For a feature map $x \to \phi(x)$ and $\phi(x) \in \mathcal{R}^q$ where $q = \infty$, e.g., using the Gaussian kernel, then we can approximate the sample covariance matrix of w by using random Fourier features [53].

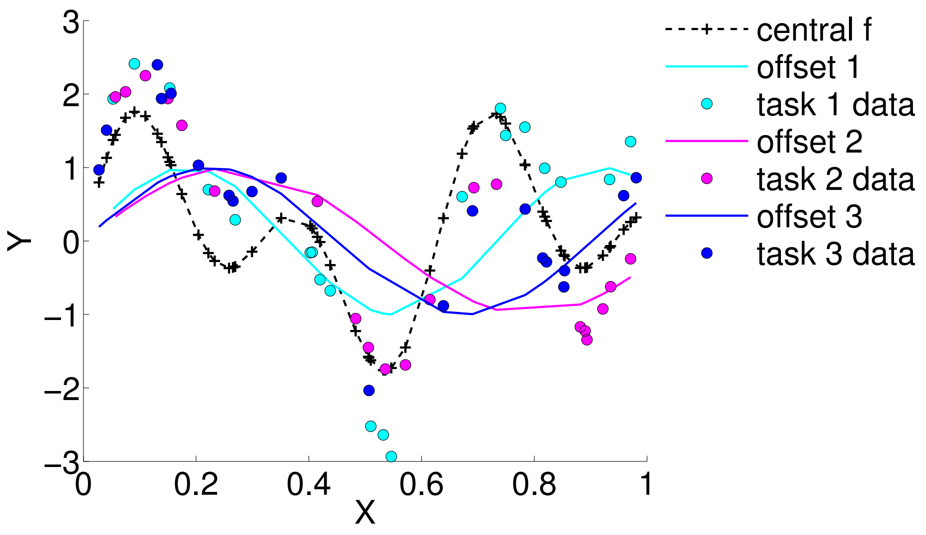


Figure 6.1: Example data with $T = 3, n_t = 20$

6.5 Active Multi-Task Learning

We can regard the multi-task learning with penalty matrix Λ as one Gaussian Process where the kernel matrix incorporated Λ^{-1}, hence the covariance matrix for a new data matrix X^* can be written as:

$$\Sigma(f^*|X, Y, X^*) = K_{X^*\Lambda^{-1}X^*} - K_{X^*\Lambda^{-1}X}(K_{X\Lambda^{-1}X} + \lambda I)^{-1}K_{X\Lambda^{-1}X^*},$$

where $K_{X^*\Lambda^{-1}X} = \Phi(X)\Lambda^{-1}\Phi^\top(X^*)$, and λ is a constant to make sure the kernel matrix is invertible.

Given the covariance matrix, we can adopt the active learning strategy which minimizes the total uncertainty:

$$\arg\min_{X_A} \text{Tr}[\Sigma(f^U|X \cup X_A, Y \cup Y_A, X_U)],$$

where X_A is the selected set of points for query, and X_U are the data points remained.

Alternatively, we can adopt the active surveying strategy where the goal is to predict the sum of all the labels, and the corresponding objective is:

$$\arg\min_{X_A} \mathbf{1}^\top\Sigma(f^U|X \cup X_A, Y \cup Y_A, X_U)\mathbf{1}.$$

6.6 Experiments

6.6.1 Synthetic Data

Multi-Task Learning Stability

To show the stability bounds under different penalties, we simulate data with T tasks. Each task t has $\{X_t, Y_t\} : Y_t = f_c + f_o + 0.1\epsilon$, where $f_c = \sin(20x) + \sin(10x)$ is the central function,

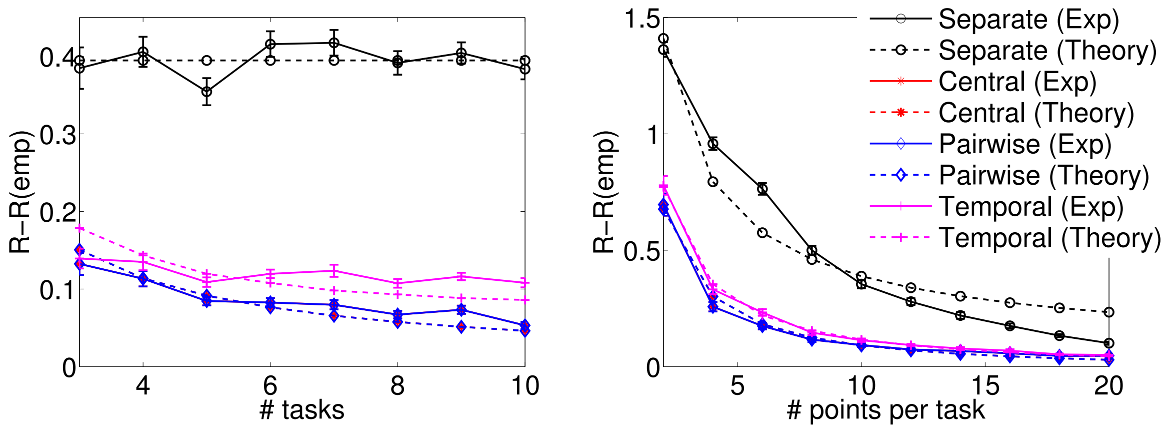


Figure 6.2: Left: $R - R_{emp}$ w.r.t # tasks (fixed $n_t = 10$ points per task); Right: $R - R_{emp}$ w.r.t # points per task (fixed $T = 5$ tasks)

and $f_o = \sin 5(1 + t_i)x$ is a smoother additive function, with $t_i \sim$ Unif$(0, 1)$, plus $\epsilon \in \mathcal{N}(0, 1)$. Fig.6.1 shows an example of the data with $T = 3$ and $n_t = 20$ per task.

In Fig.6.2, we plot the risk difference $R - R_{emp}$ (Sec. 6.2.3) w.r.t different number of tasks (fixed 10 points per task), and different number of points per task (with fixed 5 tasks), averaged over 50 experiments. We also plot the theoretical bounds (fitted to the actual curve using regression) for each case. We see that the results are consistent with our analysis. Using central+offset (Eq. 6.8), pairwise-penalty (Eq. 6.9), or temporal-penalty (Lemma 6.2.5) we achieve tighter bounds than learning each task independently (denoted as Separate). In addition, central+offset and pairwise-penalty result in the same curve (red and blue) when we set λ_p/T in central+offset equal to λ_p in pairwise-penalty, which shows the equivalence of these two methods. Further we observe that temporal-penalty gives slightly larger $R - R_{emp}$ than central+offset and pairwise-penalty, which coincides with our analysis.

Active Multi-Task Learning

To show the results for active multi-task learning, we simulated T tasks with each task consisting of $\{X_t, Y_t\}$ which satisfies: $Y_t = f_c + f_o + 0.1\epsilon$, where $f_c = \sin(20x) + \sin(10x)$ acts as the central function, and $f_o = \sin 5(1 + t_i)x$ is a smoother additive function, with $t_i \sim$ Unif$(0, 1)$, plus Gaussian noise $\epsilon \in \mathcal{N}(0, 1)$. Figure 6.3 (left) shows an example of the data with $T = 3$ and $n_t = 20$ per task. Figure 6.3 (right) shows the mean squared error w.r.t. number of iterations, with random selection and active selection. The results are averaged over 30 experiments. Each experiment is randomly initialized with 1 labeled point per task, and in each iteration one test point is selected and its label is acquired. We can see that our active selection yields a much lower MSE compared to random selection.

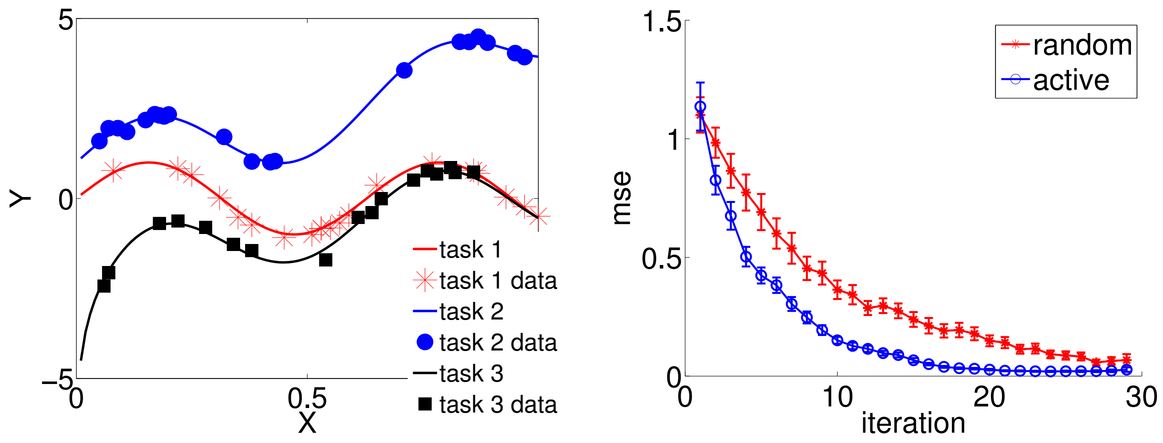


Figure 6.3: Example of the multi-task dataset used in the experiments, with $T = 3$ tasks and $n_t = 20$ data points per task(left); MSE w.r.t. number of iterations in active learning (right)

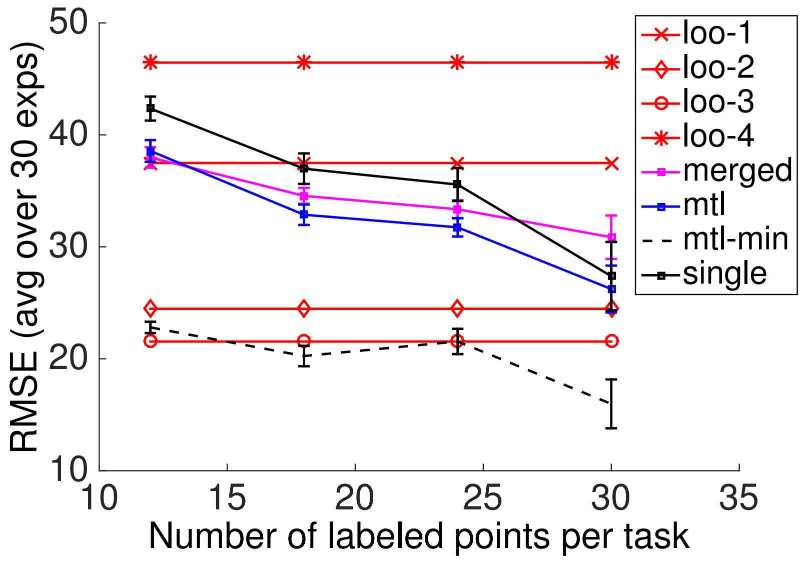


Figure 6.4: Results for multi-task learning on the AQI data

6.6.2 Real Data

Multi-Task Learning Stability

AQI data. The first real dataset is the Air Quality Index (AQI) dataset [45]. We extract bag-of-words vectors (feature X with dimension $d = 100,395$) from social media posts to predict the AQI (label Y) across cities. The results are averaged over 20 experiments. In Fig. 6.4, we show the prediction error of MT-KRR using pairwise penalty (or equivalently the central+offset penalty) with 4 cities as 4 different tasks. We see that the MT-KRR algorithm (mtl) outperforms independent-task-learning (single). In addition, we plot the leave-one-out error for each task (loo-1 through 4), and the prediction error by MT-KRR for the best task (mtl-min), which outperforms learning that task by itself (loo-3).

