COMPUTATIONAL STATISTICS: TIME SERIES AND DATA MINING

(Spine title: Plib)

(Thesis format: Monograph)

by

Shuang Ao

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO

School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....

Dr. Charles X. Ling

Supervisory Committee:

.....

Dr. W. J. Braun

.....

Dr. A. Bing

Examiners:

.....

Dr. Q. Ring

.....

Dr. W. Fing

.....

Dr. G. Hing

The thesis by

Shuang Ao

entitled:

Computational Statistics: Time Series and Data Mining

is accepted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

.....

Date

.....

Chair of the Thesis Examination Board

Acknowlegements

Abstract

This is a really silly abstract.

Keywords: Time series analysis, data mining

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

With the explosive image resources people uploaded every day, image recognition becomes a very hot topic and has draw many attentions in recent years. Every year, there are many inspiring results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). With development of recognition technology, many IT companies want to use image recognition techniques to serve their customers and some interesting applications have been developed, such as HowOld from Microsoft and Im2Calories from Google.

In order to successfully capture the diversity of different objects around us, many recognition models contain thousands or sometimes even millions of parameters and require large amount of training images to tune these parameters as well. Unfortunately, for some real applications, it is often difficult and cost to collect large set of training images. Moreover, most algorithms require that the training examples should be aligned with a prototype, which is commonly done by hand. In the real applications, collecting and fully annotating these images can be extremely expensive and could have a significant impact on the over cost of the whole system. As more companies pay attention to the individual customer experience, personalized service system become more important in recent years. For image recognition, one application of personalization can be learning the categories defined by individuals. The challenge of the self-defined categories learning comes from two aspects:

1. Flexibility: people could define any categories they want. The recognition algorithm should be more dynamic to adapt the new categories.

2. Rare learning examples: it is almost impossible to get abundant images from the self-defined categories and the algorithm should be able to learn the new categories from just a few examples.

On the other hand, recognition is one of the most important part of our human visual system. We can recognize various kinds of materials (apple, orange, grape), objects (vehicles,

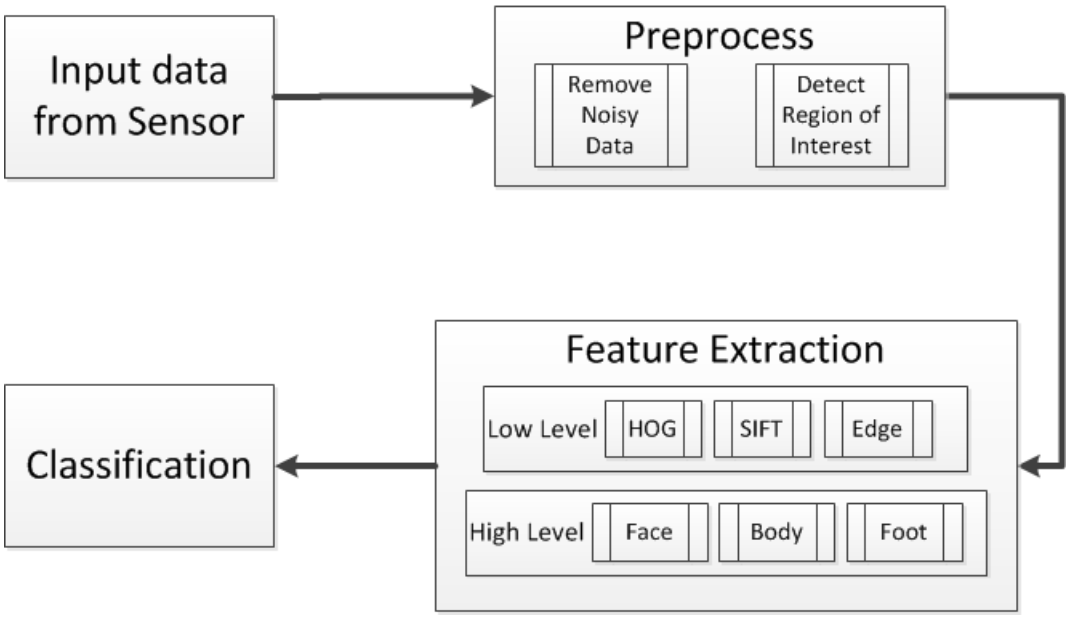


Figure 1.1: Major procedure for image recognition.

buildings) and natural scenes (forests, mountain). At the age of six, human can recognize about 10^4 object categories[6]. Our human can learn and recognize a new object with just a glance, which means we can capture the diversity of forms and appearances of a objects with just a handful examples. It could be ideal if we can find a way to train a new category with few examples.

1.1 Overview for Image Recogntion

In this section, we review the major procedures for image recognition. The procedure of the image recognition method consists of three parts: image preprocess, feature extraction and classification.

1.1.1 Preprocess

After receiving the raw data of a image from the sensor, preprocess generates a new image from the source image. This new image is similar to the source image, but differs from it considering certain apsects, e.g. the new image has smoother edge, better contrast and less noise. Here, some *pixel operations* and *local operations* are used to improve the contrast and remove the noise.

Another important operation of preprecess is segmentation according to the object, i.e. finding the region of interest. Images used for recognition should be aligned, making the target

object appear in the central of the image and remove those irrelevant area.

The result of preprocess has great impact on the final result of the recognition. Clear and noise free images can make the feature extraction more effective and significantly improve the final classification accuracy.

1.1.2 Feature Extraction

Feature extraction is a type of dimensionality reduction that efficiently represents interesting parts of an image as a compact feature vector. The feature vector is then used for either training the classifier or recognition. Therefore, feature extraction is the most important part for image recognition. The quality of the features extracted from a image have great impact on the recognition result. There are two major streams for feature extraction: one is the hand engineered method and the other one is representation learning method.

Hand Engineered Feature

Hand engineered features are low level and local features. Low level features are extracted according to some optical properties of an image. These features are low level / local features. There is a widely agreement that local features are an efficient tool for object representation due to their robustness with respect to occlusion and geometrical transformations [64]. Common low level hand engineered features include Histogram of Oriented Gradients (HOG) [19], Scale Invariant Feature Transform (SIFT) [43], Speeded Up Robust Features (SURF) [4], Local Binary Patterns (LBP) [47], and color histograms [7]. Feature descriptors obtain from these low level features refer to a pattern or distinct structure found in an image, such as a point, edge, or small image patch. They are usually associated with an image patch that differs from its immediate surroundings by texture, color, or intensity. What the feature actually represents does not matter, just that it is distinct from its surroundings. These low level features can be used directly for recognition. However, since they just represent certain local properties of an image and are not discriminative enough for recognition, discriminative high level features can be further learned by combining the low level features.

Representation Learning

Representation learning is mainly described by Deep Learning algorithms or Auto Encoders. The ideas is to learn a group of filters that are able to discern one category of images from the another category with some supervised or unsupervised algorithm. Learning representation from an image can start from either low level features (Auto Encoders) or raw pixels of an

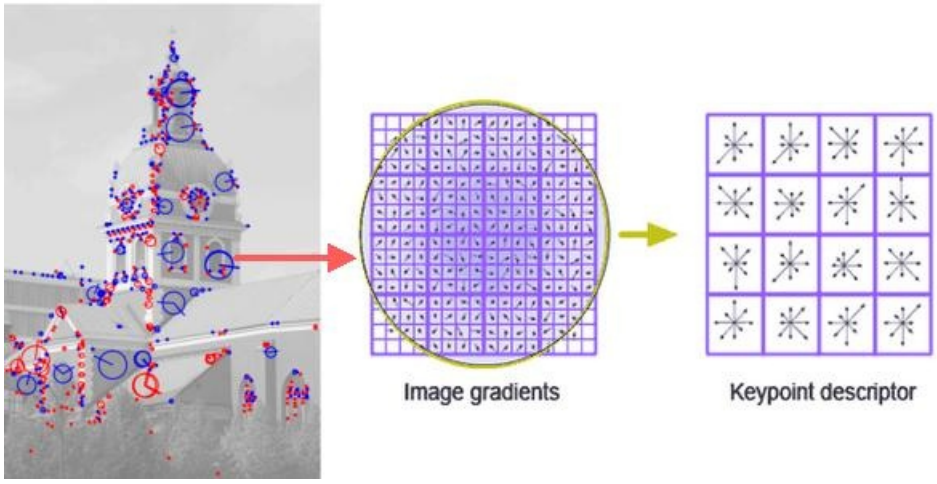


Figure 1.2: Feature extraction using SIFT.

image (Deep Learning).

Auto Encoders are widely used to combine different types of low level feature. The outputs of the Auto Encoders are some latent representations. These latent representations are learned from the given images that have lowest possible reconstruction error. Even though the high level representations from Auto Encoders are learned by minimizing the reconstruction errors, they are still not robust enough to handle all kinds of variance of the objects.

Currently, Deep Learning is the most popular approach for learning representations. It has been widely used for all kinds of image recognition tasks and achieved the state-of-the-art performance on some large scale image recognition tasks, such as ILSVRC and The PASCAL Visual Object Classes Challenge (PASCAL VOC).

Convolutional Neural Networks (CNN) is the most popular deep learning model for the image recognition tasks. The first deep CNN that had great success on image recognition is the LeNet proposed by Y.LeCan in 1989 [38]. Backpropagation was applied to Convolutional Neural networks with adaptive connections. This combination, incorporating with Max-Pooling and speeding up on graphics cards has become an important part for many modern, competition-winning, feedforward, visual Deep Learners.

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers.

- **Convolutional Layer** is the core building block of a Convolutional Network, and its output volume can be interpreted as holding neurons arranged in a 3D volume. The input to a convolutional layer is a $m \times m \times r$ image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has $r = 3$.

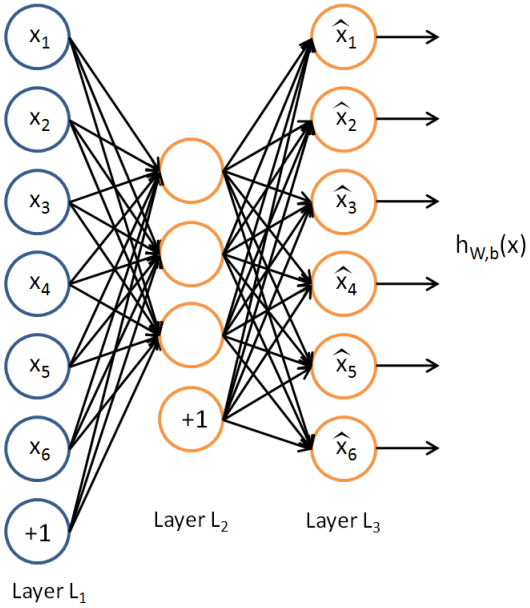


Figure 1.3: General Scheme of Auto Encoders. L1 is the input layer, possibly raw-pixel intensities. L2 is the compressed learned latent representation and L3 is the reconstruction of the given L1 layer from L2 layer. AutoEncoders tries to minimize the difference between L1 and L3 layers with some sparsity constraint.

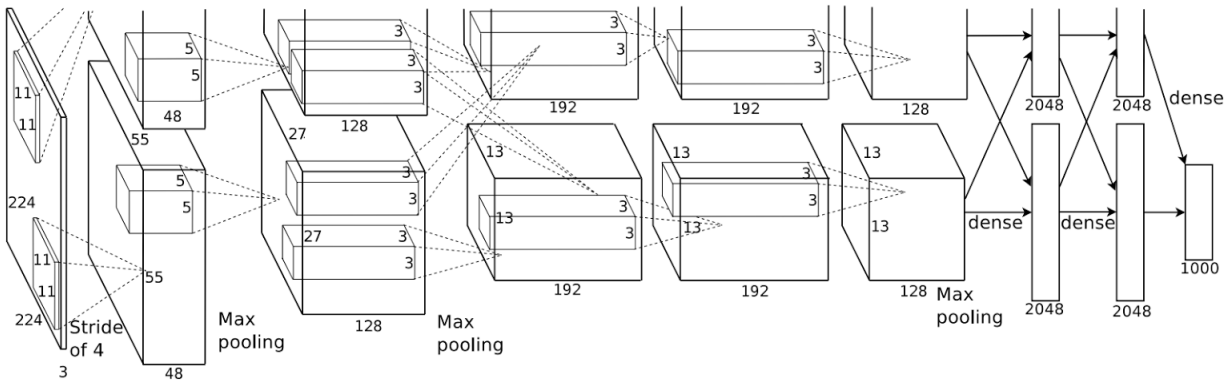


Figure 1.4: The architecture of ALEXNET (adapted from [34]).