Extension to Multi-Task Classification

Robot Data. The second dataset is a robotics dataset where the task is to predict whether it is successful for a robot snake to climb over obstacles with different heights. The data was taken by first trying to get the snake to climb over a 3.5” beam, then a 5.5” beam, then 7”, 9”, 11”, and 12.5”. Each task corresponds to a different beam height. The features include the beam height and three control parameters, and the label is a binary ”success/failure” outcome from each experiment. Since it is a classification task, we apply the proposed multi-task classification method as described in Sec. 6.3. In Figure 6.5 we show the results of the proposed multi-task learning algorithm with pairwise penalty (**mtl**) compared with **merged** (pooling all tasks together) and **single** (learning each task independently). We can see that our method gives the best results, which demonstrates the effectiveness of coupling tasks together for better learning results.

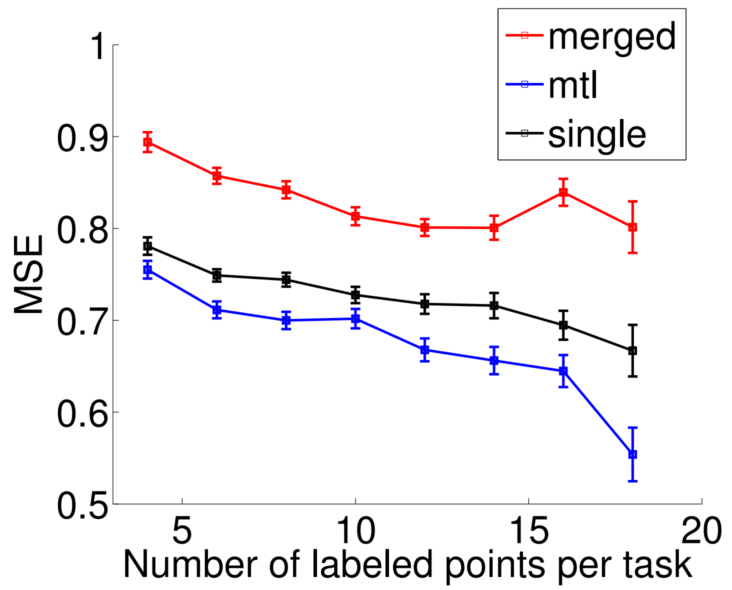


Figure 6.5: Results for multi-task learning on the Robot data (classification)

Multi-Task Learning with updated penalty matrix Λ

School Data. The dataset is a school dataset[1] [4, 6] where the task is to predict the examination scores based on school features (e.g., percentage of students eligible for free school meals, school gender, school denomination) and student features (e.g., year, gender, VR band, ethnic group) to test the effectiveness of schools. It consists of examination records from 139 secondary schools in years 1985, 1986 and 1987, with a random 50% sample with 15362 students. We regard each

[1] http://cvn.ecp.fr/personnel/andreas/code/mtl/index.html

school as a different task. In Figure. 6.6, we show the results of our proposed multi-task learning method with updated penalty matrix Λ (**mtl-rand-feat**), and the multi-task learning method with a fixed penalty matrix Λ by using the pair-wise penalty (**mtl-fixed**), compared with the merged algorithm (**merged**, by pooling all tasks together), single-task learning (**single**, learning each school independently), and the multi-task learning algorithm using sparse penalty (**mtl-sparse**, [81]). We can see that our proposed algorithm with updated penalty matrix gives the best results, which shows the effectiveness of updating the task-relation matrix learned from the task weights.

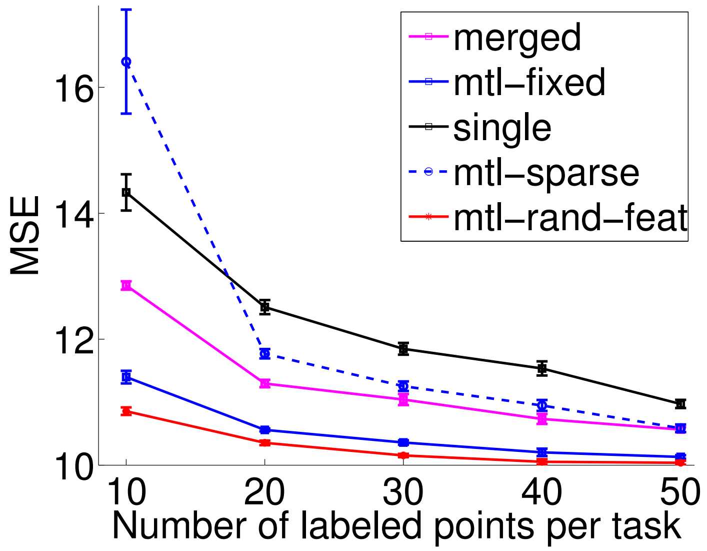


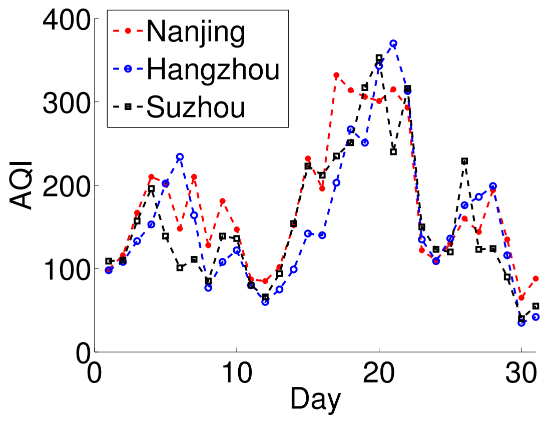
Figure 6.6: Results for multi-task learning on the School data with updated penalty matrix Λ

Active Multi-Task Learning

In Figure 6.7, we show the results of active learning on the AQI data. In the left figure we show an example of the AQI w.r.t. different days, for three cities as our three tasks. In the right figure we compare our active learning result with random selection. We can observe that again our active selection criteria yields a much lower error on the real data, which shows we can effectively predict the AQI for a certain city by querying a few labels from other related cities.

6.7 Conclusion

In this chapter we provide theory that connects the risk bounds for multi-task learning to the relation of tasks. We show that, by imposing a smooth relationship between/among tasks, we obtain favorable learning rates for multi-task learning, compared to learning tasks independently. In addition, we have shown the stability bounds for several existing multi-task learning algorithms, and we prove that certain assumptions imposed on task relations can yield tighter stability bounds than other assumptions. Further we proposed an algorithm which can automatically update the



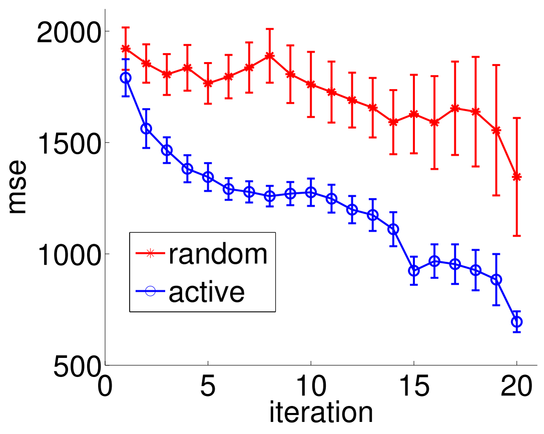


Figure 6.7: AQI dataset, with $T = 3$ tasks and $n_t = 31$ data points per task (left); MSE w.r.t. number of iterations in active learning (right)

task correlation matrix, thus capturing both positive and negative relations among different tasks. Finally, we proposed a combined active multi-task learning framework which outperforms random selection by querying a few most informative data points.

Chapter 7

Transfer Learning with Distributions

7.1 Overview

In this chapter, we propose method for estimating **distributions** in a **transfer learning** setting:

- In Section 7.2.1, we propose an approach for **distribution transfer** on the label distribution $P(Y)$ in a nonparametric setting, i.e., estimating $P(Y)$ in the target domain by transferring $P(Y)$ from the source domain.
- In Section 7.2.2, we propose an approach for **conditional distribution transfer**, i.e., estimating $P(Y|X = x^*)$ for features X in the target domain by transferring from the source distribution.
- In Section 7.3, we extend the algorithm to **multi-source distribution transfer**, where we estimate $P(Y)$ or $P(Y|X = x^*)$ in the target domain by combining multiple source distributions.

7.2 Single-Source Distribution Transfer

Notations. Let $y \in \mathcal{R}$ denote the labels and $x \in \mathcal{R}^d$ denote the d-dimensional features/observations. Throughout the section we use superscript s to indicate the variables for the *source* domain, and t to indicate the variables for the *target* domain. For example, the variables y^s and y^t refer to the variables for source labels and target labels.

Let $\mathcal{H}$ be a reproducing kernel Hilbert space with kernel K such that $k(y, y) \leq \kappa^2 < \infty$ for all $y \in Y$. Let $||.||$ denote the corresponding RKHS norm. Let ψ denote the feature mapping on y associated with kernel K, and $\Psi(Y)$ denote the matrix where the i-th column is $\psi(y_i)$. Denote $K_{YY'} = \Psi^\top(Y)\Psi(Y')$ as the kernel computed between Y and Y', i.e., $K_{YY'}(i, j) = k(y_i, y'_j)$.

Similarly, we use ϕ to denote the feature map on x, and the corresponding matrix as $\Phi(X)$. Denote $K_{XX'} = \Phi^\top(X)\Phi(X')$ as the kernel computed between feature matrix X and X', i.e., $K_{XX'}(i, j) = l(x_i, x'_j)$. Again we assume $l(x, x) \leq \kappa^2 < \infty$ for all $x \in X$.

7.2.1 Distribution Transfer on $P(y)$

In this section, we propose a method for distribution transfer on $P(y)$, without using the information from x. Formally, assume we have m source samples $Y^s = \{Y_1^s, \ldots, Y_m^s\}$ drawn from the source distribution $P(y^s)$. In addition, we have n_l target samples $Y^{tl} = \{Y_1^t, \ldots, Y_{n_l}^t\}$ drawn from the target distribution $P(y^t)$. We would like to predict $P(y^t)$ based on the source samples Y^s and the few target samples Y^{tl} (Fig. 7.1).

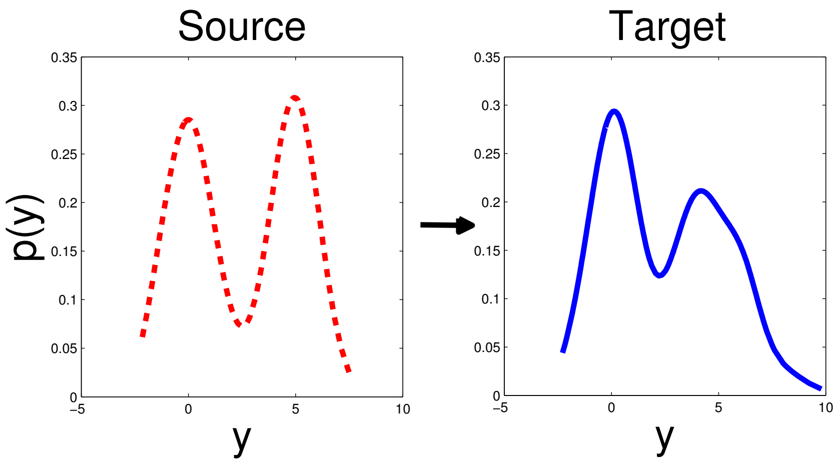


Figure 7.1: Example distribution transfer on $P(y)$

The usual assumption for transfer learning is that $m \gg n_l$, i.e., the number of source samples is sufficient but the number of labeled target samples is very limited. As a result, we will not have a very accurate estimation of $P(y^t)$ based on a few labeled target samples only. Hence the idea for transfer is that, the prediction on the target distribution should have some dependency on the source distribution $P(y^s)$, and a better estimation on $P(y^t)$ can be achieved if $P(y^s)$ and $P(y^t)$ are somewhat similar.

Specifically, we propose a reweighting w on the source Y^s to trade-off between

- **Source dependency**: the difference between the weighted source distribution and the original source distribution (characterized by the symmetrised KL divergence),
- **Matching target**: the weighted source distribution should be close to the target distribution based on the observed target samples.

For **Matching target** we minimize the MMD [26, 27, 33] between them (here we prefer using MMD instead of KL divergence because KL requires estimating both densities, and the density based on the observed target samples could be very inaccurate due to the small sample size n_l. This is not a problem for the source dependency part because both weighted source distribution and the original source distribution are estimated based on sufficient source samples).

Let q represent the weighted source distribution with weights $\mathbf{w} \in \mathcal{R}^{m\times 1}$, and p represent the estimated source distribution based on the source samples, the objective is:

$$\begin{aligned}
&\min_{\mathbf{w}} \frac{1}{2}\text{KL}(p(y^s)||q(y^s)) + \frac{1}{2}\text{KL}(q(y^s)||p(y^s)),\\
&\text{subject to } ||\Psi(Y^s)\mathbf{w} - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}|| \le \epsilon,\\
&\text{and } \forall i, w_i \ge 0, \mathbf{w}^\top \mathbf{1}_m = 1,
\end{aligned} \tag{7.1}$$

where $\mathbf{1}_{n_l}, \mathbf{1}_m$ are vectors of 1's with size $n_l \times 1$, and $m \times 1$, respectively. ϵ controls the dependency on the source domain. Here $p(y^s_j) = \frac{1}{m}\sum_{i=1}^m K_h(y^s_j - Y^s_i)$ is the estimated density based on the source data, and $q(y^s_j) = \sum_{i=1}^m \frac{w_i}{\sum_i w_i} K_h(y^s_j - Y^s_i)$ is the weighted density estimation based on the source data (here we omit the normalizing constant h since it does not affect the optimization). K_h represents the smoothing kernel used for density estimation with kernel width h. Using matrix notations we can further simplify the objective and get (note here the KL divergence is being measured only on the samples):

$$\begin{aligned}
&\frac{1}{2}\text{KL}(p(y^s)||q(y^s)) + \frac{1}{2}\text{KL}(q(y^s)||p(y^s))\\
=&C - \frac{1}{2}\sum_j p(y^s_j)\log q(y^s_j) - \frac{1}{2}\sum_j q(y^s_j)\log p(y^s_j) + \frac{1}{2}\sum_j q(y^s_j)\log q(y^s_j)\\
=&C - \frac{1}{2m}\mathbf{1}_m^\top K_s \log(K_s\mathbf{w}) - \frac{1}{2}\mathbf{w}^\top K_s \log(K_s\mathbf{1}_m/m) + \frac{1}{2}\mathbf{w}^\top K_s \log(K_s\mathbf{w}),
\end{aligned}$$

where K_s is the $m \times m$ kernel matrix on the source data, i.e., $K_s(i,j) = k(y^s_i, y^s_j)$.

Alternatively, we can minimize the MMD between the weighted source distribution and the original source distribution, i.e.,

$$\begin{aligned}
&\min_{\mathbf{w}} ||\Psi(Y^s)\mathbf{w} - \frac{1}{m}\Psi(Y^s)\mathbf{1}_m||^2,\\
&\text{subject to } ||\Psi(Y^s)\mathbf{w} - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}|| \le \epsilon,\\
&\text{and } \forall i, w_i \ge 0, \mathbf{w}^\top \mathbf{1}_m = 1.
\end{aligned} \tag{7.2}$$

Choice of ϵ: the lower bound. To ensure that we have non-empty solutions for both optimization problems Eq. 7.1 and Eq. 7.2, we first work out a lower bound of ϵ by minimizing the left hand side of the constraint, i.e.,

$$\begin{aligned}
&\min_{w} ||\Psi(Y^s)\mathbf{w} - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}||,\\
&\text{subject to } \forall i, w_i \ge 0, \mathbf{w}^\top \mathbf{1}_m = 1.
\end{aligned} \tag{7.3}$$

Denote the solution to Eq. 7.3 as $\bar{w}$, and the minimized objective function as $\bar{\epsilon}$. As long as we choose an ϵ larger than $\bar{\epsilon}$, we have non-empty solutions to the original optimization problem. With a PSD kernel matrix, the above minimization Eq. 7.3 is a convex quadratic programming problem. The solution $\bar{w}$ reweights the source to match the target without any dependence on the

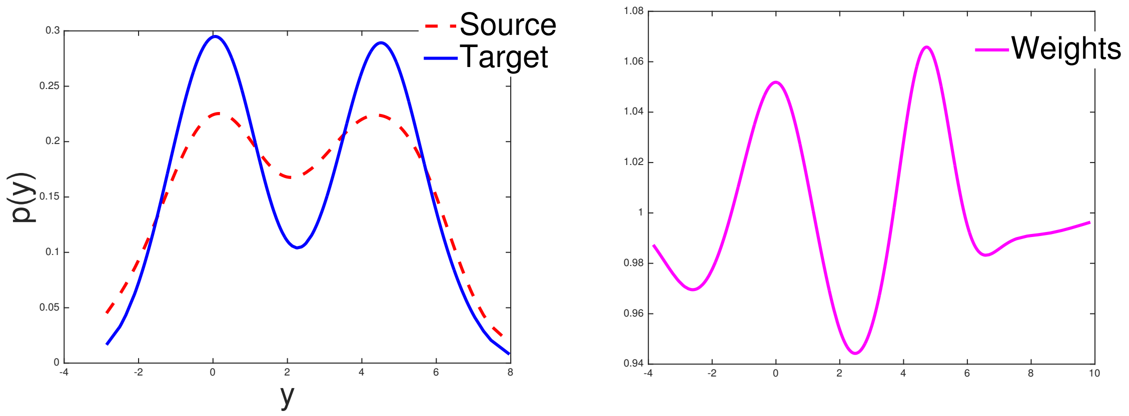


Figure 7.2: Illustration of the distribution transfer algorithm: the example reweights

source distribution. With a small number of target labels this can lead to distribution estimation on the target with high variance. By increasing ϵ and adding the objective for source dependence (i.e., objective Eq. 7.1 and Eq. 7.2), we can have a better estimation on the target distribution by assuming the source and target distributions are similar.

We use interior point methods to solve the optimization problems. After obtaining the weights $\mathbf{w}$ on the source data, we can estimate $P(y^t)$ by:

$$\hat{P}(y^t) = \frac{1}{h(\sum_i w_i + n_l)}[\sum_{i=1}^{m} w_i K_h(y^t - Y_i^s) + \sum_{i=1}^{n_l} K_h(y^t - Y_i^{tl})],$$

where K_h is the smoothing kernel with width h.

In Figure 7.2, we show an example of transferring the source distribution (red) to the target distribution (blue). The right figure shows an example of the reweights $\mathbf{w}$ after we optimize the objective. We can see that the reweights can correctly up-weight the source samples around the two peaks, and down-weight the source samples in the middle, to make the source distribution match the target distribution.

Convergence Analysis. We analyze the convergence of the reweighted source distribution to the true target distribution. In kernel mean matching (KMM) [26, 27, 33], the discrepancy minimization is between $P(x^s)$ and $P(x^t)$, by assuming sufficient samples X^s and X^t. As a result, the authors assume that replacing one y_i^s by y changes only one term with β_i, because β's are computed on x only and $P(Y|X)$ stays the same. Since we allow $P(Y|X)$ to change, replacing y_i can lead to changes in the whole vector β. Hence we give a general upper bound on β in our analysis.

Denote the mean map of a distribution P_y as $\mu[P_y] := \mathbb{E}_y[k(y, \cdot)]$, and its empirical estimate $\hat{\mu}_Y = \frac{1}{m}\sum_{i=1}^{m} k(y_i, \cdot)$, based on the samples $Y = \{y_1, ..., y_m\}$ drawn i.i.d from P_y [65]. Denote $\lambda_{\max}(A)$, $\lambda_{\min}(A)$ as the largest/smallest eigenvalue of a symmetric matrix A. We have the following theorem:

Theorem 7.2.1. *Let β be the solution to objective Eq. 7.1 or Eq. 7.2. Denote the risk R as the difference between the reweighted mean embedding and the mean map of P_{Y^t}, i.e., $R =$*

$||\Psi(Y^s)\beta - \mu[P_{Y^t}]||$, *and the empirical risk as* $\hat{R} = ||\Psi(Y^s)\beta - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}||$. *With probability* $1-\delta$, *we have*

$$R \leq \hat{R} + n_l^{-1/2}(C + (\frac{2\lambda_{\max}^{1/2}(K_{Y^sY^s})(\epsilon n_l + \kappa)}{\lambda_{\min}^{1/2}(K_{Y^sY^s})} + \kappa)\sqrt{2\log 2/\delta}),$$

where $C > 0$ *is some constant.*

Proof. Let Y^{tl} and $(Y^{tl})'$ differ by exactly one data point. Let β be the solution to objective Eq. 7.1 or 7.2 with Y^{tl}, and β' be the solution to the same objective with $(Y^{tl})'$. Since β and β' satisfy the constraints, using triangle inequality we have

$$\begin{aligned}
||\Psi(Y^s)\beta - \Psi(Y^s)\beta'|| &\leq ||\Psi(Y^s)\beta - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}|| \\
&\quad + ||\Psi(Y^s)\beta' - \frac{1}{n_l}\Psi[(Y^{tl})']\mathbf{1}_{n_l}|| + ||\frac{1}{n_l}\Psi[(Y^{tl})']\mathbf{1}_{n_l} - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}|| \\
&\leq 2\epsilon + \frac{2\kappa}{n_l},
\end{aligned}$$

since $||\frac{1}{n_l}\Psi[(Y^{tl})']\mathbf{1}_{n_l} - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}||$ is bounded by $\frac{1}{n_l}||\psi(y_i^{tl}) - \psi[(y_i^{tl})']|| \leq \frac{2}{n_l}\kappa$.
By the standard property of the Rayleigh quotient for PDS matrices, i.e., $\lambda_{\min}^{1/2}(A)||z||_2 \leq ||Az||_2 \leq \lambda_{\max}^{1/2}(A)||z||_2$ for any $z \in \mathbb{R}^{n\times 1}$ and symmetric matrix $A \in \mathbb{R}^{n\times n}$, the left hand side

$$||\Psi(Y^s)\beta - \Psi(Y^s)\beta'|| \geq \lambda_{\min}^{1/2}(K_{Y^sY^s})||\beta - \beta'||.$$

Hence

$$||\beta - \beta'|| \leq \frac{1}{\lambda_{\min}^{1/2}(K_{Y^sY^s})}(2\epsilon + \frac{2\kappa}{n_l}).$$

Let $\hat{R}' = ||\frac{1}{m}\Psi(Y^s)\beta' - \frac{1}{n_l}\Psi(Y^{tl})'\mathbf{1}_{n_l}||$, we have

$$\begin{aligned}
|\hat{R} - \hat{R}'| &\leq ||\frac{1}{m}\Psi(Y^s)\beta - \frac{1}{m}\Psi(Y^s)\beta'|| + ||\frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l} - \frac{1}{n_l}\Psi(Y^{tl})'\mathbf{1}_{n_l}|| \\
&\leq \frac{1}{m}\lambda_{\max}^{1/2}(K_{Y^sY^s})||\beta - \beta'|| + \frac{2}{n_l}\kappa,
\end{aligned}$$

since $||\psi(y)|| \leq \sqrt{K(y,y)} \leq \kappa$.
Similarly, let $R' = ||\frac{1}{m}\Psi(Y^s)\beta' - \mu[P_{Y^t}]||$, we have

$$\begin{aligned}
|R - R'| &\leq ||\frac{1}{m}\Psi(Y^s)\beta' - \frac{1}{m}\Psi(Y^s)\beta|| \\
&\leq \frac{1}{m}\lambda_{\max}^{1/2}(K_{Y^sY^s})||\beta - \beta'||.
\end{aligned}$$