- **Pooling Layer** is widely used in all kinds of CNN architecture for dimensional reduction and computational efficiency. In general, two kinds of pooling strategy, Max Pooling and Average Pooling, are commonly used in CNN architecture. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice [45] [57]. Max Pooling is been widely used in all kinds CNN architectures [9] [67].

- **Fully Connected (FC) Layer** has full connections to all activations in the previous layer, as seen in regular Neural Networks. Recent work show that FC layers with Rectified Linear Units and Dropout can greatly improve the learning speed as well as avoid overfitting for deep CNNs [25] [46].

Two typical techniques, Rectified Linear Units (ReLUs) for Activation and DropOut, are also extensively used in CNN. 4Recent work show that FC layers with Rectified Linear Units and Dropout can greatly improve the learning speed as well as avoid overfitting for deep CNNs [25] [46].

- **Rectified Linear Units (ReLUs) for Activation** is used as the activation function in CNN and can be written as

$$\sum_{i}^{N} \sigma(x - i + 0.5) \approx \log(1 + e^{x}) \tag{1.1}$$

  where $\sigma(x)$ is the sigmoid function. In practice, Rectified Linear Units use the function

$$f(x) = \log(1 + e^{x}) \approx \max(x, 0) \tag{1.2}$$

  Recent work show that FC layers with Rectified Linear Units and Dropout can greatly improve the learning speed as well as avoid overfitting for deep CNNs [25] [46].

- **DropOut**. In FC layer, nodes are connected to each other and this leads to a large number of parameters. Generally, larger number of parameters means more power for Neural Networks and more easily prone to overfitting. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training [55]. Technically, DropOut can be interpreted as adding extra noise into the training procedure. Without actually adding noise, FC layer with DropOut is tolerant of higher level of noise (20 %-50%). Randomly dropping out the nodes, for any node in FC layer, it can't rely on the other nodes to adjust its result. By eliminating the co-adaptation of hidden units, DropOut becomes a technique that can be applied to any general domain and improve the performance of neural nets.

In addition, a good CNN model is a reasonable combination of these layers and techniques and contains million or even billions of parameters, which is particularly suitable for those difficult large scale image recognition tasks. However, due to its size of parameters, the CNN model should be trained from scratch with a huge image database to avoid overfitting.

Classification

After extracting feature representation from the images, a classifier should be used to train a recognition model as well as for predicting the new coming images. A supervised model is always used for training the recognition model. Discriminative classifier such as Support Vector Machine (SVM) is widely used as the classifier for recognition [17]. As we mentioned before, in order to capture different variances of the images for one category, the size of the feature representation for a image is usually very large. In order to avoid overfitting, the size of the training set should be at least the same size of the size of the feature representation as well. Some classifiers such as Bayesian method or decision tree require to consider the correlations between each feature and the class labels and suffer from the large feature dimension. However, Discriminative model are more convenient for training and can be effectively optimized with stochastic gradient descent which is suitable for very large training set [8]. we will introduce linear method in Section 2.1.

1.2 Our Scenario (to be refined)

1.2.1 Importance of learning self-defined categories

Nowadays, more and more companies provide individual service for their clients. Personalized recommendation system has been widely used in many electronic commerce, such as Amazon and eBay. The requirement of personalized service is growing every year. Personalized system means it is possible to handle the variance of the request from individuals. For image recognition, many interesting applications have been proposed. However, unlike other personalized system, it is difficult to train personalized recognition system for individuals. There could be enormous variances of one object and it is always difficult for a model to capture these variances.

Even though, the state-of-the-art recognition system can do as well as a human, its great success is achieved based on the fact that millions of labeled training images are used for training. Selecting large amount of images for the new categories is always a tedious job. Moreover, an important property of self-defined category images is that the source of the data

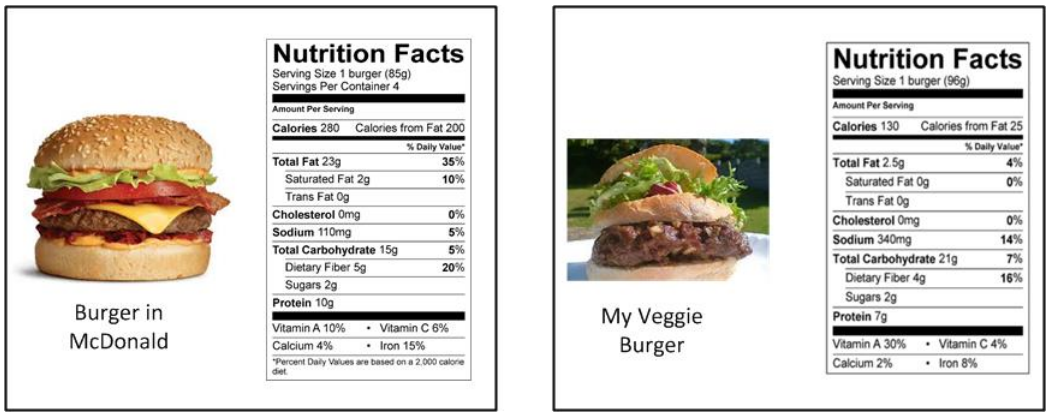


Figure 1.5: Different nutrition facts between the burgers in McDonald and home-made.

is inherently scarce. Therefore, it is not possible to obtain abundant training data. For example, users of Google's im2calories can track their nutrition of every meal by taking a pictures of their foods via image recognition techniques. The system firstly check the category of each food item and find their nutritions in the database. However, the existing model can only recognize the general food categories which means it is not possible for a user to track the nutrition of his/her daily home-made meals. These home-made foods can be similar to some existing food categories (home-made veggie burger can be similar to burger in McDonald), but they may have different nutritions. This scenario can be applied to learning any exclusive category in our life. Therefore, a model that can learn these self-defined categories can be important.

Our human is good at learning the new objects. For our human, all the information acquired is stored in our memory. These information are organized according to the properties. When we see a new concept, we don't treat it isolated, but connect it to certain previous knowledge we stored in our memory. By comparing a new concept with the organized information in our memory, we can capture the property of a new concept effectively. When referring to visual tasks, several examples can be given to show this cognitive ability. For instance, when we describe the animal "zebra", we would probably say that: zebra is a horse with while and black striped coats. People who never see a zebra could instantly have a rough idea of a zebra.

This means to learn a new category effectively, we should be able to make use of the gained knowledge instead of learn it from scratch. This process is commonly refer to as transfer learning. Traditional machine learning methods work under the common assumption: training data and testing data are drawn from the same feature space and same distribution. When considering adding classifying the data from the new categories, the distribution of the data changes and a new model should be rebuilt from newly collected data.Those data from the original distribution is called source data and data from the new distribution is called target

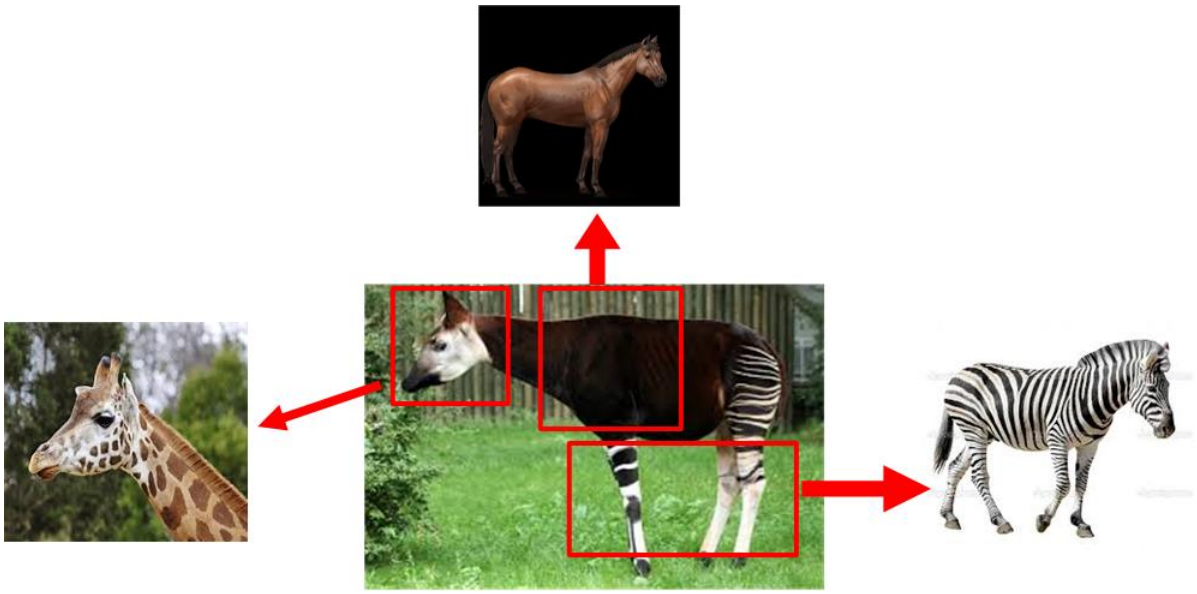


Figure 1.6: Multi-source category case: an okapi can be roughly described as the combination of a body of a horse, legs of the zebra and a head of giraffe.

data. Transfer learning is used to utilize the source knowledge from the source data to help training the new model to classify the target data.

In this thesis, we try to use transfer learning approaches to solve the problem of learning self-defined categories. We can find many examples in knowledge engineering where transfer learning does benefit the learning process [48]. This implies that, by leveraging the learned knowledge properly, we can learn the self-defined categories with a few examples. In this thesis, we also assume that the self-defined categories are not isolated and similar to certain existed categories. Therefore, we can use leverage the knowledge from those existed categories to help us learn the self-defined categories. Then, we split the self-defined categories into two groups according to their relationship to the exited categories:

- One source category, a category that is very similar to one existed category. For example, my veggie burger can be very similar to general burger and less similar to other categories.

- Multi-source category, a category that doesn't have an explicit similarity to one category but share some properties with several existed categories (see figure 1.6 for instance).

Due to different properties of these two groups, we should design different strategies to adapt them.

1.2.2 Assumption of the source knowledge

In transfer learning, the source knowledge can be presented in two different ways [48]:

- Instance transfer learning. Even though the source data can not be re-used directly, certain part of the data can be used incorporating with a few labeled target data to train the model.

- Parameter transfer learning. Instead of utilizing the source data, parameter transfer learning approaches re-use the parameters of the model trained from source data (called the source model).

In this thesis, we try to explore a method to learn self-defined categories via the parameter transfer learning approach and try to utilizing the source data in a hard way, by assuming that the source data is access prohibited and we can only access the trained model from the source data. This assumption is made based on the following two facts: (1) In some situations, we may not be able to access the source data and only the model trained from the source data is available. There are many credential datasets. Therefore, it is not always possible to fully access the source data. (2) The source data could be very large and it could be tedious to determine which part of the data could benefit the transfer learning. On the other hand, the trained model from the source data can be as informative as the source data itself. For example, the information extracted from the support vectors (SVs) of a SVM model trained from a dataset could be as much as the information of the whole dataset. Some results from recent work also show that it is possible to obtain a good model by even just utilize the source models and learn new categories from a few examples [24] [59] [60].

In this section, we discuss the scenario of the learning self-defined category problem. We conclude that learning self-defined categories can be achieved by transfer learning. We also make an assumption that we can only access the model trained from the source data and the source data itself is prohibited. In the next section, we discuss the challenges of our problem.

1.3 Challeges

In this section, we discuss several major challenges that we may encounter when solving the learning self-defined category problem. To solve the learning self-defined category problem, there are the following major problems:

1. What's the classifier and parameter we should use to train the model that can transfer the knowledge from the source data effectively?

2. How are the knowledges transferred from the source task to target one and control the amount of the knowledge that should be transferred?

3. How to guarantee the result of transfer model?

The first challenge is to determine the classifier we use for the source data as well as the target data. There are two criterion for us to choose the classifier: (1) This classifier should be popular and has been used in many tasks. We would expect our method to be general enough and can be applied to many different scenarios. (2) Because we can only access the trained source model, this source model should be able to restore as much information as its training data to make sure that there is enough knowledge to be leveraged for the transferred model. Then we have to choose the parameters of the model to be used for transfer learning. As we discussed above, the parameter should be informative enough to represent the source data.

The second challenge of out problem is to determine the way of the knowledge transferred from source model to target one. To learn a new category by transfer learning, according to our human experience, we would firstly select several known categories that are similar to the new one and then describe the new category as a combination of these known ones (see figure 1.6). It is worthy to note that despite of the knowledge from the source models, we will also extract some new knowledge from the examples of the new category. This indicates that how to combine those knowledge from learned categories (e.g. the parameters of the source models) and the knowledge from the new category is the key for our task.