Hence, with one element replaced in $R - \hat{R}$, we have the difference bounded by:

$$\begin{aligned}
c_i &\leq |\hat{R} - \hat{R}'| + |R - R'| \\
&\leq \frac{2}{m}\lambda_{\max}^{1/2}(K_{Y^sY^s})||\beta - \beta'|| + \frac{2}{n_l}\kappa \\
&\leq \frac{4\lambda_{\max}^{1/2}(K_{Y^sY^s})(\epsilon + \frac{\kappa}{n_l})}{m\lambda_{\min}^{1/2}(K_{Y^sY^s})} + \frac{2\kappa}{n_l},
\end{aligned}$$

and

$$\sum_{i=1}^{n_l} c_i^2 = \frac{1}{n_l}(\frac{4\lambda_{\max}^{1/2}(K_{Y^sY^s})(\epsilon n_l + \kappa)}{m\lambda_{\min}^{1/2}(K_{Y^sY^s})} + 2\kappa)^2.$$

Using Mcdiarmid's inequality and plugging in the bound c_i, we have with probability $1 - \delta$,

$$\begin{aligned} &|(R - \hat{R}) - E_{Y^{tl}}[R - \hat{R}]| \\ &\leq \frac{1}{\sqrt{n_l}}(\frac{2\lambda_{\max}^{1/2}(K_{Y^sY^s})(\epsilon n_l + \kappa)}{\lambda_{\min}^{1/2}(K_{Y^sY^s})} + \kappa)\sqrt{2\log\frac{2}{\delta}}. \end{aligned} \tag{7.4}$$

Now we bound on the expectation (Thm. 2, [65]):

$$E_{Y^{tl}}[R - \hat{R}] \leq E_{Y^{tl}}||\frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l} - \mu[P_{Y^t}]|| = O(n_l^{-1/2}). \tag{7.5}$$

Combining Eq 7.4 and Eq 7.5 concludes the proof. □

Remark. The above analysis suggests that a good choice of ϵ should be in the order of $O(n_l^{-1})$ for good convergence properties. On the other hand, a possible tighter bound can be achieved by using $||\Psi(Y^s)\beta - \mu[P_{Y^t}]|| \leq ||\Psi(Y^s)\beta - \frac{1}{m}\Psi(Y^s)\mathbf{1}_m|| + ||\frac{1}{m}\Psi(Y^s)\mathbf{1}_m - \mu[P_{Y^s}]|| + ||\mu[P_{Y^s}] - \mu[P_{Y^t}]|| \leq \epsilon_1 + C_1 m^{-1/2} + ||\mu[P_{Y^s}] - \mu[P_{Y^t}]||$ where ϵ_1 is the minimized objective in Eq. 7.2 and $C_1 > 0$ is some constant. Since $m \gg n_l$, and at the early stage of transfer n_l is small, we should choose a relatively larger ϵ such that $\epsilon_1 = ||\Psi(Y^s)\beta - \frac{1}{m}\Psi(Y^s)\mathbf{1}_m||$ is small. In this case, when the source distribution is close to the true target distribution, which means $||\mu(P_Y^s) - \mu(P_Y^t)||$ can be much smaller than C/n_l, then we get a tighter bound on the difference of risks, which is the benefit we get from transfer.

7.2.2 Conditional Distribution Transfer

If we further have the information on x, we can predict the distribution of y^t given a certain x^*, i.e., $P(y^t|X^t = x^*)$ (Fig. 7.3).

Let $(X^s, Y^s) = \{(X_1^s, Y_1^s), \ldots, (X_m^s, Y_m^s)\}$ denote m labeled source samples and (X^{tl}, Y^{tl}) denote n_l labeled target samples. First we have the embedding of conditional distribution $P(y|X = x^*)$ given x^*[67]:

$$\begin{aligned} \hat{\mu}(Y|X = x^*) &= \Psi(Y)(K_{XX} + \lambda I)^{-1}K_{Xx^*} \\ &= \Psi(Y)\mathbf{w}^* = \sum_i w_i\psi(y_i), \end{aligned}$$

where $\mathbf{w}^* = (K_{XX} + \lambda I)^{-1}K_{Xx^*}$ is a $|Y| \times 1$ vector. It is easy to see that the weighting w_i on $\psi(y_i)$ effectively measures the closeness of x^* to each x_i.

The key idea here for transfer is that, we want to modify the weighting w_i on the source data, such that the conditional distribution $P(y|X = x^*)$ can be matched between the source and the target domain. More specifically, we minimize the discrepancy between a reweighted source

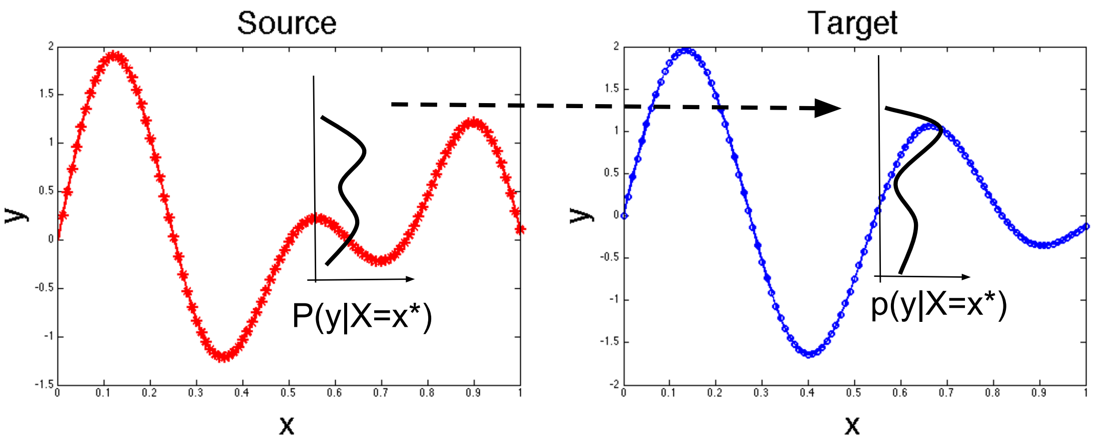


Figure 7.3: Distribution transfer on $P(y|X = x^*)$

distribution with reweights $\beta \in \mathcal{R}^{m\times 1}$ and the original source distribution, while keeping the discrepancy between the reweighted $\hat{\mu}(Y^s|X^s = x^*)$ and $\hat{\mu}(Y^{tl}|X^{tl} = x^*)$ small, i.e.,

$$\begin{aligned} &\min_{\beta} ||\Psi(Y^s)\beta - \Psi(Y^s)\mathbf{w}^s||^2, \\ &\text{subject to } ||\Psi(Y^s)\beta - \Psi(Y^{tl})\mathbf{w}^{tl}|| \le \epsilon, \\ &\text{and } \forall i, \beta_i \ge 0, \beta^\top \mathbf{1}_m = 1, \end{aligned} \tag{7.6}$$

where $\mathbf{w}^{tl} = (K_{X^{tl}X^{tl}} + \lambda I)^{-1} K_{X^{tl}x^*}$ is the weighting on $\Psi(Y^{tl})$ based on X^{tl}, and $\mathbf{w}^s = (K_{X^sX^s} + \lambda I)^{-1} K_{X^sx^*}$ is the weighting on $\Psi(Y^s)$ based on X^s. Again ϵ controls the dependency on the source. Similarly, after obtaining β we estimate $P(y^t|X = x^*)$ by [24, 57]:

$$\hat{P}(y^t|X = x^*) = \frac{1}{h(\sum_i \beta_i + \sum_i w_i^{tl})}[\sum_{i=1}^{m} \beta_i K_h(y^t - Y_i^s) + \sum_{i=1}^{n_l} w_i^{tl} K_h(y^t - Y_i^{tl})].$$

It is easy to see that when the kernel width for X goes to infinity, all the data points will have the same similarity to x^*, i.e., each data point is assigned a weight of $1/m$ in the source domain, and $1/n_l$ in the target domain, respectively. Then the above objective Eq. 7.6 becomes equivalent to Eq. 7.2, i.e., for distribution prediction without information on x, we simply set $w_i^s = \frac{1}{m}$ and $w_i^{tl} = \frac{1}{n_l}$.

7.3 Multi-Source Distribution Transfer

In this section, we extend the previous algorithm to multi-source distribution transfer on $P(y)$, and multi-source conditional distribution transfer on $P(y|X = x^*)$.

7.3.1 Multi-Source Distribution Transfer on $P(y)$

Assume we have S sources, where each source s_i has m_i samples $Y^{s_i} = \{Y_1^{s_i}, \ldots, Y_{m_i}^{s_i}\}$. The goal is to estimate $P(y^t)$ by combining $P(y^{s_i})$ from S sources.

We assume a convex combination of the source distribution and let $\alpha \in \mathbb{R}^{S \times 1}$ be the weighting on the sources, the objective we would like to minimize is:

$$\begin{aligned} &\min_{\alpha} || \sum_i \alpha_i [\Psi(Y^{s_i})\beta^{s_i}] - \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}||^2, \\ &\text{subject to } \forall i, \alpha_i \geq 0, \sum_i \alpha_i = 1. \end{aligned} \tag{7.7}$$

Here β^{s_i} is the optimized reweight of objective Eq. 7.1 or Eq. 7.2 for source s_i.

7.3.2 Multi-Source Conditional Distribution Transfer

Similarly, for multi-source conditional distribution transfer, we have S sources, where each source s_i has m_i labeled samples $(X^{s_i}, Y^{s_i}) = \{(X_1^{s_i}, Y_1^{s_i}), \ldots, (X_{m_i}^{s_i}, Y_{m_i}^{s_i})\}$. We have n target samples $X^t = \{X_1^t, \ldots, X_n^t\}$, but only n_l of them are labeled, denoted as (X^{tl}, Y^{tl}). The goal is, given a query point x^* in the target domain, we would like to estimate $P(y^t|X^t = x^*)$ by combining S sources. Using the taxi example, consider each source/target as data (pickup and dropoff location pairs) from different time frames (different hour or day). Assume we are given a pickup location x^* in the current time frame, we would like to estimate the distribution of the dropoff locations $P(y^t|X^t = x^*)$ by combining the data from previous time frames. Intuitively, we should favor the sources with $P(Y|X)$ closer to the $P(Y|X)$ of the target. For example, given the target data from 8-9am on a Monday, we should favor the sources from adjacent time frames, and sources from weekdays rather than weekends.

Similar to Eq. 7.7, the objective for multi-source conditional distribution transfer is:

$$\begin{aligned} &\min_{\alpha} || \sum_i \alpha_i [\Psi(Y^{s_i})\beta^{s_i}] - \Psi(Y^{tl})\mathbf{w}^{tl}||^2, \\ &\text{subject to } \forall i, \alpha_i \geq 0, \sum_i \alpha_i = 1. \end{aligned} \tag{7.8}$$

Here β^{s_i} is the optimized reweight of Eq. 7.6 for source s_i.

A Reweighted Set-Kernel Point of View. We analyze why this objective yields the desired property on the weights α for each source. Denote $\tilde{Y}^{s_i} = \Psi(Y^{s_i})\beta^{s_i}$, and $\tilde{Y}^{tl} = \frac{1}{n_l}\Psi(Y^{tl})\mathbf{1}_{n_l}$ for distribution transfer or $\tilde{Y}^{tl} = \Psi(Y^{tl})\mathbf{w}^{tl}$ for conditional distribution transfer. Using Lagrange multipliers Eq. 7.7 or Eq. 7.8 can be written as:

$$L = \alpha^\top \bar{K}_{\tilde{Y}^s\tilde{Y}^s}\alpha - 2\alpha^\top \bar{K}_{\tilde{Y}^s\tilde{Y}^{tl}} - 2\alpha^\top \mathbf{u} + 2\lambda(\alpha^\top \mathbf{1}_S - 1),$$

where $\bar{K}_{\tilde{Y}^s\tilde{Y}^s}$ is an $S \times S$ matrix with

$$\bar{K}_{\tilde{Y}^s\tilde{Y}^s}(i,j) = (\beta^{s_i})^\top K_{Y^{s_i}Y^{s_j}}\beta^{s_j},$$

$\bar{K}_{\tilde{Y}^s\tilde{Y}^{tl}}$ is an $S \times 1$ vector with

$$\bar{K}_{\tilde{Y}^s\tilde{Y}^{tl}}(i) = (\beta^{s_i})^\top K_{Y^{s_i}Y^{tl}}(\frac{1}{n_l}\mathbf{1}_{n_l}) \text{ or } (\beta^{s_i})^\top K_{Y^{s_i}Y^{tl}}(\mathbf{w}^{tl}),$$

and $\mathbf{u}$ is an $S \times 1$ vector with $u_i \geq 0$, $\lambda > 0$. By the KKT stationarity conditions we have:

$$\alpha = (\bar{K}_{\tilde{Y}^s\tilde{Y}^s})^{-1}(\bar{K}_{\tilde{Y}^s\tilde{Y}^{tl}} + \mathbf{u} - \lambda\mathbf{1}_S).$$

Note here $\bar{K}_{\tilde{Y}^s\tilde{Y}^{tl}}$ measures the discrepancy between two distributions, the source distribution reweighted by β^{s_i}, and the target distribution (or conditional distribution). In other words, it measures how "good" each source is when transferring to the target. We can think of $(\beta^{s_i})^\top K_{Y^{s_i}Y^{tl}}$ as a **reweighted** set kernel [9, 23] between $\Psi(Y^{s_i})\beta^{s_i}$ and $\Psi(Y^{tl})$, which effectively measures the similarity between two sets of labels: the reweighted source labels and the target labels.

7.3.3 Regions with Different $P(X)$'s: Combining Both $P(Y)$ and $P(X)$

In this section, we extend to the case when we are interested in regions with varying $P(X)$, instead of a single location x^*. Again using the taxi example, consider each source as data from different pickup regions, i.e., different $P(X)$'s. As a result, we should favor the sources with both $P(X)$ and $P(Y)$ closer to the target. For example, given target data from a certain pickup region, the source data from nearby pickup regions (more similar $P(X)$) should be more favorable. In addition, among those sources, the one(s) with similar types (e.g., residential or business, thus are more likely to yield a similar $P(Y)$) to the target should be assigned with larger weights than the rest.

Formally, let $\gamma \in \mathbb{R}^{S\times 1}$ be the weighting on the sources, the objective is:

$$\begin{aligned}\min_\gamma\ &||\sum_i \gamma_i[\frac{1}{m_i}\Phi(X^{s_i})\mathbf{1}_{m_i}][(\beta^{s_i})^\top\Psi^\top(Y^{s_i})] - \\ &[\frac{1}{n}\Phi(X^t)\mathbf{1}_n][(\mathbf{1}_{n_l})^\top\Psi^\top(Y^{tl})]||^2, \\ &\text{subject to } \forall i, \gamma_i \geq 0, \sum_i \gamma_i = 1.\end{aligned} \tag{7.9}$$

Here $\frac{1}{|X|}\Phi(X)\mathbf{1}$ is the empirical estimator of $\mu[P_x]$, and β^{s_i} is the optimized reweight of Eq. 7.1 or Eq. 7.2 for source s_i (assume normalized). Note we use all target observations X^t to estimate $\mu[P_{X^t}]$, while the estimation on $P(Y)$ is done on Y^{tl} since we only have access to n_l target labels.

In the following we analyze why this objective yields the desired property on the weights γ for each source. Similar to Sec. 7.3, using Lagrange multipliers the objective can be written as:

$$L = \gamma^\top A\gamma - 2\gamma^\top\mathbf{b} - 2\gamma^\top\mathbf{u} + 2\lambda(\gamma^\top\mathbf{1}_S - 1),$$

where A is an $S \times S$ matrix with

$$A_{ij} = \frac{1}{m_i m_j}[(\mathbf{1}_{m_i})^\top K_{X^{s_i}X^{s_j}}\mathbf{1}_{m_j}][(\beta^{s_j})^\top K_{Y^{s_j}Y^{s_i}}\beta^{s_i}],$$

$\mathbf{b}$ is an $S \times 1$ vector with

$$\mathbf{b}_i = \frac{1}{m_i n}[(\mathbf{1}_{m_i})^\top K_{X^{s_i}X^t}\mathbf{1}_n][\frac{1}{n_l}\mathbf{1}_{n_l}K_{Y^{tl}Y^{s_i}}\beta^{s_i}].$$

$\mathbf{u}$ is an $S \times 1$ vector with $u_i \geq 0$, $\lambda > 0$. m_i, m_j represent the number of data points in source s_i and s_j, respectively. By the KKT stationarity conditions we have:

$$\gamma = A^{-1}(\mathbf{b} + \mathbf{u} - \lambda \mathbf{1}_S).$$

Note here $\mathbf{b}$ acts as a key term that affects the relative value of γ. We obtain larger b_i if both $(\Phi(X^{s_i}),\ \Phi(X^t))$, and $(\Psi(Y^{tl})\mathbf{1}_{n_l},\ \Psi(Y^{s_i})\beta^{s_i})$ are more similar. Again we can think of both terms as the reweighted set kernel [9, 23], which effectively measures the similarity between the source and the target distributions for both X and Y.

7.4 Experiments

7.4.1 Synthetic Data

Distribution Transfer on $P(y)$ **(Sec. 7.2.1)**. We consider the source distribution $1/2 \cdot \mathcal{N}(0,1) + 1/2 \cdot \mathcal{N}(5,1)$ with 400 samples, and the target distribution $1/2 \cdot \mathcal{N}(0,1) + 1/2 \cdot \mathcal{N}(4.5,1)$ with varying number of labeled samples. We solve Eq. 7.1 (denoted as "DT-KL" in blue, distribution transfer using KL) and Eq. 7.2 (denoted as "DT-MMD" in black, distribution transfer using MMD). Figure 7.4 (left) shows an example where "source" (red) represents the source distribution, and "true target" (green) represents the true target distribution. "label-target" (blue) refers to the estimated density based on a few labeled target samples. "weighted" (black) shows the reweighted source distribution by the proposed approaches.

Figure 7.4 (right) shows a comparison between the two proposed approaches (DT-KL and DT-MMD) and the baselines: (1) "source", which estimates $p(y)$ based on the source samples only; (2) "target", which estimates $p(y)$ based on the labeled target samples only; (3) "merged", which estimates $p(y)$ based on merged source samples and labeled target samples. The error metric (y-axis) is the L_2 error between the estimated density and the true density $\frac{1}{|Y^t|}\sum_{y^t \in Y^t}(\hat{p}(y^t) - p(y^t))^2$. The results are averaged over 20 experiments. We plot the mean and the standard error of the mean in all figures with empirical results in the following. For density estimation we use Gaussian kernel as the smoothing kernel K_h. The parameters (kernel width, λ to ensure that the kernel matrix is invertible) are chosen by cross-validation. We see that the proposed approaches effectively solve the distribution transfer problem, which yield a much smaller error than all the baselines.

Multi-Source Distribution Transfer on $P(y)$ **(Sec. 7.3.1)**. We consider the following 3 source distributions (1000 samples each): (1) $1/2 \cdot \mathcal{N}(0,1) + 1/2 \cdot \mathcal{N}(4,1)$; (2) $\mathcal{N}(2, 2^2)$; (3) $1/2 \cdot \mathcal{N}(0, 1.5^2) + 1/2 \cdot \mathcal{N}(4.5, 1.5^2)$. The target distribution is $1/2 \cdot \mathcal{N}(0,1) + 1/2 \cdot \mathcal{N}(4.5,1)$, with 30 observed labels. The source distributions are shown in Figure 7.5 (red). In the same figure, we show the estimated density based only on the labeled target (blue), and the true target distribution (green). We see that the proposed method (black) with 30 target labels can correct the source distribution (red) by shifting certain probability mass towards the distribution of the few labeled targets (blue).

In Fig. 7.6 (left), we observe that based on different sources we have different learning curves (L_2 error w.r.t. different number of labeled targets), which justifies the need for multi-source

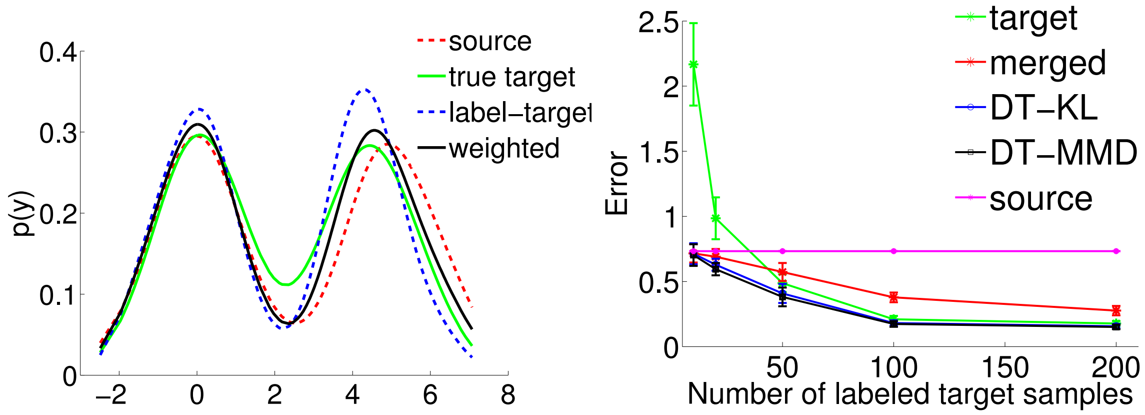


Figure 7.4: Distribution Transfer: an example with 30 target labels (top); comparison between the proposed algorithm and baselines (bottom)