The last challenge is to guarantee the performance of our transfer method. A baseline criterion is that after leveraging the knowledge from the source model, the performance of the transfer model to distinguish the self-defined categories will be improved or at least won't get worse when there is little useful knowledge in the source model. This is an important criterion for transfer learning. We should always expect to leverage knowledge of the source model to help training the target model. However, inappropriately leveraging the knowledge of the source models would decrease the performance of the target model. For example, we could never expect to obtain a good model to distinguish apple and other fruits by relying too much on the knowledge of distinguishing human and animal.

Chapter 2

Related Work

In this chapter, we review some previous work related to ours. In Section we review.....

2.1 Classifiers for Image Recognition

In our scenario, we have to face the problem of image recognition. Due to the large dimension of the feature representation for each image as well as the size of training image, manual classification is hopeless. As we mentioned in Section 1.1, there should always be a recognition model to distinguish the objects from different categories automatically trained by supervised learning.

In supervised image recognition, we train a recognition model from a set of training images along with their labels provided. The labels are predefined in a category space. Thus, the task of image recognition is to classify each image as one predefined category. If there are only two categories, this recognition task is called binary classification. For the task recognizing the objects from more than two categories, the recognition task is called multi-class recognition [2] [34].

Here, we give a formal definition of the scenario of binary classification and for multi-class scenario, we can decompose the multi-class learning task into a set of binary scenarios by training a binary classifier for each class, e.g. One-VS-Rest strategy (see figure 2.1)[50] [63].

The binary scenario for classification can be defined as follow: given a dataset from domain $\mathcal{X}\times\mathcal{Y}$ where $\mathcal{X}$ is the input feature representation and $\mathcal{Y}$ is the binary label set $\{1,-1\}$ (for some classifier, $\{1,0\}$ is also used). We usually use the label 1 to denote the examples belong to one certain category and -1 to denote examples not belong to that category. We assume that the training image set $D_{train} = \{(x_i, y_i)\} \subset \mathcal{X}\times\mathcal{Y}$ and the test image set $D_{test} = \{(x_i^t, y_i^t)\} \subset \mathcal{X}\times\mathcal{Y}$ are given and separated from each other. Each pair (x_i, y_i) denotes the input feature representation

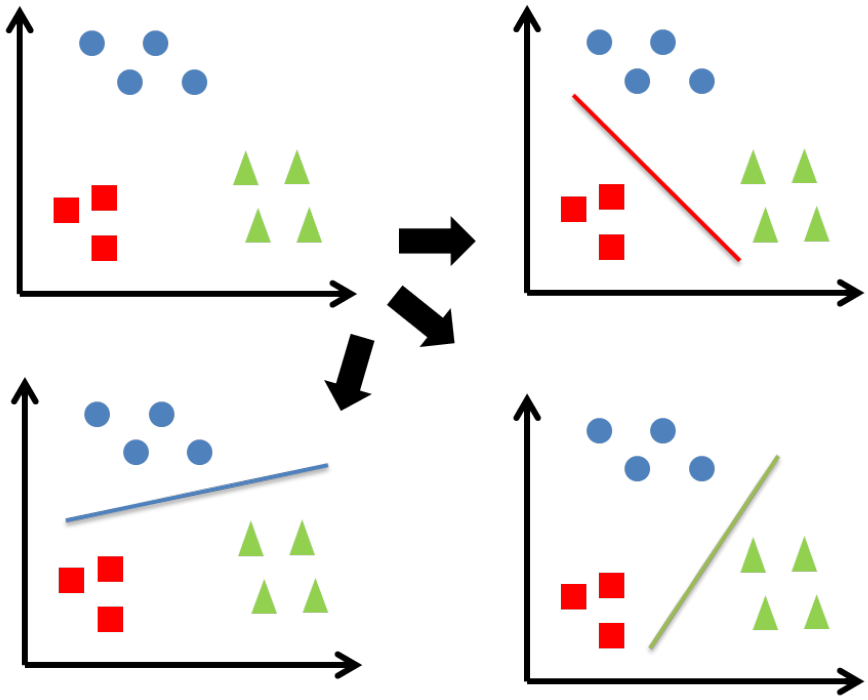


Figure 2.1: One-vs-Rest strategy for multi-class scenario. A three classes problem can be decomposed into 3 binary classification sub-problems.

x_i and its corresponding label y_i for the *ith* image in the both set. Our goal of the classification problem is to learn a decision function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from the training set D_{train} such that f can achieve good performance on both D_{train} and D_{test}.

In the next subsection, we first review.....

2.1.1 Linear Method: Softmax Classifier

In this subsection, we will introduce a widely used linear classifier softmax classifier. Linear classifier is commonly used as the classification model for image recognition. A linear classifier achieves this by making a classification decision based on the value of a linear combination of the input feature representations of a image. A linear classifier consists of two parts: a score function and a loss function. The score function maps the input data into the class scores and the loss function that quantifies the agreement between the predicted scores and the ground truth labels. Linear classifier often work very well when the number of dimensions of the input is large. Therefore, it is widely used as the classifier for image recognition.

Softmax classifier is widely used for image classification especially on Neural Networks and Convolutional Neural Networks [38]. Softmax classifier (also called multinomial logistic

regression) is a generational form of logistic regression for the multi-class scenario. As logistic regression can only handle the binary classification scenario, Softmax classifier adapts the one-vs-rest strategy where several logistic regression models are trained for each class.

Given a training set $\{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, we assume the label $y_i \in \{1, 0\}$ and the input feature $x_i \in \mathcal{R}^n$. For each binary logistic regression model, The score function takes the form:

$$f_w(x) = \frac{1}{1 + \exp(-w^T x)} \tag{2.1}$$

and the parameters w are optimized to minimize the following loss function:

$$l(w) = -[\sum_{i=1}^{m} y_i \log f_w(x_i) + (1 - y_i) \log(1 - f_w(x_i))] \tag{2.2}$$

For softmax classifier, it is used to handle the multi-class classification problem and suppose there are N classes. Therefore, y can take from N different values $\{1, 2, 3, ..., N\}$ instead of just two. For a given test example x, the score function estimate the probability $P(y = n|x)$ for each value of $n = 1, 2, 3, ..., N$, i.e. estimate the probability that each score assigns the input x to the N classes associated to the N different possible values. Thus, the score function will output a N-dimensional vector providing N estimated probabilities whose sum of elements is 1. The N dimensional output of the score function can be generated according to the following form:

$$f_w(x) = \begin{bmatrix} P(y = 1|x; w) \\ P(y = 2|x; w) \\ ... \\ P(y = n|x; w) \end{bmatrix} = \frac{1}{\sum_{j=1}^{N} \exp\left(w^{(j)T} x\right)} \begin{bmatrix} \exp\left(w^{(1)T} x\right) \\ \exp\left(w^{(2)T} x\right) \\ ... \\ \exp\left(w^{(N)T} x\right) \end{bmatrix} \tag{2.3}$$

Here $w^{(1)T}, w^{(2)T}, ..., w^{(N)T}$ are the parameters of the softmax classifier model. The term $\frac{1}{\sum_{j=1}^{N} \exp\left(w^{(j)T} x\right)}$ is called the normalization term so that the N distributions sum to 1.

To optimize the parameters $w^{(1)T}, w^{(2)T}, ..., w^{(N)T}$, the cross-entropy loss used for the softmax classifier is defined as:

$$J(w) = -\left[\sum_{i=1}^{l} \sum_{k=1}^{N} \ell\{y_i = k\} \log \frac{\exp\left(w^{(k)T} x_i\right)}{\sum_{j=1}^{N} \exp\left(w^{(j)T} x_i\right)}\right] \tag{2.4}$$

Here ℓx is the 0-1 loss function:

$$\ell\{x\} = \begin{cases} 1 & x \text{ is true} \\ 0 & x \text{ is false} \end{cases} \tag{2.5}$$

It is noted that Eq (2.4) is a generalized form of Eq (2.2). Minimizing Eq (2.4) can be interpreted as minimizing the negative log likelihood of the correct class, which is equivalent to performing Maximum Likelihood Estimation (MLE) [31].

The minimum of $J(w)$ can be obtained by gradient descent method while taking the gradient:

$$\nabla J(w^{(n)}) = -\sum_{i=1}^{l}\left[x_i\left(\ell\{y_i = n\} - \frac{\exp\left(w^{(k)T}x_i\right)}{\sum_{j=1}^{N}\exp\left(w^{(j)T}x_i\right)}\right)\right] \tag{2.6}$$

2.1.2 Kernel Method: Support Vector Machines

In this subsection, we will review another widely used discriminant classifier, Support Vector Machine (SVM) [17]. SVM is another classifier that has been adoptive in many image recognition tasks [14] [53] [67]. In this thesis, we also use a classifier based on SVM. We will give a detailed description of SVM.

As we mentioned before, the linear classifier consists of two parts: the score function and loss function. SVM classifier can be divided into two categories based on their score function: linear SVM that uses a linear discriminant function and kernel SVM that uses kernel function. Kernel SVM can be considered as a extension version of linear SVM where kernels are used for calculate the scores of the inputs. Another difference for between linear and kernel SVM is linear SVM can be solved on the primal problem while kernel SVM is mostly optimized on its dual [17] [54].

First, we will introduce the linear SVM. Linear SVM uses the simplest representation of a score function by taking the linear combination of the input vector:

$$f(x) = w^T x + b \tag{2.7}$$

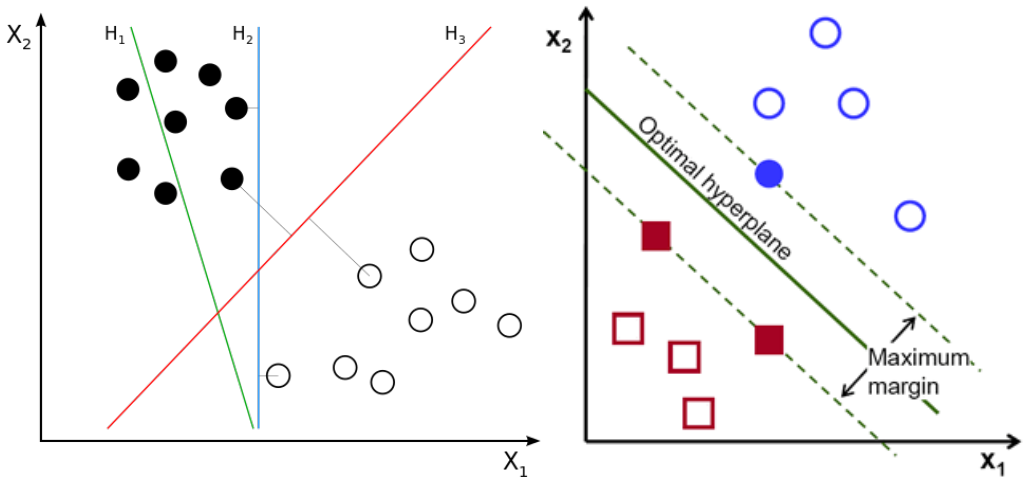
where w is called the weight vector and b is called the bias. In a binary scenario, for the input example x and the class labels $y \in \{c1, c2\}$, x is assigned to class $c1$ if $f(x) \geq 0$ and $c2$ otherwise. Therefore, the corresponding decision surface is defined by $f(x) = 0$. For two points x_1 and x_2 lie on the decision surface, we have $f(x_1) = f(x_2) = 0$. Then we can have $w^T(x_1 - x_2) = 0$ and hence the weight vector w is orthogonal to every point lying within the decision surface, i.e w determines the orientation of the decision surface. Similarly, if x lies on the decision surface, the normal distance from the origin to the decision surface is given by:

$$\frac{w^T x}{||w||} = -\frac{b}{||w||} \tag{2.8}$$

Therefore, we can see that, the location of the decision surface is determined by the bias b.

Hard Margin SVM

Hard margin SVM is used to find the optimal solution for the data sets that are linear separable. Given a set of n training points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where $y_i \in \{1, -1\}$, we expect to



(a) Different sparating hyperplanes (b) Max-Margin hyperplanes

Figure 2.2: Support Vector Machine

find a decision surface that can separate the data from two classes. However, when the data from these two classes can be linearly separated, there could be several decision surfaces that can separate the data. The idea of SVM is to choose the maximal margin decision surface so that the distance between these two classes is as large as possible (called maximal margin hyperplane). For example, in figure 2.2(a), $H2$ and $H3$ are two candidate decision surface that can separate the data. However, $H3$ is has the largest distance to all the data from two classes and SVM will choose $H3$ as the optimal hyperplane.