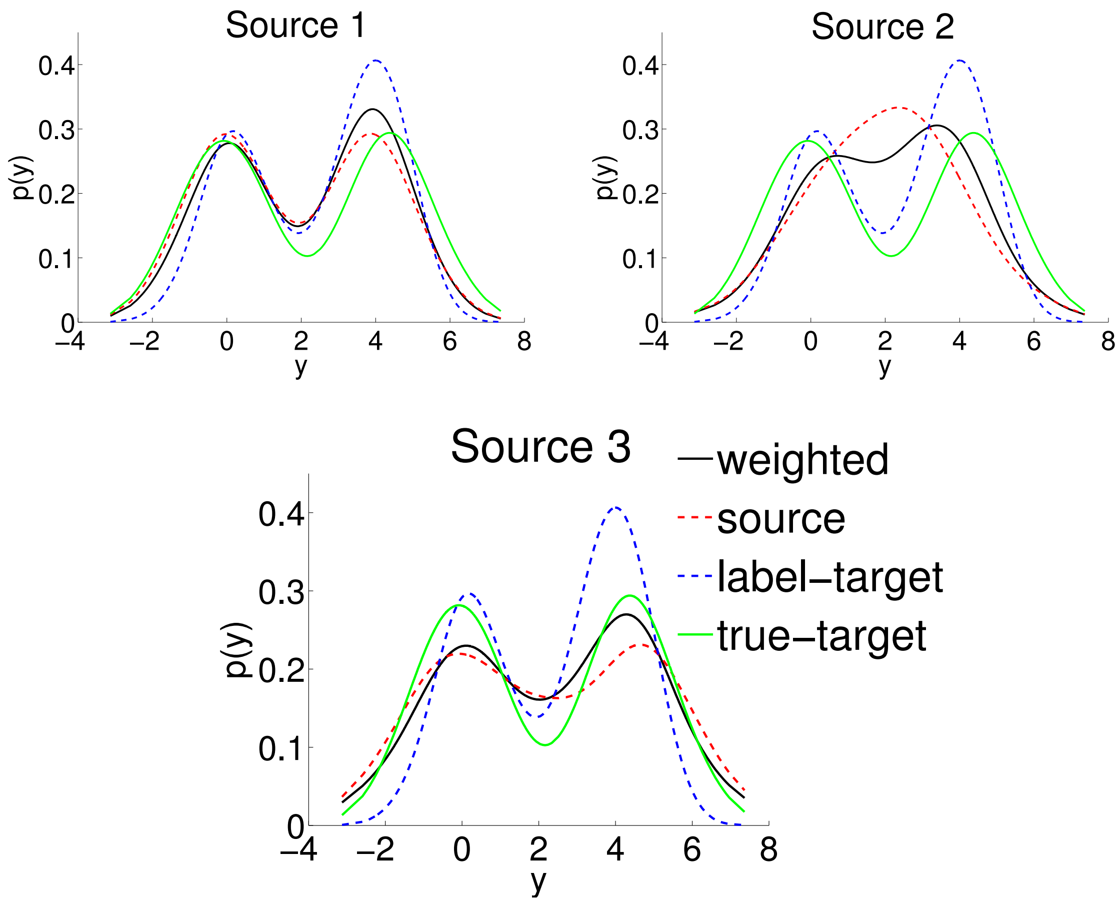
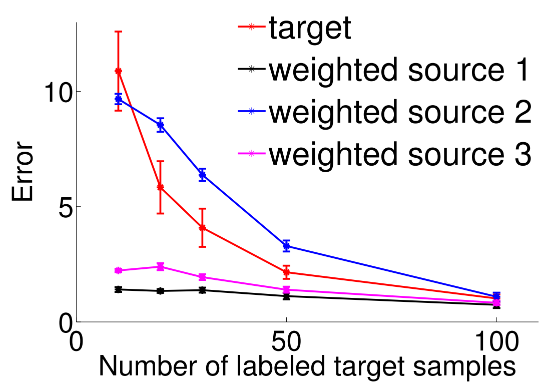


Figure 7.5: Example of source distributions and the weighted correction of target distributions under each source

distribution transfer. In Figure 7.6 (right), we show that the proposed multi-source distribution transfer (**multi-s**, Eq. 7.7) yields results that are very close to the results using the best source



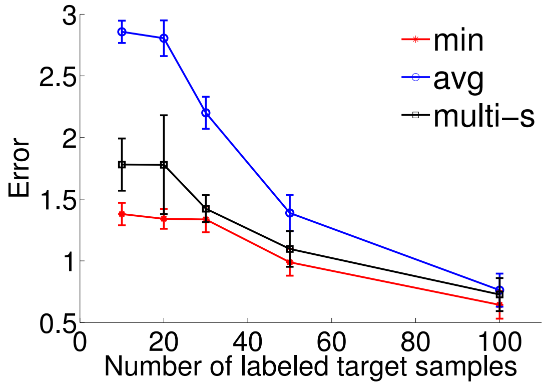
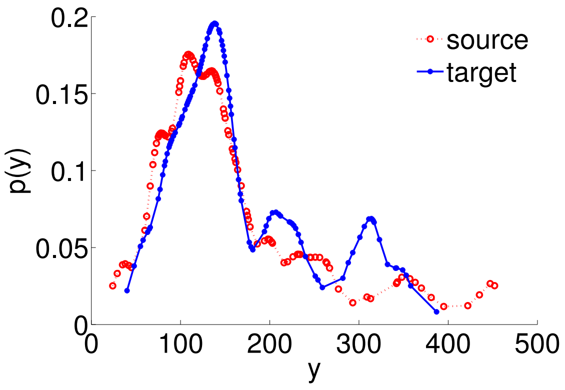


Figure 7.6: Single-source distribution transfer (top); Multi-source distribution transfer (bottom)

(**min**), which is unknown to the learner. The proposed method also yields much better results than the averaged combination of the sources (**avg**, i.e., $\alpha_i = 1/S$, while each source is still weighted by β^{s_i}).

7.4.2 Real Data



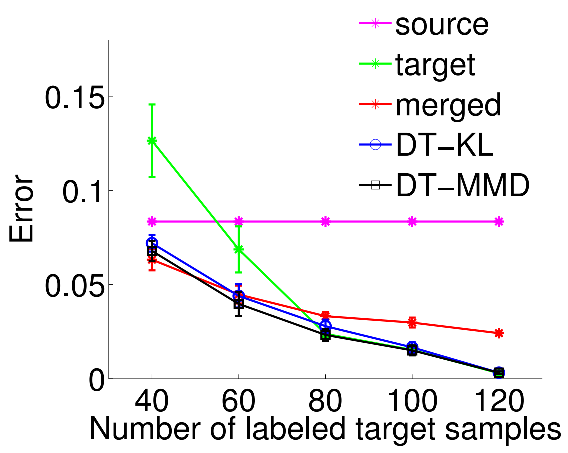


Figure 7.7: Distribution transfer: Example of source/target distributions on the AQI data (top); Comparison with baselines (bottom)

Predicting AQI distribution for Cities

Dataset Description. The dataset is an Air Quality Index (AQI) dataset [45] during a 31-day period from 107 cities. We aggregate the data by regions to form AQI distributions, and the task is to predict the AQI distribution for a target region based on the data from other regions (sources).

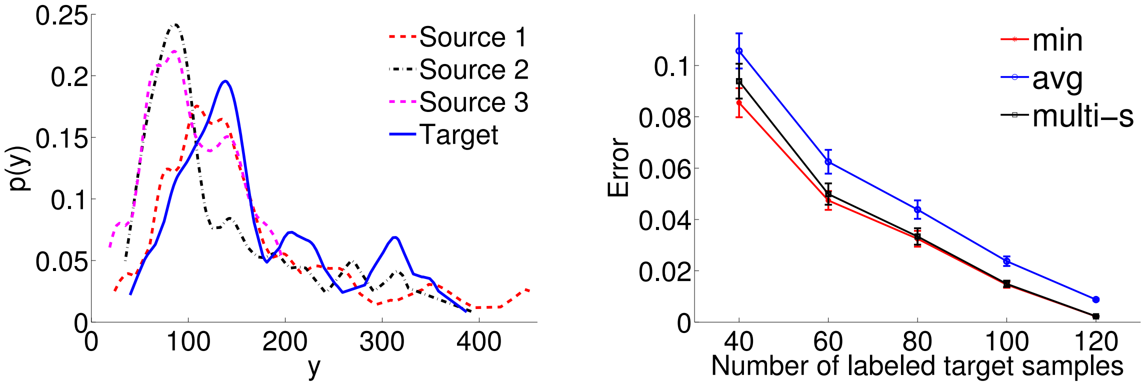


Figure 7.8: Multi-source distribution transfer: An example of multiple source distributions on the AQI dataset (left); Comparison with baselines (right)

Distribution Transfer on $P(y)$ **(Sec. 7.2.1)**. Figure 7.7 shows the results on the AQI dataset. The left figure shows an example of the source and the target distribution. From the right figure we can see that the proposed approaches (DT-KL or DT-MMD) effectively recover the target distribution, while outperforming all the baselines.

Multi-source Distribution Transfer on $P(y)$ **(Sec. 7.3.1)**. Figure 7.8 shows the comparison with baselines when transferring from multiple sources. The left figure shows an example of 3 source distributions and a target distribution (blue). The right figure shows the multi-source approach (black) outperforms average prediction and is again very close to the best prediction (min, unknown to the learner).

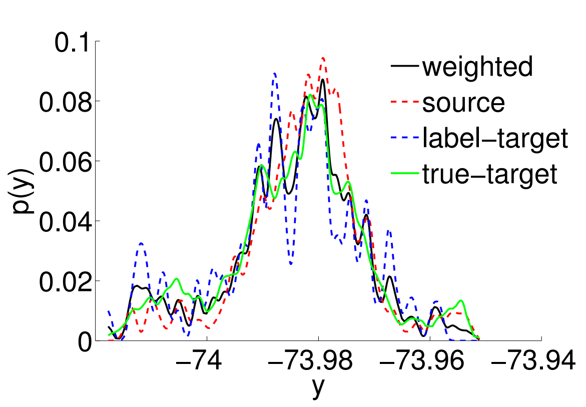
Predicting Drop-off Locations for Taxis

Dataset Description. The NYC taxi data[1] consists of pickup time, pickup location and dropoff location for taxis in New York city during the year of 2013. Each location is represented by a real-valued pair: (longitude, latitude).

Conditional Distribution Transfer (Sec. 7.2.2). The source/target are data from different time of the day, with pickup locations within a 0.008 radius of $x^* = (-73.985, 40.758)$ (longitude and latitude, which is near Time Square, New York city). The desired output distribution is the distribution of dropoff longitudes given the pickup location x^*.

Fig. 7.9 (left) shows an example of distribution transfer with 100 labeled target samples (source: 6-7am; target: 8-9am). The red line is the source distribution $P(y^s|X = x^*)$, and the green solid line approximates the true target distribution $P(y^t|X = x^*)$ by estimating from 1370 target samples. We observe that given the limited labeled target samples, $P(y^{tl}|X = x^*)$ (blue) is not a very good estimator of the true target distribution $P(y^t|X = x^*)$ (green). Using the conditional distribution transfer algorithm (Eq. 7.6), we obtain a more accurate estimation (black solid line) of the target distribution by utilizing a similar source distribution.

[1]http://www.andresmh.com/nyctaxitrips/



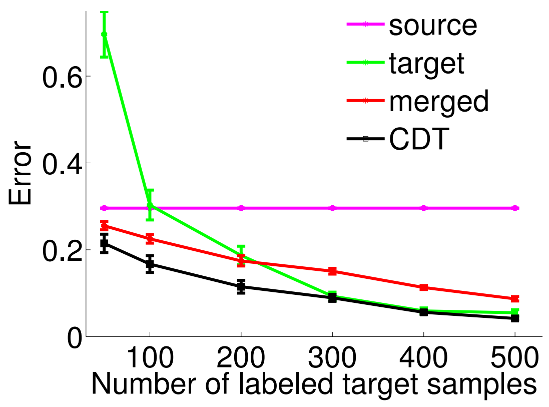
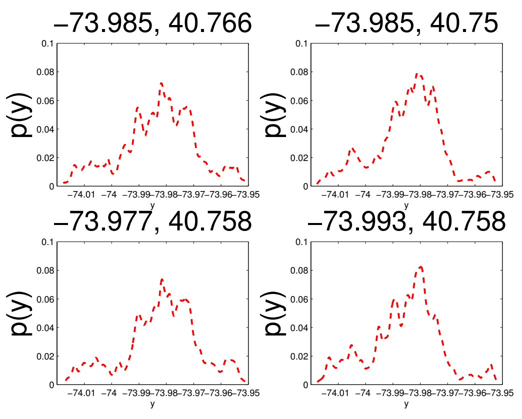


Figure 7.9: Example distributions on the taxi data (left); Single-source conditional distribution transfer (right)

Figure 7.9 (right) shows a comparison of the proposed approach (CDT, Eq. 7.6) with three baselines (source, target, and merged), averaged over 20 experiments. We observe that when the number of labeled target samples is small, prediction using the labeled target only is highly inaccurate, and utilizing the source clearly helps the prediction. When the number of labeled target gets larger, prediction using the source does not perform well due to the difference between the source and the target, and our proposed algorithm can effectively recover the target distribution by reweighting the source distribution.

Multi-source Conditional Distribution Transfer (Sec. 7.3.2). For multi-source distribution transfer with a fixed x^*, we use data samples (X, Y) with pickup locations within a 0.008 radius of $x^* = (-73.985, 40.758)$. Among those, the sources are data from different time frames of the day except 8-9am, and the target is the data within 8-9am. As we can see from Fig. 7.11 (left), the proposed **multi-source** prediction (black, Eq. 7.8) yields a smarter weighting than averaging over different sources (**avg**, blue), and it is very close to the optimal prediction (**min**, the minimum over all the sources in each iteration, shown in red), which is unknown to the learner.



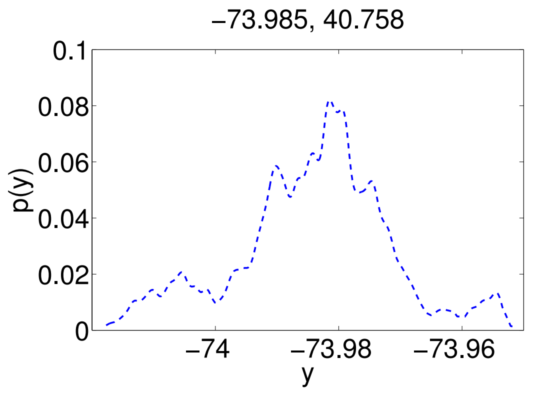


Figure 7.10: Example of sources $P(y^s)$ (left); target $P(y^t)$ (right)

Multi-Source distribution transfer for regions with different $P(X)$'s (Sec. 7.3.3). We use data from different pickup regions as different sources. The target (Fig. 7.10 (right)), consists of data samples within a 0.008 radius of $x^* = (-73.985, 40.758)$. In Fig. 7.10 (left), we plot examples of $p(y^s)$ from 4 pickup regions, each is within a 0.008 radius of x^{s_i} (the center x^{s_i} for each region is indicated by the title of each figure) which is close to x^* in different directions. Note that each source has almost the same distance to x^*, but the distribution $P(y^{s_i})$ varies. This suggests that combining multiple sources using only $P(X)$ (as in most previous work) is not sufficient for distribution transfer. In Fig. 7.11 (right), we compare the proposed method (**combined**, Eq. 7.9) with **avg** (average prediction). We also compare with **wDA**, a distribution weighted domain adaptation method [43], which considers only $P(X)$ and assume there is no shift in $P(Y|X)$. We optimize $D_\alpha(P(X^t)||\sum_i \lambda_i P(X^{s_i}))$ as suggested in Mansour et al. [43]. Since the original algorithm focuses on predicting the labels rather than distributions, we augment the weighted prediction $h_\lambda = \sum_i \frac{\lambda_i Q_i}{Q_\lambda} h_i$ using the distribution predicted by each source s_i as h_i. Further we compare with a recently proposed multi-source domain adaptation method **ms-DA** [12], with parameters $\theta, \gamma_A, \gamma_I$ chosen as described in their paper. We observe that the proposed method (black) yields the best results, which indicates that it is necessary to accommodate both the change in $P(X)$ and the distribution shift in $P(Y)$.

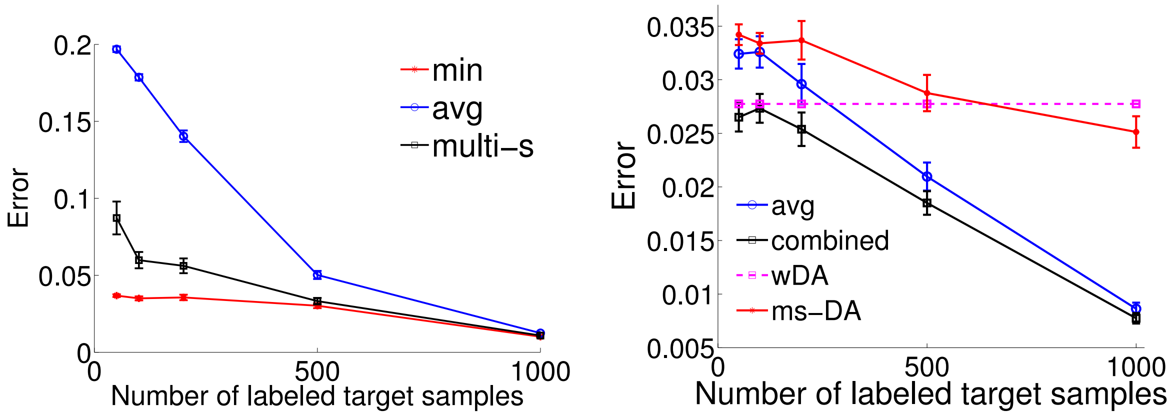


Figure 7.11: Multi-source conditional distribution transfer (left); multi-source transfer for regions with different $P(X)$'s (right)

7.5 Conclusion

In this chapter we propose methods for distribution transfer, which is an important problem in many real-world applications when a distributional output is desired. We show that a trade-off between source dependency and matching target yields good results on predicting the target distribution. In addition, we extend the framework to multi-source distribution transfer, and show that the discrepancy between each source and the target can affect the weighting for each source. Experiments on both synthetic and real data confirm the effectiveness of our proposed methods.

Chapter 8

Conclusion and Future Directions

8.1 Conclusion

This thesis focuses on active transfer learning under the model shift assumption. While the majority of previous work handles transfer learning by assuming only the marginal distribution changes across domains, we propose several algorithms for transfer learning in a much broader context, where the conditional distribution can change across domains, which is useful in many real-world applications with label-feature relation changing across datasets. We also show that the learning rate of these algorithms closely relates to the smoothness of change between the domains. In addition, for transfer learning with more than two domains, i.e., multi-task learning, we analyzed the algorithmic stability which again connects the learning rate of the algorithm to the smoothness of task relations. Further we extend the algorithm to learning on distributions.

Specifically, our main contributions include:

- We propose two transfer learning algorithms that allow changes in all marginal and conditional distributions with the additional assumption that the changes are smooth as a function of X. The first approach is based on conditional distribution matching, and the second is based on modeling the source/target task and the offset between using Gaussian Processes. We then propose an active learning method which yields a combined active transfer algorithm. Results on both synthetic datasets and a real-world dataset demonstrate the effectiveness of our proposed methods.
- We provide theoretical analysis for the propose algorithms for transfer learning under model shift. Unlike previous work on covariate shift, the model shift poses a harder problem for transfer learning, and our analysis shows that we are still able to achieve a similar rate as in covariate shift/domain adaptation, modified by the smoothness of the transformation parameters. We also show conditions when transfer learning works better than no-transfer learning.
- We extend the transfer learning algorithm so that it handles both support and model shift across domains. The algorithm transforms both X and Y by a parameterized-location-scale shift across domains, then the labels in these two domains are matched such that both transformations can be learned. Since we allow more flexible transformations than mean-

centering and variance-scaling, the proposed method yields better results than traditional methods. Results show that this additional flexibility pays off in real applications.

- For multi-task learning problems,we analyzed the stability of a family of multi-task learning algorithms which has many instantiations in the existing literature. We provide theory that connects the risk bounds for multi-task learning to the relation of tasks. We show that, by imposing a smooth relationship among tasks, we obtain favorable learning rates compared to learning tasks independently.
- We extend the transfer learning framework to distribution transfer. We propose a simple algorithm for transferring the distribution on the labels, and then we propose an algorithm for transferring the conditional distributions. We further extend the algorithm to multi-source distribution transfer.
- Experimental results on the synthetic data, as well as several real-world datasets, demonstrate the effectiveness of our proposed methods.

8.2 Future Directions

The number of algorithmic models people proposed is increasing and algorithms are becoming cheaper, while the cost of manual-labeling the data remains the same/decreases much slower, so it is becoming more and more important to design techniques that utilize/adapt existing models on new, scarcely-labeled datasets. We believe in the future transfer learning will remain an interesting direction to pursue, and how to relate existing models to the datasets that are best suited for that certain model remains an open question to answer. Specifically, for transfer learning, we believe these following directions are worth pursuing.

8.2.1 Transferability

We would like to answer fundamental questions like “when to transfer” and “what to transfer”. For example, if we are given any two datasets, there is no guarantee that we would benefit from transferring knowledge from one dataset to another. In the future we would like to develop algorithms that can automatically detect whether it is beneficial to transfer (e.g., we can test whether the marginal/conditional distributions in the two domains are similar if the datasets have the same feature/label space), and thus help make the “transfer or not” decision. In some cases it might be beneficial to transfer part of the knowledge from one dataset to another, and we would like to develop algorithms that can tell us which part to transfer.

The examples/real-datasets in this thesis are chosen to have some similarity across domains based on our domain knowledge, and we show that under certain conditions transfer learning achieves a favorable generalization bound than no-transfer at all. However, given two arbitrary datasets on which we do not have any domain knowledge, determining whether one dataset can be transferred to another requires additional computational cost, hence it is still unknown to us whether transfer learning helps reduce the total computational complexity in this case.

8.2.2 Transfer across different feature spaces

We would like to develop algorithms that allow any kind of transfer, even across different feature spaces. For example, if we have an image dataset and a text dataset, how to we handle the transfer between them when they have very different feature representations? One idea is to learn transitions for different feature representations (e.g., [17] models the path of linkage across feature spaces by a Markov chain and risk minimization), but how to achieve this automatically and robustly is still an open problem. Another idea is to project both feature representations into a shared latent space (e.g., [63] achieves transfer across feature spaces via spectral embedding), and how to find the best latent representation for both domains is still an interesting direction to pursue.

8.2.3 Transfer in deep networks

There has been an increasing interest in transfer learning/domain adaptation using deep neural networks. The main idea to increase transferability for the higher layers in the network is to match the distributions of the hidden representations in the higher layers [40], or to add extra layers to make the feature distributions more similar [21]. These methods, similar to the algorithms proposed for covariate shift, only focus on marginal distributions of the features (or higher-level features). With a few additional labels in the target domain, we can imagine using our proposed methods for conditional distribution matching (or even more flexible matching on both features and labels), thus potentially achieving a larger gain on deep networks.

8.2.4 Generalization bounds for active transfer

The generalization bounds we provided in this thesis is for transfer learning only, so it is possible to combine the generalization bounds for active learning with the transfer learning bound. Existing work [29] analyzes the bound on the label complexity of agnostic active learning but requires a finite VC dimension d for the hypothesis space, and [7] proposed an importance weighted active learning algorithm for binary classifiers, with sample complexity bounds related to $\ln |H|$ where H is the hypothesis space. It would be an interesting future direction to carry out stability analysis on the combined active transfer learning framework.

8.3 Discussions

8.3.1 Extension on SMS

In the SMS approach, it is possible to further relax the support containing constraint by only requiring a sub region of the source support to match a sub region of the target support, and we can add a penalty to prefer larger overlaps on the support. For the non-overlap part of the target support, we can rely on the smoothness of the target function space and the labeled target data.

8.3.2 Multi-source transfer combined with active learning

We can further combine active learning in the multi-source distribution transfer. After we obtained the initial reweights α on the source distributions, we have a prediction over the target data with the corresponding predictive covariance. Using this predictive covariance we can apply various active selection criteria to choose the next x^* and observe a sample $y^t \sim P(y^t|x^*)$. This new sample (x^*, y^t) allows the reweights α to be updated on the source distributions.

8.3.3 Connection with MDL

The minimum description length (MDL) principle is to find the best hypothesis for a given set of data that leads to the best compression of the data. In transfer learning, we can change the objective to be finding a transformation from the source to the target such that it has the minimum description length, which means we prefer a simpler model on the transformation.

8.3.4 Elastic deformations in distribution transfer

It is possible to apply elastic deformation (e.g., Terzopoulos' snakes) model in transfer learning on distributions. When transfer the distributions, the main objective is to keep the transferred distribution close to the target distribution, while not deviating too much from the source distribution. In this thesis we use KL divergence and MMD to measure the distance between distributions, but it is possible to plug in any other distance measure on the distributions to achieve this objective.