The optimal hyperplane can be found by minimizing the following objective function:

$$\begin{aligned} \min \quad & \|w\|^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 \quad \text{for all } 1 \leq i \leq n \end{aligned} \tag{2.9}$$

Soft Margin SVM

In real world application, most data are not linear separable. Therefore, SVM introduce the concept of slack variable to handle this situation. Slack variable is defined as:

$$\xi_i = hinge(x_i) = \max(0, 1 - y_i(w^T x_i + b)) \tag{2.10}$$

The value of slack variable is 0 if the example x_i lies on the correct side of the margin. For those data on the wrong side, its value is proportional to the distance from the correct margin.

Therefore, to find the parameters of hyperplane, i.e. weight vector w and bias b, soft-margin

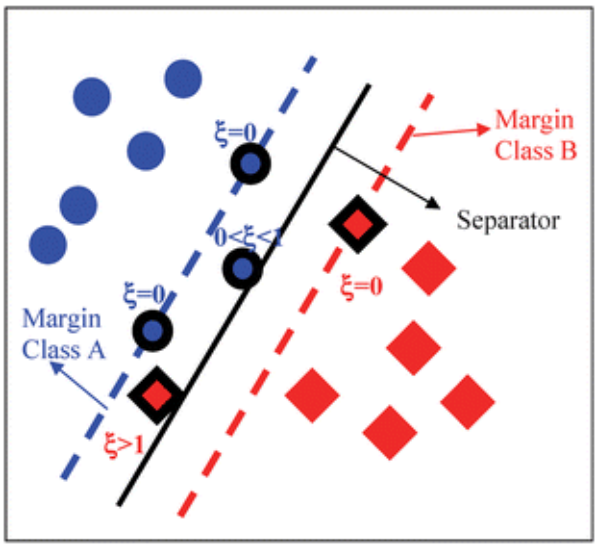


Figure 2.3: Slack variables for soft-margin SVM

SVM minimize the following loss function:

$$\begin{aligned} \min \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i}^{n}\xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \text{for all } 1 \leq i \leq n \end{aligned} \tag{2.11}$$

The objective function is called primal of SVM. A stochastic sub-gradient descent can be used to find the optimal solution for eq. (2.11) effectively [54].

Kernel SVM

Linear soft-margin SVM works well when number of features is larger than number of training examples. However, when the size of the training example is larger than the features, kernel SVM (such as Gaussian Kernel) with proper parameters outperforms linear SVM [32].

The idea of kernel was first introduced into pattern recognition by Aizerman *et. al.* [1]. When the size of the training examples are significantly larger than the dimension of the input features and the distribution become more complex, these data can not be easily separated by a straight line in the feature space. Instead of obtaining the optimal hyperspace in the input feature space, kernel SVM tries to map the inputs into a high-dimensional feature spaces and find the optimal hyperplane in the high-dimensional feature spaces. Given a feature space mapping $\phi(x)$, the score function for kernel SVM can be re-written as:

$$f(x) = w^T \phi(x) + b \tag{2.12}$$

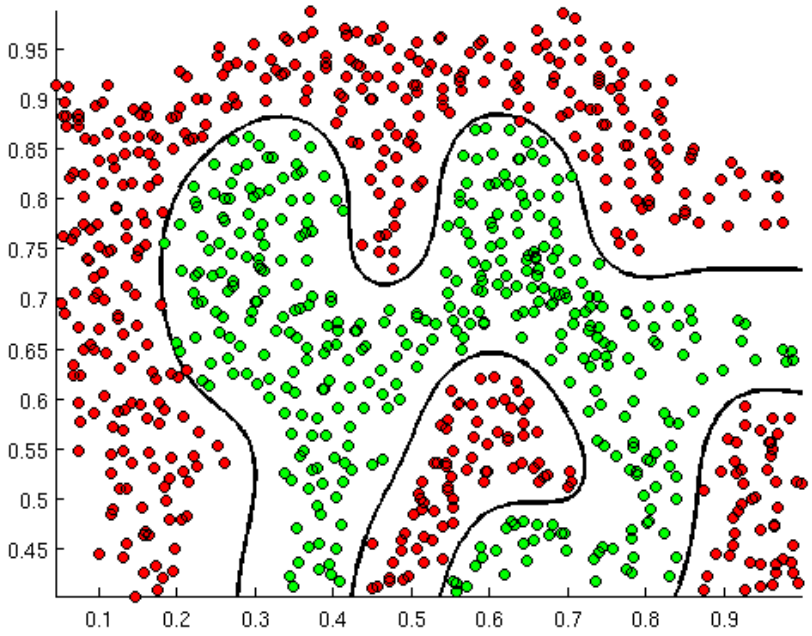


Figure 2.4: The hyperplane of SVM with RBF kernel for non-linear separable data.

From eq (2.12) we can see that, when we take the identity mapping $\phi(x) = x$, the kernel SVM becomes a linear SVM. Therefore, the linear SVM can be considered as a special case of kernel SVM where identity mapping is used as the kernel. The loss function of kernel SVM is almost identical to eq. (2.11) except for replacing the term x with $\phi(x)$:

$$\begin{aligned} \min \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i}^{n}\xi_i \\ \text{s.t.} \quad & y_i(w^T\phi(x)_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \text{for all } 1 \leq i \leq n \end{aligned} \tag{2.13}$$

By introducing the Lagrangian term to the primal (2.13) and some transformation, we obtain the dual of kernel SVM function:

$$\begin{aligned} \max \quad & \sum_{i}^{n}\alpha_i - \sum_{i}^{n}\sum_{j}^{n} y_i y_j \alpha_i \alpha_j \phi(x_i)\phi(x_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\lambda} \\ & \sum_{i}^{n} y_i \alpha_i = 0 \quad \text{for all } 1 \leq i \leq n \end{aligned} \tag{2.14}$$

We can obtain the solution of (2.14) by Sequential Minimal Optimization (SMO) [49] or Dual Coordinate Descent [27].

There are several advantages of using SVM as the classifier:

- **Generalization ability.** SVM provides good generalization ability by maximizing the margin between the examples of the two classes. By setting the proper parameters and generalization grade, SVM can overcome some bias from the training set. Therefore, SVM is able to make correct prediction for unseen data. This ability can be very useful for image recognition as there is no image dataset that can cover all the transformation of the objects. Moreover,the idea of soft-margin makes it robust against noisy data.

- **Kernel transformation.** By introducing the non-linear transformation of the input, SVM can model complex non-linear distributed data. The kernel trick can greatly improve the computational efficiency.

- **Unique solution.** The objective function of SVM is convex. Compared to other methods, such as Neural Networks, which are non-convex and have many local minima, SVM can deliver a unique solution for any given training set and can be solved with efficient methods, like sub-SGD [54] or SMO [49].

To summarize, in this section, we have briefly reviewed 2 types of classifier for image recognition, namely softmax classifier and SVM. Softmax classifier is typically used as the last layer of deep neural network. SVM is a more general method method for classification task. Moreover, The generalization ability of SVM classifier can reduce the bias from the training data. Therefore, in our work, we choose SVM as our classifier for image recognition.

2.2 Transfer Learning

As we mentioned before, we use the transfer learning method to learn the self-defined categories. In this section, we review some problems transfer learning faces and the popular methods to learn new categories in transfer learning.

Traditional machine learning algorithms try to build the classifiers from a set of training data and apply to the test data with the same distribution to the training data. In contrast, transfer learning attempts to change this by transfer the learned knowledge from one or several tasks (called source tasks) to improve a related new task (called target task).

Transfer learning can be categorized into 3 sub-settings: *inductive transfer learning, transductive transfer leanring* and *unsupervised transfer learning* based on the different situations of the source and target domains and tasks. [48]. We compared the differences of these three sub-categories and show them in Table 2.2. Recalling our learning self-defined category problem, we already obtained the source models from labelled source data and our goal is to utilize

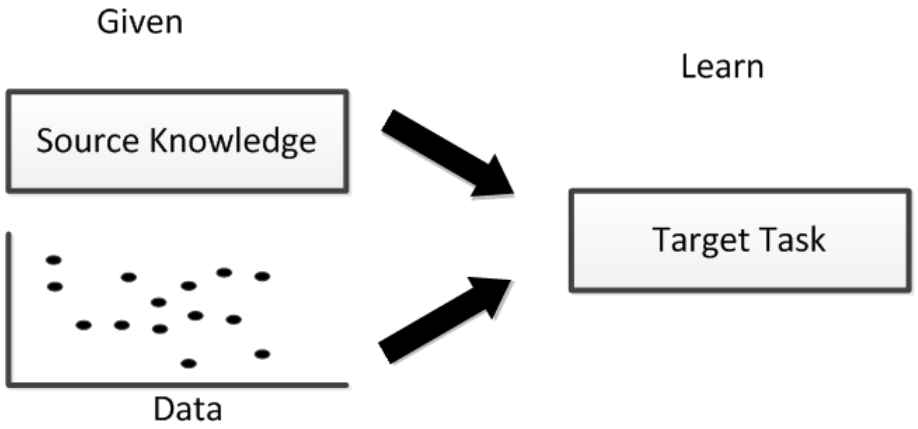


Figure 2.5: Apart from the standard machine learning, transfer learning can leverage the information from an additional source: knoweldge from one or more related tasks.

the knowledge from the source models and help us to train the models from a small set of labelled examples, i.e. examples from our self-defined categories. From Table 2.1 and 2.2 we can see that, our problem should be classified as inductive transfer learning. More specifically, it belongs to the multi-task transfer learning.

Table 2.1: Relationship between traditional machine learning and different transfer learning settings

Learning settings		Source target domain	Source target task
Traditional machine learning		the same	the same
Transfer learning	Inductive transfer learning	the same	different but related
	Unsupervised transfer learning	different but related	different but related
	Transductive transfer learning	different but related	the same

2.2.1 Inductive Transfer Learning

In inductive transfer learning, we try to utilize the knowledge from the source task to help learning the target task. Here we give a formal definition of inductive transfer learning [48]:

Definition (Inductive Transfer Learning) Given a source domain D_s and the source learning task T_s, a target domian D_t and the target learning task T_t, inductive transfer learning aims to help improve the learning of the target task function $f_t(\cdot)$ in D_t using the knowledge from D_s and T_s where $T_s \neq T_t$.

Table 2.2: Various settings of transfer learning

	Related areas	Source data	Target data
Inductive transfer learning	self-taught learning	unlabeled	labeled
	multi-task learning	labeled	labeled
Transductive transfer learning	domain adaptation, Sample selection bias	labeled	unlabeled
Unsupervised transfer learning		unlabeled	unlabeled

According to the type of the source knowledge comes from, inductive transfer learning can be split into 3 major streams: instance transfer, feature representation transfer and parameter transfer.

The core idea of instance transfer learning is to select some useful data from the source task to help learning the target task. Dai et al. [18] propose a method (called TrAdaBoost) that can select the most useful examples from the source task as the additional training examples for the target task. These useful examples are iteratively re-weighted according to the classification results of some base classifiers. Jiang et al. [29] propose a method that can ignore the "misleading" examples from the source data based on the conditional probabilities on the source task $P(y_t|x_t)$ and target task $P(y_s|x_s)$. Liao et al. [40] propose a active learning method that selects and labels the unlabeled data from the target data with the help of the source data. Ben-David et al. [5] provide a theoretical analysis the lowest target test error for different source data combination strategies when the source data is large and target training set is small.

Feature representation transfer aims to find a good feature representations to reduce the gap between the source and target domains.According to the size of labeled examples in the source data, feature representation transfer consists of two approaches: supervised feature construction and unsupervised feature construction. When the source data are labeled, supervised feature transfer learning is used to find the feature representations shared in related tasks to reduce the difference between the source and target tasks. Evgeniou et al. [22] propose a method that can learn sparse low-dimension feature representations that can share between different tasks. Jie et al. [30] reconstruct the feature representations for the target data by using the outputs of the source models as the auxiliary feature representations. In unsupervised feature representation transfer learning, Daume III [20] proposes a simple feature reconstruction method for both source and target data so that source and target data are triple argumented and a SVM model is trained on both source and target data.

Parameter transfer assumes that there should be some parameters or prior distribution of the hyperparameters in the individual models of related tasks. Most of the approaches are

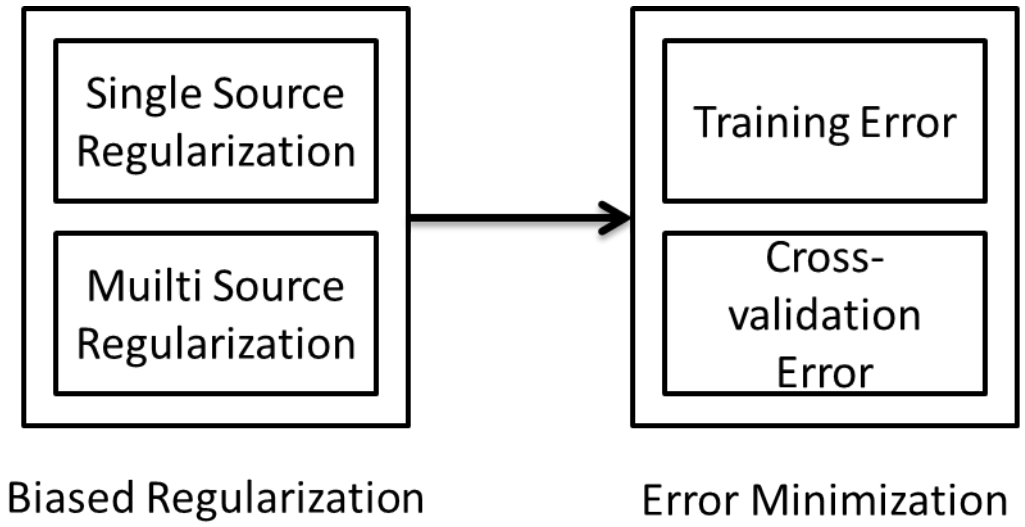


Figure 2.6: Two steps for parameter transfer learning. In the first step multi-source and single source combination are usually used to generate the regularazation term. The hyperplane for the transfer model can be obtained by either minimizing training error or cross-validation error on the target training data.

designed under the multi-task learning scenario. Therefore, in this thesis, we also focus on the parameter transfer approach to leverage the knowledge from the source data. In parameter transfer learning, there are three major frameworks: a regularization framework, a Bayesian framework and a neural network framework.

- Regularization framework: In the regularization framework, some researchers propose to transfer the parameters of the SVM following the assumption that the hyperplane for the target task should be related to the hyperplane of the source models. Evgeniou et al. [23] propose an idea that the hyperplane of the SVM for the target task should be separate into two terms: a common term shared over tasks and a specific term related to the individual task. Inspired by this idea, some researchers propose different strategies to combine these two terms for transfer learning [2] [59] [69]. Most of these work contains two steps. In the first step, a SVM objective function with a biased regularization term for the target model is build. Then another objective function is build to reduce the empirical error of the target model on the target data.

- Neural network framework. In neural network framework, the idea is to use the parameters of a CNN pre-trained from a very large dataset as an initialization to reduce the bias when the target data is small. Yosinski et al. [70] show that the high level layer parameters are more related to a specific task while the low level layer ones are more

general and transferable. This framework is widely used for image recognition task. By re-using and fine-tuning the parameters of some layers in the pre-trained model, the bias of the target task can be greatly alleviated [13] [26] [71] [72].

- Bayesian framework. In Bayesian framework, one or several posterior probabilities of the source data or parameters of the source model can be used to generate a prior probability for the target task. With this prior probability, a posterior probability for the target task can be obtained with the target data. Li et al. [24] use a prior probability density function to model the knowledge from the source and modify it with the data from target to generate posterior density for detection and recognition. Rosenstein et al. [52] use hierarchical Bayesian method to estimate the posterior distribution for all the parameters and the overall model can decide the similarities of the source and target tasks.

2.2.2 Avoid Negative Transfer

In transfer learning, for a given target task, the performance of a transfer method depends on two aspects: the quality of the source task and the transfer ability of the transfer algorithm. The quality of the source task refers to how the source and target tasks are related. If there exists a strong relationship between the source and target, with a proper transfer method, the performance in the target task can be significantly improved. However, if the source and target tasks are not sufficiently related, despite of the transfer ability of the transfer algorithm, the performance in the target task may fail to be improved or even decrease. In transfer learning, negative transfer refers to the degraded performance compare to a method without using any knowledge from the source [48]. For example, we can use a teacher-student diagram to illustrate the procedure of transfer learning. The student (target model) would like to learn the new knowledge (target task) with the assistance of a teacher (source knowledge). If the teacher can provide helpful knowledge (related knowledge), the student can learn the new knowledge very quickly (positive transfer). If the teacher can only provide useless knowledge, the student could not learn the new knowledge effectively or even get confused (negative transfer).

Another important aspect that affect the learning performance is the transfer ability of the algorithm used for the target task. An ideal transfer algorithm would be able to produce positive transfer on related tasks while avoiding negative transfer on unrelated tasks. However, in practice, it is not easy to achieve these two goals simultaneously. Approaches that can avoid negative transfer often bring some affects on positive transfer due to their caution. On the other hand, approaches using aggressive transfer strategies often have little or no protection against negative transfer [61]. Even though voiding negative transfer is an important issue in transfer learning, how to avoid negative transfer has not been widely addressed [44] [48]. Previous

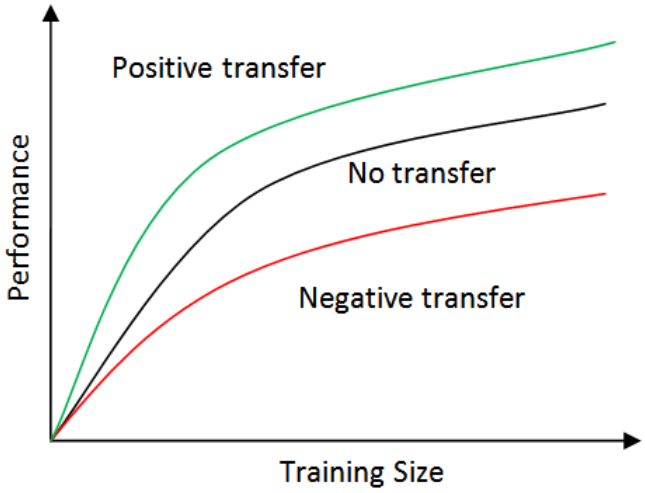


Figure 2.7: Positive transfer VS Negative transfer.

work show suggest that negative transfer can be alleviated through 3 approaches [61]:

- Rejecting unrelated source information. A important approach to avoid negative transfer is to recognize and reject unrelated and harmful source knowledge. The goal of this approach is to minimize the impact of the unrelated source, so that the transfer model performs no worse than the learned model without transfer. Therefore, in some extreme situation, the transfer model is allowed to completely ignore the source knowledge. Torrey et al. [62] propose a method using advice-taking algorithm to reject the unrelated source knowledge. Rosenstein et al. [52] present an approach that use naive Bayes classifier to detect and reject the unrelated source.

- Choosing correct source task. When the source knowledge come from more than one candidate source, it is possible for the transfer model to select the best source knowledge of the candidates. In this scenario, leverage the knowledge from the best candidate may be effective against negative transfer as long as the best source knowledge is sufficiently related. Talvitie et al. [58] propose a method that can iteratively evaluate the candidate sources through a trail-and-error approach and select the best one to transfer. Kuhlmann et al. [35] construct a kernel function from certain sources for the target task by estimating the bias from a set of candidate sources whose relationship to the target task is unknown.

- Measuring task similarity. To achieve a better transfer performance, it is reasonable for a transfer method to transfer the knowledge from multiple sources instead of just choosing a single source. In this approach, some methods try to involve all the source knowledge

without considering the explicit relationship between the source and target. The other methods try to model the relationships between the source and target tasks and use the information as a part of their transfer methods which can significantly reduce the affect of negative transfer. Bakker et al. [3] propose a method to provide guidance on how to avoid negative transfer by using clustering and Bayesian approach to estimate the similarities between the target task and multiple source tasks. Tommasi et al. [60] construct the transfer model by using some transfer parameters to measure the relationships between each source and the target tasks and the transfer parameters are optimized by minimizing the cross-validation error of the transfer model. Similar approaches can be found in [30] [37]. Kuzborskij et al. [36] provide some theoretical analysis of transfer learning and show that regularized least square SVM with truncation function and leave-one-out cross-validation for source task measurement can reduce negative transfer even though the training data of the target task is relatively small.

Here we can see that most of the previous approaches focus on measuring the similarity of the source and target tasks, i.e. try to assign the most related source tasks to the target one through various of metrics and use aggressive transfer algorithm to exploit the source knowledge. Just a few work [59] [36] addressed the problem that a sophisticated transfer algorithm should be designed to better exploit the source knowledge as well as avoid negative transfer. Therefore, in this thesis, we mainly focus on how to design a better transfer algorithm for transfer learning while certain source knowledge is assigned.

To summarize, in this section, we reviewed 3 types of inductive transfer learning approaches and discussed how to avoid the negative transfer in transfer learning. In our problem, as we have the assumption that the source data is access prohibited, we are not able to use instance transfer and feature transfer approaches. Therefore, we propose a transfer method based on parameter transfer approach. To reduce the affect negative transfer, in our transfer method, we adopt several ideas from the work mentioned above.

2.3 Summary

Chapter 3

Learning Single Source Categories

In this chapter, we study the problem that transfer the knowledge from a single source to recognize single source self-defined categories. As we mentioned in Chapter 1.3, we defined two kinds of self-defined categories: single-source and multi-source self-defined categories. We first introduce our strategy to handle single source categories and the multi-source can be solved using the similar strategy.

To deal with this problem, we firstly propose a data argumentation strategy for single source category. We use the transfer parameters to control the amount of knowledge transferred from the source. To estimate the transfer parameters, we propose a novel objective function that be solved effectively and avoid negative transfer. With extensive experiments on the public benchmark MNIST, we empirically show that our method can effectively transfer the knowledge from a single source and avoid negative transfer while other benchmark transfer methods suffer.

The rest of this chapter is organized as follow:

3.1 Issues in Transfer Learning

The motivation of transfer knowledge between different domains is to apply the previous information from the source domain to the target one, assuming that there exists certain relationship, explicit or implicit, between the feature space of these two domains [48]. Technically, previous work can be concluded into solving the following three issues: what, how and when to transfer [60].

What to transfer. Previous work tried to answer this question from three different aspects: selecting transferable instances, learning transferable feature representations and transferable model parameters. Instance-based transfer learning assume that part of the instances in the

source domain could be re-used to benefit the learning for the target domain. Lim et al. proposed a method of augmenting the training data by borrowing data from other classes for object detection [41]. Learning transferable features means to learn common feature that can alleviate the bias of data distribution in target domain. Recently, Long et al. proposed a method that can learn transferable features with deep neural network and showed some impressive results on the benchmarks [42]. Parameter transfer approach assumes that the parameters of the model for the source task can be transfered to the target task. Yang et al. proposed Adaptive SVMs by transferring parameters by incorporating the auxiliary classifier trained from source domain [69]. On top of Yang's work, Ayatar et al. proposed PMT-SVM that can determine the transfer regularizer according to the target data automatically [2]. Tommasi et al. proposed Multi-KT that can utilize the parameters from multiple source models for the target classes [60]. Kuzborskij et al. proposed a similar method to learn new categories by leveraging over the known source [37].

When and how to transfer. The question *when to transfer* arises when we want to know if the information acquired from previous task is relevant to the new one (i.e. in what situation, knowledge should not be transfered). *How to transfer* the prior knowledge effectively should be carefully designed to prevent inefficient and negative transfer. Some previous work consists in using generative probabilistic method [21] [66] [73]. Bayesian learning methods can predict the target domain by combining the prior source distribution to generate a posterior distribution. Alternatively, some previous max margin methods show that it is possible to learn from a few examples by minimizing the Leave-One-Out (LOO) error for the training model [37] [59]. Previous work shows that there is a closed-form implementation of LOO cross-validation that can generate unbiased model estimation for LS-SVM [12].

Our work correspond to the context above. In this chapter, I propose SMITLe based on parameter transfer approach with LS-SVM. I address my work on how to prevent negative transfer when the source data is not accessible. Compared to other works, I propose a novel strongly convex objective function for transfer parameters estimation. I show that SMITLe can converge at the rate of $O(\frac{\log(t)}{t})$. By optimizing this objective function, SMITLe can autonomously adjust the transfer parameters for different prior knowledge. I theoretically and empirically show that, without any data distribution assumption, the superior bound of the training loss for SMITLe is the loss of a method learning directly (i.e. without using any prior knowledge). Experiment results show that when the prior knowledge hurts the transfer procedure, SMITLe can avoid negative transfer by ignoring the unrelated prior knowledge autonomously. Extensive experiments also show that when the prior knowledge is very related (positive transfer), my method can outperform other methods by exploiting the prior knowledge greatly.

3.2 Related Work

In this part, We will introduce the framework my work based on and some related previous work. We first introduce the basic principle of LS-SVM. Based on LS-SVM, several transfer methods are introduced, including A-SVM, PMT-SVM, Multi-KT and MULTIpLe. In these methods, the first two can only adopt the knowledge from single source model and the rest can adopt the knowledge from multiple ones.