Bibliography

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. In *Journal of Machine Learning Research*, 2005. 2.4

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. 2.4

[3] A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2007. 2.4

[4] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. In *Machine Learning, 73, 3, 243-272, Special Issue on Inductive Transfer Learning*, 2008. 1.2, 6.6.2

[5] Julien Audiffren and Hachen Kadri. Stability of multi-task kernel regression algorithms. In *ACML*, 2013. 2.4, 6.2.4

[6] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. In *Journal of Machine Learning Research*, 2003. 1.2, 2.4, 6.6.2

[7] Alina Beygelzimer, Sanjay Dasgupta, and John Langford. Importance weighted active learning. In *ICML*, 2009. 8.2.4

[8] J. Blitzer, K. Crammer, A. Kulesza, F Pereira, and J. Wortman. Learning bounds for domain adaptation. In *NIPS*, 2007. 2.1.1

[9] Liefeng Bo and Cristian Sminchisescu. Efficient match kernels between sets of features for visual recognition. In *NIPS*, 2009. 7.3.2, 7.3.3

[10] E. Bonilla, K.M. Chai, and C. Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems (NIPS)*, 2008. 2.4

[11] Olivier Bousquet and Andre Elisseeff. Stability and generalization. In *JMLR*, 2002. 4.3.1, 6.2.3, 6.2.1, 6.2.3

[12] Rita Chattopadhyay, Jieping Ye, Sethuraman Panchanathan, Wei Fan, and Ian Davidson. Multi-source domain adaptation and its application to early detection of fatigue. In *KDD*, 2011. 2.1.1, 3.5.2, 4.6.2, 7.4.2

[13] Rita Chattopadhyay, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Joint transfer and batch-mode active learning. In *ICML 2013*, 2013. 1.1, 2.3

[14] J. Chen, L. Tang, J. Liu, and J. Ye. A convex formulation for learning shared structures from multiple tasks. In *ICML*, 2009. 2.4

[15] Corinna Cortes, Mehryar Mohri, Dmitry Pechyony, and Ashish Rastogi. Stability of transductive regression algorithms. In *ICML*, 2008. 2.4, 6.2.1

[16] Koby Crammer, Michael Kearns, and Jennifer Wortman. Learning from multiple sources. In *JMLR*, 2008. 2.1.1

[17] Wenyuan Dai, Yuqiang Chen, Gui rong Xue, Qiang Yang, and Yong Yu. Translated learning: Transfer learning across different feature spaces. In *NIPS*, 2008. 8.2.2

[18] Cuong B Do and Andrew Y. Ng. Transfer learning for text classification. In *Neural Information Processing Systems Foundation (NIPS)*, 2005. 2.1

[19] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *KDD*, 2004. 2.4, 6.2.2, 6.2.2

[20] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. In *JMLR*, 2005. 2.4

[21] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 2.6, 8.2.3

[22] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff Schneider, and Richard Mann. Bayesian optimal active search and surveying. In *ICML*, 2012. 1.1, 2.2, 3.4, 5.3

[23] T. Gartner, P.A. Flach, A. Kowalczyk, and A.J. Smola. Multi-instance kernels. In *ICML*, 2002. 7.3.2, 7.3.3

[24] J. G. D. Gooijer and D. Zerom. On conditional density estimation. In *Statistica Neerlandica, 57*, 2003. 2.5, 7.2.2

[25] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Scholkopf, and Alex Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems (NIPS)*, 2007. 1.4.1, 2.1

[26] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. In *NIPS*, 2007. 1.3, 4.1, 7.2.1, 7.2.1

[27] Arthur Gretton, Alexander J. Smola, Jiayuan Huang, Marcel Schmittfull, Karsten M. Borgwardt, and Bernhard. Schölkopf. Covariate shift by kernel mean matching. In *MIT Press*, 2009. 2.5, 7.2.1, 7.2.1

[28] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *ICML 2005*, 2005. 2.2

[29] Steve Hanneke. A bound on the label complexity of agnostic active learning. In *ICML*, 2007. 8.2.4

[30] Nils Lid Hjort and Ingrid K. Glad. Nonparametric density estimation with a parametric start. In *The Annals of Statistics*, 1995. 2.5

[31] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. In *Statistical Science, Vol. 14, No. 4, 382-417*, 1999. 4.5

[32] Michael P. Holmes, Alexander G. Gray, and Jr. Charles Lee Isbell. Fast nonparametric conditional density estimation. In *UAI*, 2007. 2.5

[33] Jiayuan Huang, Alex Smola, Arthur Gretton, Karsten Borgwardt, and Bernhard Schlkopf.

Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems (NIPS)*, 2007. 1.4.1, 1.3, 2.1, 2.5, 3.5.1, 3.5.2, 4.1, 5.4.1, 7.2.1, 7.2.1

[34] L. Jacob, F. Bach, and J. P. Vert. Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems (NIPS)*, 2008. 2.4

[35] Ming Ji and Jiawei Han. A variance minimization criterion to active learning on graphs. In *AISTATS 2012*, 2012. 1.1, 2.2, 3.4, 5.3

[36] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *Proc. 45th Ann. Meeting of the Assoc. Computational Linguistics, pp. 264-271*, 2007. 2.1

[37] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of gaussian processes: An exploration-exploitation approach. In *ICML 2007*, 2007. 2.2

[38] N.D. Lawrence and J.C. Platt. Learning to learn with the informative vector machine. In *ICML*, 2004. 2.4

[39] X. Liao, Y. Xue, and L. Carin. Logistic regression with an auxiliary data source. In *Proc. 21st Intl Conf. Machine Learning*, 2005. 2.1

[40] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015. 2.6, 8.2.3

[41] Yifei Ma, Roman Garnett, and Jeff Schneider. Sigma-optimality for active learning on gaussian random fields. In *Advances in Neural Information Processing Systems (NIPS)*, 2013. 1.1, 2.2, 3.4, 5.3

[42] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *NIPS*, 2008. 2.1.1, 4.6.2

[43] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009. 2.1.1, 4.1, 4.3.2, 7.4.2

[44] Andreas Maurer and Massimiliano Pontil. Excess risk bounds for multitask learning with trace norm regularization. In *JMLR*, 2013. 2.4

[45] Shike Mei, Han Li, Jing Fan, Xiaojin Zhu, and Charles R. Dyer. Inferring air pollution by sniffing social media. In *ASONAM*, 2014. 1.2, 3.5.2, 4.6.2, 6.6.2, 7.4.2

[46] Lilyana Mihalkova, Tuyen Huynh, and Raymond J. Mooney. Mapping and revising markov logic networks for transfer learning. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-2007)*, 2007. 2.1

[47] Alexandru Niculescu-Mizil and Rich Caruana. Inductive transfer for bayesian network structure learning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007. 2.1

[48] S. Nuske, K. Gupta, S. Narasihman, and S. Singh. Modeling and calibration visual yield estimates in vineyards. In *International Conference on Field and Service Robotics*, 2012. 1.2, 1.2

[49] Junier B. Oliva, Barnabas Poczos, and Jeff Schneider. Distribution to distribution regression. In *ICML*, 2013. 2.5

[50] Junier B. Oliva, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Fast

distribution to real regression. In *AISTATS*, 2014. 3.5.2

[51] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. In *TKDE 2009*, 2009. 1.1

[52] B. Poczos, A. Rinaldo, Singh A., and L. Wasserman. Distribution-free distribution regression. In *AISTATS*, 2013. 2.5

[53] A. Rahimi and B Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2007. 3.3.2, 6.4

[54] Piyush Rai, Avishek Saha, Hal Daume III, and Suresh Venkatasubramanian. Domain adaptation meets active learning. In *Active Learning for NLP (ALNLP), Workshop at NAACL-HLT 2010*, 2010. 1.1, 2.3

[55] Rajat Raina, Andrew Y. Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *Proceedings of the Twenty-third International Conference on Machine Learning*, 2006. 2.1

[56] Bernardino Romera-Paredes, Min Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil. Multilinear multitask learning. In *ICML*, 2013. 2.4

[57] M. Rosenblatt. Conditional probability density and regression estimates. In *Multivariate Analysis II, Ed. P.R. Krishnaiah, pp. 25-31. New York: Academic Press.*, 1969. 2.5, 7.2.2

[58] Ben-David S., Blitzer J., K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *NIPS*, 2006. 2.1.1

[59] Avishek Saha, Piyush Rai, Hal Daume III, Suresh Venkatasubramanian, and Scott L. DuVall. Active supervised domain adaptation. In *ECML*, 2011. 1.1, 2.3

[60] A. Schwaighofer, V. Tresp, and K. Yu. Learning gaussian process kernels via hierarchical bayes. In *Advances in Neural Information Processing Systems (NIPS)*, 2005. 2.4

[61] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: Active data selection and test point rejection. In *IJCNN*, 2000. 1.1, 2.2, 3.4, 5.3

[62] Xiaoxiao Shi, Wei Fan, and Jiangtao Ren. Actively transfer domain knowledge. In *ECML*, 2008. 1.1, 2.3

[63] Xiaoxiao Shi, Qi Liu, Wei Fan, and Philip S. Yu. Transfer across completely different feature spaces via spectral embedding. In *IEEE Transactions on Knowledge and Data Engineering*, 2013. 8.2.2

[64] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. In *Journal of Statistical Planning and Inference, 90 (2): 227-244*, 2000. 2.1, 2.5, 4.1

[65] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *ALT*, 2007. 7.2.1, 7.2.1

[66] Matthieu Solnon, Sylvain Arlot, and Francis Bach. Multi-task regression using minimal penalties. In *JMLR*, 2013. 2.4, 6.2.2, 6.2.2

[67] Le Song, Jonathan Huang, Alex Smola, and Kenji Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *ICML 2009*, 2009.

2.1, 3.3.1, 3.3.1, 4.2.1, 4.3.1, 7.2.2

[68] Qian Sun, Rita Chattopadhyay, Sethuraman Panchanathan, and Jieping Ye. A two-stage weighting framework for multi-source domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, 2011. 2.1

[69] Matthew Tesch, Jeff Schneider, and Howie Choset. Expensive function optimization with stochastic binary outcomes. In *ICML*, 2013. (document), 1.2

[70] Xuezhi Wang and Jeff Schneider. Flexible transfer learning under support and model shift. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 1.4.1

[71] Xuezhi Wang and Jeff Schneider. Generalization bounds for transfer learning under model shift. In *31st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015. 1.4.1

[72] Xuezhi Wang, Roman Garnett, and Jeff Schneider. Active search on graphs. In *KDD*, 2013. 1.1, 2.2, 5.3

[73] Xuezhi Wang, Tzu-Kuo Huang, and Jeff Schneider. Active transfer learning under model shift. In *the 31st International Conference on Machine Learning (ICML)*, 2014. 1.4.1, 5.4.1, 5.4.2

[74] Xuezhi Wang, Junier B. Oliva, Jeff Schneider, and Barnabás Póczos. Nonparametric risk and stability analysis for multi-task learning problems. In *25th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016. 1.4.2

[75] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with dirichlet process priors. In *Journal of Machine Learning Research*, 2007. 2.4

[76] Yi Yao and Gianfranco Doretto. Boosting for transfer learning with multiple sources. In *CVPR*, 2010. 2.1.1

[77] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 2.6

[78] S. Yu, V. Tresp, and K. Yu. Robust multi-task learning with t-processes. In *ICML*, 2007. 2.4

[79] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. 2.4

[80] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *ICML*, 2013. 2.1, 3.3.1, 3.5.1, 5.4.1

[81] Yi Zhang and Jeff Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In *NIPS*, 2010. 6.6.2

[82] Yu Zhang. Multi-task learning and algorithmic stability. In *AAAI*, 2015. 2.4, 6.2.3

[83] Leon Wenliang Zhong and James T. Kwok. Convex multitask learning with flexible task clusters. In *ICML*, 2012. 2.4

[84] Jiayu Zhou, Lei Yuan, Jun Liu, and Jieping Ye. A multi-task learning formulation for predicting disease progression. In *KDD*, 2011. 2.4, 6.2.2