3.2.1 LS-SVM Classifier

Least Square SVM is proposed is least squares versions of support vector machines (SVM), which are a set of related supervised learning methods that analyze data and recognize patterns [56]. By replacing the hinge loss in classical SVM with L2 loss, LS-SVM classifier is obtained by reformulating the minimization problem as:

$$\begin{aligned} \min \quad & L_{LSSVM} = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}\varepsilon_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{3.1}$$

The primal Lagrangian for this optimization problem given the unconstrained minimization problem can be written as:

$$L(w, b, \alpha, \varepsilon) = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}{\varepsilon_i}^2 - \sum_{i=1}^{l}\alpha_i\{wx_i + b + \varepsilon_i - y_i\} \tag{3.2}$$

Where $\alpha = [\alpha_1, ..., \alpha_l]^T$ is the vector of Lagrange multipliers. The solution to minimise this problem is give by:

$$\begin{bmatrix} K + \frac{1}{C}\mathrm{I} & 1 \\ 1^T & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{3.3}$$

Where $K \in R^{l\times l}, K_{i,j} = x_i \times x_j^T$. I is the identity matrix and $\mathbf{1}$ is a column vector with all its elements equal to 1. With:

$$\psi^{-1} = \begin{bmatrix} K + \frac{1}{C}\mathrm{I} & 1 \\ 1^T & 0 \end{bmatrix}^{-1} \tag{3.4}$$

Problem (3.2) can be solved by:

$$\begin{bmatrix} \alpha \\ b \end{bmatrix} = \psi^{-1}\begin{bmatrix} y \\ 0 \end{bmatrix} \tag{3.5}$$

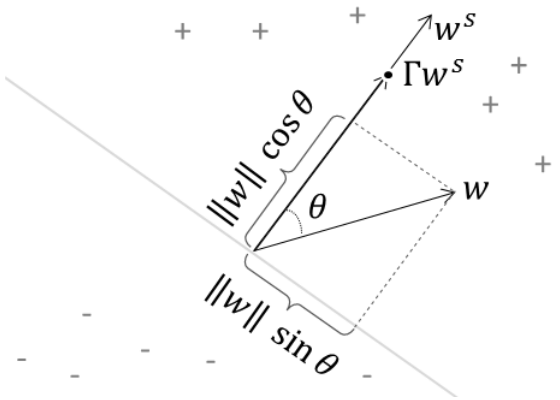


Figure 3.1: Projecting w to w' in PMT-SVM (adapted from [2]).

3.2.2 ASVM & PMT-SVM

Adaptive SVM (ASVM) is the first work using LS-SVM for transfer learning for vision related tasks [68]. The goal of ASVM is to minimize the distance between the target hyperplane w and source one w' incorporating with the transfer parameter γ. The objective function is defined as follow:

$$\begin{aligned} \min \quad & L_{ASVM} = \frac{1}{2}\|w - \gamma w'\|^2 + \frac{C}{2}\sum_{i=1}^{l}\varepsilon_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{3.6}$$

Here, γ controls the amount of transfer regularization. Intuitively, the regularization term of ASVM is like a spring between w and $\gamma w'$. Equivalently, assume $\|w'\|^2 = 1$ and the regularization term can be expended as:

$$\|w - \gamma w'\|^2 = \|w\|^2 - 2\gamma \|w\| \cos\theta + \gamma^2$$

Where θ is the angel between w and w'. However, the term $-\gamma \|w\| \cos\theta$ also encourages $\|w\|$ to be larger (as this reduces the cost) which prevents margin maximization. Thus γ, which defines the amount of transfer regularization, becomes a tradeoff parameter between margin maximization and knowledge transfer.

Based on this, Projective Model Transfer SVM (PMT-SVM) is proposed to solve the transfer problem by optimizing the following objective function [2]:

$$\begin{aligned} \min \quad & L_{PMT} = \frac{1}{2}\|w\|^2 + \gamma\|Pw'\|^2 + \frac{C}{2}\sum_{i=1}^{l}\varepsilon_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \\ & w^T w' \geq 0 \end{aligned} \tag{3.7}$$

Where P is the the projection matrix $P = I - \frac{w'^T \times w'}{w' \times w'^T}$. Therefore, $\|Pw\|^2 = \|w\|^2 \sin^2\theta$ is the squared norm of the projection of the w onto the source hyperplane (see Figure 3.1). As $\gamma \to 0$, the loss function (3.7) becomes a classic LS-SVM loss function. Because (3.7) is convex, it can be solved effectively by quadratic optimization.

In summary, both ASVM and PMT-SVM are designed to answer the question: how to transfer by solving some convex objective function. However, they both require a pre-defined parameter γ, which controls the amount of the knowledge to be transfered, to complete the objective function. In most cases, this parameter can only be set according to the background knowledge. Also, both methods can only adopt the knowledge from single source model which limits the their performance.

3.2.3 Multi-KT

To learning from many sources, Multi Model Knowledge Transfer (Multi-KT) is proposed to reply on multiple sources by assigning different weight for each of them [60]. Similar to the objective function (3.6), its objective function is defined as follow:

$$\begin{aligned} \min \quad & L_{Multi-KT} = \frac{1}{2}\left\| w - \sum_k \beta_k w'_k \right\|^2 + \frac{C}{2}\sum_{i=1}^{l}\zeta_i\varepsilon_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + \varepsilon_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{3.8}$$

Here, β_k is the weight assigned for the kth source model and ζ_i is defined as:

$$\zeta_i = \begin{cases} \frac{N}{2N^+} & \text{if } y_i = 1 \\ \frac{N}{2N^-} & \text{if } y_i = -1 \end{cases}$$

where N^+ and N^- are number of positive and negative examples respectively and N is the total number of examples.

The primal Lagrangian for optimization problem (3.8) can be written as:

$$L_{Multi-KT}(w, \beta, \varepsilon, \alpha) = \frac{1}{2}\left\| w - \sum_k \beta_k w'_k \right\|^2 + \frac{C}{2}\sum_{i=1}^{l}\zeta_i\varepsilon_i^2 + \sum_{i=1}^{l}\alpha_i\left[wx_i + b + \varepsilon_i - y_i\right] \tag{3.9}$$

So problem (3.8) can be solved once β is set. Different from ASVM and PMT-SVM which require background knowledge to select proper transfer parameter, Multi-KT can estimate the transfer parameter β itself by using the closed-form Leave-One-Out (LOO) error. According to [12], the closed-form LOO error is defined as:

$$y_i - \hat{y}_i = \frac{\alpha_i}{\psi_{ii}^{-1}} \qquad \text{for} \qquad i = 1, ..., l \tag{3.10}$$

Here ψ_{ii}^{-1} is its *ith* diagonal element of ψ^{-1} in (3.4).

To estimate β, a loss function, similar to hinge loss, is defined as:

$$\begin{aligned} \min \qquad & \mathcal{L} = \sum_{i}^{l} \zeta_i |1 - y_i \hat{y}_i|_+ \\ \text{s.t.} \qquad & \|\beta\| \leq 1 \end{aligned} \tag{3.11}$$

Here $|x|_+ = max(x, 0)$. Intuitively, if β is properly set, $y_i \hat{y}_i$ should be positive for each i. However, focusing only on the sign of those quantities would result in a non-convex formulation with many local minima. By adding the $|\cdot|_+$ function, formula (3.13) becomes convex and can be solved by gradient descent method.

3.2.4 MULTIpLE

MULticlass Transfer Incremental LEarning (MULTIpLE) focuses on adding a new class to a existing N class source problem while preserving the performance on the old classes [37]. To preserving the overall performance of the classifier among $N + 1$ classes, MULTIpLE contains two parts: incremental learning for existing N classes and transfer learning for the new class.

For the existing N classes, MULTIpLE uses the similar strategy as ASVM, setting the transfer parameter γ to 1. For the new class, it adopts the strategy of Multi-KT, combining knowledge from existing N-class models. As a result, the objective function for the hyperplanes in MULTIpLE is defined as:

$$\begin{aligned} \min \qquad & L_{MULTIpLE} = \frac{1}{2} \sum_{n=1}^{N} \|w_n - w'_n\|^2 + \frac{1}{2} \left\| w_{N+1} - \sum_{k=1}^{N} w'_k \beta_k \right\|^2 + \frac{C}{2} \sum_{n=1}^{N+1} \sum_{i=1}^{l} \varepsilon_{i,n}^2 \\ \text{s.t.} \qquad & \varepsilon_{i,n} = Y_{in} - x_i w_n - b_n \end{aligned} \tag{3.12}$$

Similar like Multi-KT, MULTIpLE uses LOO error in (3.10) to estimate the transfer parameter β for the new class. The objective function for β estimation is defined by [15]:

$$\begin{aligned} \min \qquad & \mathcal{L}(\beta, i) = \begin{cases} \max\limits_{n \neq y_i} \left| 1 - \hat{Y}_{in}(\beta) - \hat{Y}_{iy_i}(\beta) \right|_+ & : y_i \neq N + 1 \\ \left| 1 - \hat{Y}_{in}(\beta) - \hat{Y}_{iy_i}(\beta) \right| & : y_i = N + 1 \end{cases} \\ \text{s.t.} \qquad & \|\beta\| \leq 1 \end{aligned} \tag{3.13}$$

We can find the optimal β with projected subgradient descent [10].

3.3 Linear Combination Strategy

A major challenge in transferring the knowledge from the source is to design a strategy to combine the source knowledge with the knowledge from the target task. In this section, we propose propose a combination strategy that using linear argumentation to combine the source and target knowledge. An important advantage of this strategy is that we can adjust the impact of the knowledge comes from the source model more flexibly by just modifying the value of its coefficient (called transfer parameter). As long as the source knowledge hurts the transfer model, we can reduce its impact to avoid negative transfer.

we generally assume that we can't access the source data

3.3.1 $\mathcal{D}$-relationship between the hypothesis and the distribution

Intuitively, when transferring the knowledge from another task, the performance of the model for the target task is greatly determined by the relationship of these two tasks. Here we use the $\mathcal{H}$-divergence to measure the relationship of two different tasks. $\mathcal{H}$-divergence can be defined as follow:

Definition 1 *(from Kifer et al. [33]) Given a domain $\mathcal{X}$ with two probability distributions, let $\mathcal{H}$ be a hypothesis class on $\mathcal{X}$ and denote by $I(h)$ the set for which $h \in \mathcal{H}$ is the characteristic function where $x \in I(h) \leftrightarrow h(x) = 1$. The $\mathcal{H}$-divergence between these two probability distribution D and D' is*

$$d_{\mathcal{H}}(D, D') = 2 \sup_{h \in \mathcal{H}} |\Pr_D [I(h)] - \Pr_{D'} [I(h)]|$$

When $d_{\mathcal{H}}(D, D')$ is small. it means for any $(x, y) \in D$ and $(x', y') \in D'$, if $y = y'$, there exists a hypothesis h such that the difference between $h(x)$ and $h(x')$ can be small with a high probability. Otherwise, they are unrelated. From its definition we can see that $\mathcal{H}$-divergence is a metric to measure the dissimilarity of two distributions. The more similar two distributions are, the smaller $\mathcal{H}$-divergence is. Ben et al. [5] show that the performance of the target model is affected by the $\mathcal{H}$-divergence of the probability distributions of the two tasks. Specifically, the smaller $\mathcal{H}$-divergence is, the more benefit the transfer model can get from the source task.

Suppose D and D' are two distributions for the binary classification problem with class labels $\{1, -1\}$ and there is a binary hypothesis h' that is consistent with D', i.e. for any $(x', y') \in D'$, $h'(x') = y'$. If D and D' are related, according to Definition 1, for any example $(x, y) \in D$,

we would expect a high probability for $\Pr_{(x,y)\sim D}(h'(x) = y)$. Here, we call the hypothesis h' and distribution D are positive related. On the other hand, as D' is a distribution for binary classification problem, we can use $-h'$ to denote the complement of the hypothesis h'. If $-h'$ is consistent with D', we should expect a high probability for $\Pr_{(x,y)\sim D}(h'(x) = -y)$. Here we call the hypothesis h' and distribution D are negative related. We use $d_{h'\sim D'}(D)$ to denote the $\mathcal{D}$-relationship of the hypothesis h' and the distribution D. The $\mathcal{D}$-relationship of a hypothesis h' and a distribution D is defined as follow:

Definition 2 *Let D and D' be two distributions for the binary classification problem $\{1, -1\}$. Given a hypothesis h' consistent with D' and , the $\mathcal{D}$-relationship of the hypothesis h' and the distribution D can be written as:*

$$d_{h'\sim D'}(D) = \min\left(\underbrace{\Pr_{(x,y)\sim D}(h'(x) = y)}_{\textit{Positive term}}, \underbrace{\Pr_{(x,y)\sim D}(h'(x) = -y)}_{\textit{Negative term}}\right)$$

Here we call the two terms $\sum_{(x,1)\in D} h'(x)$ and $\sum_{(x,-1)\in D} h'(x)$ positive and negative terms respectively. As we discussed above, when h' and D are positive related, we should expect most positive and negative examples in D be classified as positive and negative in hypothesis h'. Therefore, $d_{h'\sim D'}(D)$ equals $\Pr_{(x,y)\sim D}(h'(x) = y)$. When they are negative related, $d_{h'\sim D'}(D)$ equals $\Pr_{(x,y)\sim D}(h'(x) = -y)$. Therefore, when h' and D are related, the $\mathcal{D}$-relationship should be large. When h' and D are unrelated, the output of h' should be more random for both positive and negative examples. Therefore, both positive and negative terms are closed to 0.5 and the $d_{h'\sim D'}(D)$ should be closed to 0.5 as well. Obviously, $\mathcal{D}$-relationship describes the agreement of the hypothesis h' and the distribution D. In a binary classification scenario, h' and D are related if h' is highly agree with D (positive related) or disagree with D (negative related). If $\mathcal{D}$-relationship is closed to 0.5, h' and D are unrelated. Similarly, we can get a conclusion based on the conclusion of [5] that the smaller $\mathcal{D}$-relationship is, the less benefit a transfer model can get from the source model(s).

3.3.2 Leverage the Source Knowledge with Data Arguementation

Then our self-defined category learning problem can be described as follow: Given a task T to distinguish whether an example is from one certain category C from a domain $\mathcal{X} \times \mathcal{Y}$ with D probability distribution over $\mathcal{X}$, $\mathcal{X}$ is the input feature and $\mathcal{Y}$ is the label set $\{1, -1\}$. We use an examples with label 1 to denote the positive example (i.e. belongs to the category) and an example with label -1 to denote the negative example (i.e. not belong to this category). There is another group of binary source models $F' = \{f'_1, ..., f'n\}$ trained from another independent task S on another domain $\mathcal{X}' \times \mathcal{Y}'$ with D' probability distribution over $\mathcal{X}'$. The F' and D are

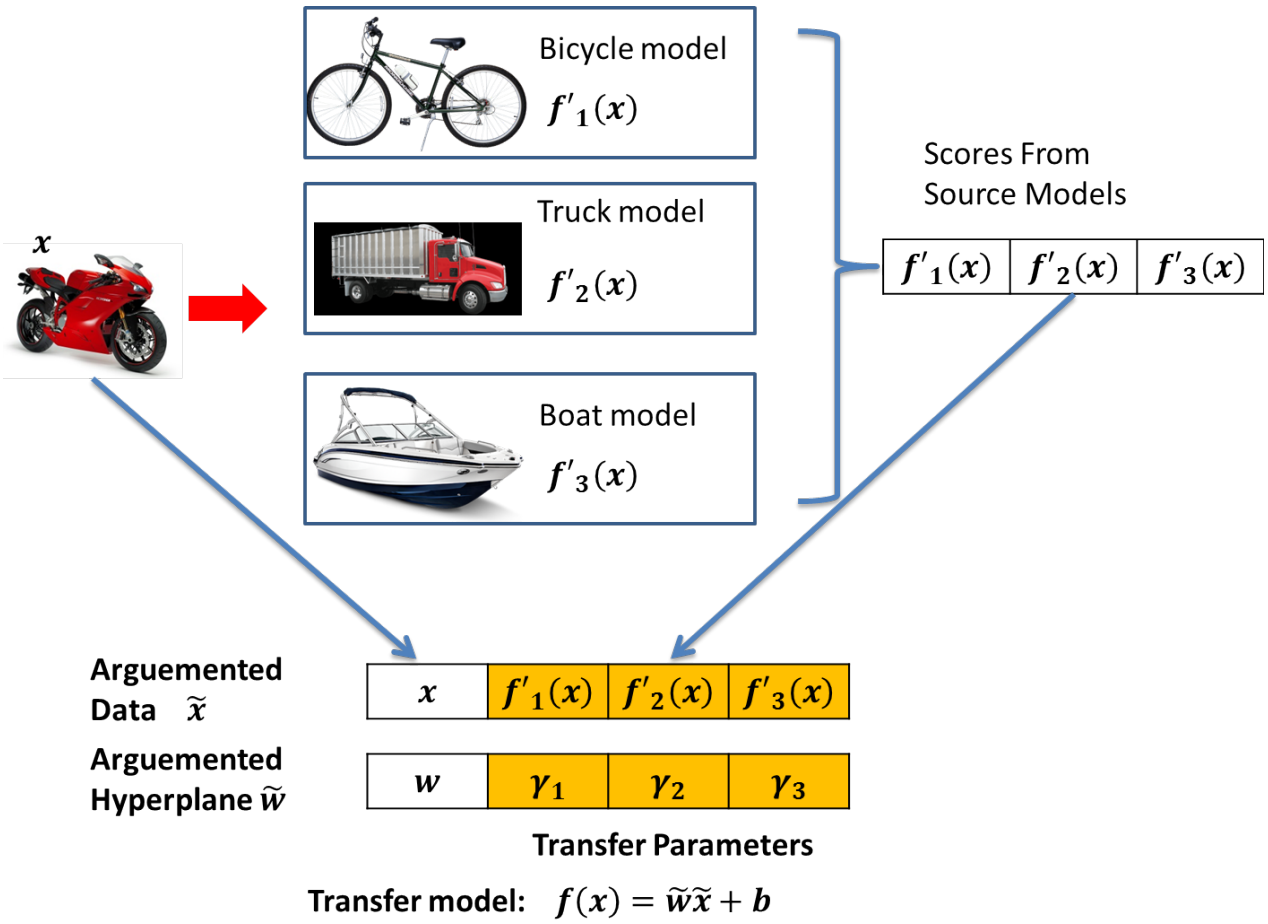


Figure 3.2: A graphical representation of linear argumentation. The score from the source model can be considered as an auxiliary feature and the transfer parameter controls the value of the auxiliary feature. For linear classifiers, it can be considered as a linear combination of the decision from the source model and target data (see (3.15)).

related. Given a small training set $T_{train} = \{(x, y)\} \subset \mathcal{X} \times \mathcal{Y}$, $\mathcal{X}$, our target task is to learn a classification model $f : \mathcal{X} \to \mathcal{X}$ from T_{train} incorporating with $F' = \{f'_1, ..., f'k\}$ so that f can perform well on an independent testing set $T_{test} = \{(x, y)\} \subset \mathcal{X} \times \mathcal{Y}$.

Here, let's set the both source and target model f and F' in the hypothesis space of function $\mathcal{H}$ which equals to space of all the linear models of the form.

$$f(x) = w^T x + b \tag{3.14}$$

To leverage the knowledge from the source models F', we consider to use the decision scores from the source model as the auxiliary information and use the auxiliary information to adjust the decision for our transfer model. By introducing the auxiliary information from the

source models, we can have several advantages especially when we use linear model such as SVM.

The first advantage is simplicity. Given an example x and the source models F', we can simply add the decision scores of each $f'_i(x)$ from F' to the target model to affect its decision. This process can be written as:

$$\begin{aligned} f(x) =& w^T x + b + \sum_{i=1}^{k} \gamma_i f'_i(x) \\ \text{st.} \quad & f'_i(x) = w'^T_i x \end{aligned} \tag{3.15}$$

Here $\gamma = \{\gamma_1, ..., \gamma_k\}$ is a hyper-parameter to control the amount of the knowledge transferred from each of the source model f'_i. This process is equivalent to the data argumentation approach where the scores of the source model is used as the auxiliary feature (see Figure 3.2). By introducing the auxiliary information from the source model, we expect to reduce the bias from the target training set and get an improvement from the auxiliary feature. With this data argumentation, the transfer learning problem becomes a traditional machine learning problem. The transfer model can be determined as long as we get the hyperplane w

Another advantage is flexibility. As the decision scores from the source model are used as the auxiliary knowledge. By using the transfer parameters to control the amount of the knowledge from each source model, the model for the target task is more adaptive to different situations. If the knowledge from the source model is related, we should increase its impact, i.e. , otherwise, its impact should be decreased.

In the previous work [37] [60], according to the definition of $\mathcal{H}$-divergence, they consider that there are only two relationships between the target task and a source model: related or unrelated. When they are related, the transfer parameter should be set to positive and 0 otherwise. In our work, according to Definition 2, there could be 3 different relationships between the target task and a specific source model: unrelated, positive related and negative related.

For a multi-class scenario, our goal is to use a linear system to distinguish different categories with One-VS-All strategy. For each category, we should build a binary classification model and assign the label according to $sign\,[f(x)]$. According to Eq. (3.15), for any example x, the decision score of our binary model consist of two parts: a score learned from our target task which is the term of $w^T x + b$ and the knowledge from the source models, i.e. $\sum_i \gamma_i f'_i(x)$. In order to The transfer parameters should be set as follow:

- When the source model and the target task are not related, according to Definition 2, the expectation of the decision score from the source model is closed to be 0.5 and it can't provide much useful information for the target task. Therefore, the auxiliary feature from

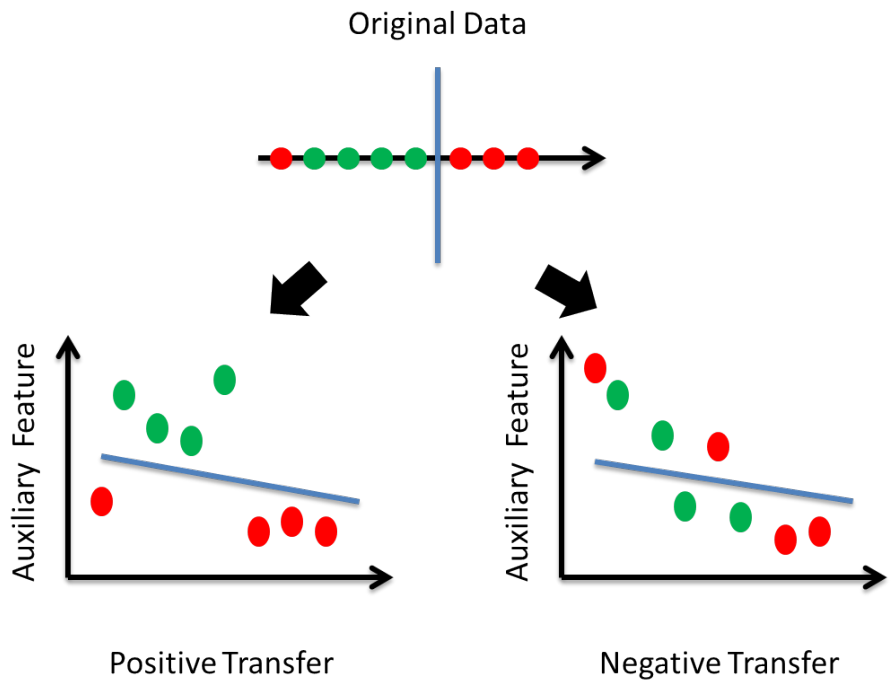


Figure 3.3: Suppose the original data is one dimensional and we use the blue line to denote the optimal decision surfce of a linear model (the upper figure). By adding an related auxiliary feature, we can improve the performacne of the classifier (the bottom-left figure) while unrelated one can decrease the performance and lead to negative transfer (the bottom-right figure).

this source model is considered to be redundant. Thus, to eliminate its affect, the transfer parameter should be set closed to 0. As a result, the impact of the decision of the source model is minimized and the target model is less likely to suffer from negative transfer. In the extreme case where the transfer parameter is set 0, no auxiliary information is introduced to adjust the bias in the target task and therefore, we can guarantee that, at least, negative transfer won't happen.

- When the source and target tasks are negative related, i.e. the expected decision score for positive examples in target task is negative and for positive

Therefore, let $\tilde{x} = (x, f'_1(x), ..., f'_k(x))$ and $\tilde{w} = (w, \gamma_1, ..., \gamma_k)$. With the LS-SVM setting, our transfer problem can be solved by solving the following optimization problem with argument data:

$$\begin{aligned} \min \quad & L_{LSSVM}(\tilde{w}) = \frac{1}{2}\|\tilde{w}\|^2 + \frac{C}{2}\sum_{i=1}^{l} e_i^2 \\ \text{s.t.} \quad & \tilde{w} = (w, \gamma_1, ..., \gamma_k) \\ & y_i = \tilde{w}\tilde{x}_i + b + e_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{3.16}$$

Table 3.1: Notations used in this chapter

$f'(x)$	binary model for source task
$f(x)$	binary model for target task
$\phi(x)$	function mapping the input sample into a high dimensional feature space.
X	instance matrix with each row representing one instance
$\boldsymbol{W}$	(N+1)-column hyperplane matrix for target task. Each column represents one hyperplane of a binary model
$\boldsymbol{W'}$	hyperplane matrix for the source task
$\boldsymbol{a'}$	the Lagrangian multiplier matrix for source problem. Each column represents a set of Lagrangian multiplier for a binary SVM model
$\boldsymbol{a}$	the Lagrangian multiplier matrix for target problem
$\boldsymbol{b'}, \boldsymbol{b}$	the bias vector for source and target task
$\boldsymbol{a_i}$	i_{th} column of matrix $\boldsymbol{a}$
$\boldsymbol{\beta}$	row vector $[\beta_1, ..., \beta_N]$ to control the prior knowledge for the new category
ε_{ny_i}	loss parameter. $\varepsilon_{ny_i} = 1$ if $n = y_i$ and 0 otherwise
ψ, ψ^{-1}	ψ is the modified kernel matrix for solving binary LS-SVM and ψ^{-1} is the inverse matrix of ψ

Here, we can treat γ and w as a whole parameter $\tilde{w}$ and solve the problem (3.16) directly with the method described in subsection 3.2.1 to obtain the values of them. However, this would case serious problem if the dimension of w is much larger than the dimension of γ. In this situation,

When we consider the transfer parameter γ as a hyperparameter that can be determined by certain prior knowledge. To regularize $\|\tilde{w}\|^2$ is equivalent to regularize $\|\tilde{w} - \gamma w'\|^2$. Therefore, function (3.16) can be represented as:

$$\begin{aligned} \min \quad & L_\gamma(w) = \frac{1}{2}\left\| w - \sum_i^n \gamma_i w_i' \right\|^2 + \frac{C}{2}\sum_{i=1}^{l} e_i^2 \\ \text{s.t.} \quad & y_i = wx_i + b + e_i \quad \text{for} \quad i \in \{1, 2, ..., l\} \end{aligned} \tag{3.17}$$

To solve the problem (3.17), we first introduce some important notations used in the rest of this chapter in Table 3.1. We use any letter with apostrophe to denote the information from the source data, e.g. if $f(x)$ denotes the model for the target task, $f'(x)$ denotes the model for the source one.

The primal Lagrangian for this optimization problem 3.17 given the unconstrained mini-

mization problem can be written as:

$$L_\gamma(w, b, \alpha, e) = \frac{1}{2}\left\| w - \sum_i^n \gamma_i w_i' \right\|^2 + \frac{C}{2}\sum_{i=1}^{l} {e_i}^2 - \sum_{i=1}^{l} \alpha_i \{w x_i + b + e_i - y_i\} \tag{3.18}$$

The optimal solution can be found by satisfying the following condition:

$$\begin{aligned}
&\frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_i^n \gamma_i w_i' + \sum_i^l \alpha_i x_i \\
&\frac{\partial L}{\partial b} = 0 \rightarrow \sum_i^l \alpha_i = 0 \\
&\frac{\partial L}{\partial e_i} = 0 \rightarrow C e_i = \alpha_i \qquad i = 1, ..., l \\
&\frac{\partial L}{\partial \alpha_i} = 0 \rightarrow y_i = w x_i + b + e_i \qquad i = 1, ..., l
\end{aligned} \tag{3.19}$$

Let $X = [x_1, x_2, ..., x_l]$ and $Y = [y_1, y_2, ..., y_l]$, Eq (3.19) can be written in the following compact format:

$$\begin{bmatrix} XX^T + \frac{\mathbf{I}}{C} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix} \tag{3.20}$$

where $\mathbf{1}$ is a column vector whose elements are 1.

In real applications, when we have n single source categories and their corresponding source model $f_1'(x), f_2'(x), ..., f_n(x)$, their loss function can be represented as:

$$\begin{aligned}
\min \qquad & L_\gamma(w_1, w_2, ..., w_n) = \frac{1}{2}\sum_{i=1}^{n} \left\| w_i - \gamma_i w_i' \right\|^2 + \frac{C}{2}\sum_{i=1}^{n}\sum_{j=1}^{l} e_{ij}^2 \\
\text{s.t.} \qquad & y_{ij} = w_i x_i + b_i + e_{ij} \quad \text{for} \quad i \in \{1, 2, ..., l\}, j \in 1, 2, ..., n
\end{aligned} \tag{3.21}$$

Let $\mathbf{W} = [w_1, w_2, ..., w_n]$, $\mathbf{W}' = [w_1', w_2', ..., w_n']$ and $D(\gamma)$ be the diagonal matrix $diag(\gamma_1, \gamma_2, ..., \gamma_n)$. The corresponding solution for problem (3.21) is:

$$\begin{bmatrix} XX^T + \frac{\mathbf{I}}{C} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} + \begin{bmatrix} D(\gamma)X(W')^T \\ 0 \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix} \tag{3.22}$$

Here $Y = [y_{ij} | i = 1, .., l; j = 1, ..., n]$ is the encoded label matrix where for each example (x_i, y_i):

$$y_{ij} = \begin{cases} 1 & \text{if } y_i = j \\ -1 & \text{otherwise} \end{cases} \tag{3.23}$$

We can see that Eq. (3.21) can be solved by directly once the transfer parameters $\gamma = \{\gamma_i | i = 1, ..., n\}$ is determined.

To summarize this section, we propose a strategy which uses linear combination to combine the knowledge of the source model and the knowledge from the target dataset. We show that this linear combination strategy can be considered as a data argumentation approach and thus, the transfer learning problem becomes a traditional machine learning problem. Moreover, we show that by setting different values of the transfer parameters, we can control the amount of the knowledge from the source model effectively. In the next section, we show that we can estimate the cross-validation error of the LS-SVM in a closed form.

3.4 Cross-Validation Error for LS-SVM

In this section, we introduce the cross-validation estimation for LS-SVM. As we mentioned in Section 3.3 that once we determine the value of the transfer parameters γ, problem (3.21) can be solved directly. In Section 3.3, we treat the transfer parameters as the parameters to be set in advance. In this section, we introduce how we can estimate the unbiased transfer parameters by using an important property of LS-SVM for parameter estimation.

3.4.1 Closed Form Cross-validation Estimation for LS-SVM

To estimate the transfer parameter, we adopt cross-validation for parameter estimation. Cross-validation is widely used for parameter estimation. Suppose we have a model with one or more unknown parameters, and a data set to which the model can be fit (the dataset). The fitting process optimizes the model parameters to make the model fit the data as well as possible. On the other hand, we have to make sure that this model have the generalization ability, i.e. it can also work well on other independent validation dataset. Otherwise, the model is overfitted to the training data set. However, in practical problem we may not be able to get another independent data set except for the original dataset. Therefore, we have to manually create the validation set by selecting some of the data from the original dataset and use them for evaluation. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). By repeating these process several rounds with different partitions, we can significant reduce the variability of our estimation using the average results over the rounds. There are several different types of cross-validation methods due to their partition strategies. K-fold cross-validation is the most popular type among them. In k-fold cross-validation, the original dataset is randomly partitioned into k equal sized subsamples. K-fold cross-validation is performed in K rounds and in each round, each one of the K partition is used exactly once as the validation set and the rest partitions are

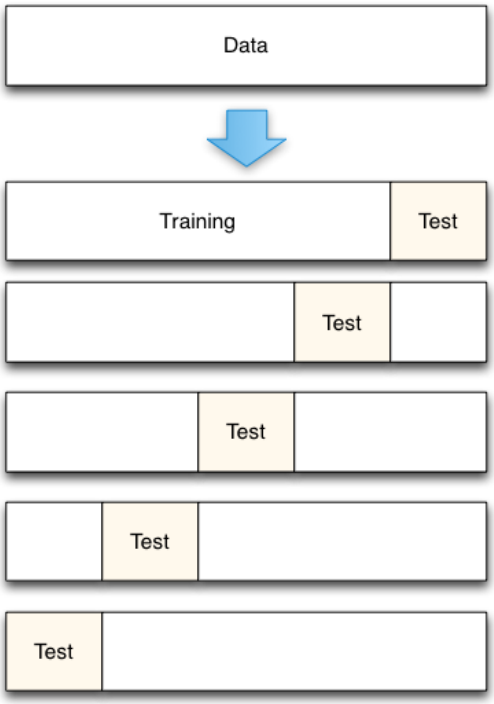


Figure 3.4: An illustration of 5-fold cross-validation

used as the training set.

In our work, we also employ cross-validation for parameter estimation. Here we show that the cross-validation error for LS-SVM can be represented in an efficient way. As a result, we don't have to actually perform cross-validation and re-train the models in each round to obtain the cross-validation error.

Theorem 3.4.1 (An extension of [12]) *Given a dataset* $D = \{(x_i, y_i)|i = 1, ..., l\}$*, the solution of a LS-SVM on* D *can be written as Eq.* (3.1)*. Assume that* $D^{(n)} = \{(x_i, y_i)|i = 1, ..., n\}$ *is a subset of* D *and* $D \backslash D^{(n)}$ *is the complement of* $D^{(n)}$ *in* D*. Let* $[\alpha_1, ..., \alpha_n]$ *be the corresponding* n *rows of* α *in Eq.* (3.3) *and* S_n*,* s *and* $S_{(l-n+1)}$ *represent the square blocks of matrix:*

$$\mathcal{S} = \left[\begin{array}{c|c} S_n & s \\ \hline s^T & S_{(l-n+1)} \end{array}\right] = \left[\begin{array}{cc} K + \frac{1}{C}\mathrm{I} & 1 \\ 1^T & 0 \end{array}\right] \tag{3.24}$$

The unbiased leave out error of a LS-SVM trained from $D \backslash D^{(n)}$ *on* $D^{(n)}$ *can be estimated as:*

$$ERR_{leave-out} = \left(S_n - sS^{-1}_{(l-n+1)}s^T\right)[\alpha_1, ..., \alpha_n]^T$$

We show the proof in Appendix A.2. This remarkable result allows cross-validation to be used while only fitting the model once to all available data.

3.4.2 Leave-one-out Cross-validation for estimation

Therefore, we have a convenient solution to estimate the unbiased cross-validation error. However, to perform a cross-validation in practice, decide the partition strategy. Suppose we have l examples in the original data and we want to perform a K-fold cross-validation on it. Thus we have $\mathbf{C}_{\mathbf{l}}^{\mathbf{l/K}}$ possible combinations. This means larger K leads to less computation time. Moreover, larger K means less bias towards overestimating the true expected error (as training folds will be closer to the total dataset). When we set $K = l$, we perform l-fold cross-validation, i.e. leave-one-out cross-validation (LOO-CV). Vapnik et al. [65] show that LOO-CV can provide an almost unbiased estimation error. Previous work also show that transfer learning on the small target training set can be benefit from LOO-CV to better estimate the transfer parameters as well as prevent negative transfer [36]. Due to the reasons above, in this thesis, we use LOO-CV to estimate the transfer parameters to optimize our transfer model.

According to Theorem 3.4.1, the LOO-CV error for example x_i can be represented as:

$$y_i - \hat{y}_i = \frac{\alpha_i}{\mathcal{S}_{ii}^{-1}} \tag{3.25}$$

Where $\mathcal{S}^{-1}$ is the inverse matrix of $\mathcal{S}$ in Eq. (3.24) and $\mathcal{S}_{ii}^{-1}$ is the ith diagonal matrix of $\mathcal{S}^{-1}$.

To summarize this section, we show that we can calculate the cross-validation error for LS-SVM in a closed form which is an very effective approach to estimate the transfer parameters. Due to the advantages of LOO-CV, such as computational efficiency and alleviation of negative transfer, we choose LOO-CV as our cross-validation strategy. In the next section, we use the LOO-CV error as an estimation to obtain a multi-class loss for SVM model.

3.5 Multi-class Loss with LOO error

From the previous section we can see that, the unbiased estimated decision value of LOO-CV for an example can be written in a closed form. In this section, we introduce the multi-class loss function and use the LOO-CV error to estimate the multi-class loss. We propose a novel objective function based on the estimated multi-class loss to estimate the transfer parameters.

In previous section, we discussed the LS-SVM in binary class scenario. In practice, we have to handle the multi-class scenario. A common strategy to convert the binary class scenario to a multi-class scenario is to assign label to the highest confidence score of each binary SVM. Suppose we trained n binary SVMs $f_i, i \in 1, ..., n$ for a n-class problem, the label of an example x is assigned by:

$$H_{\mathcal{F}}(x) = \hat{y} = \arg\max_{i \in 1,...,n} f_i(x) \tag{3.26}$$

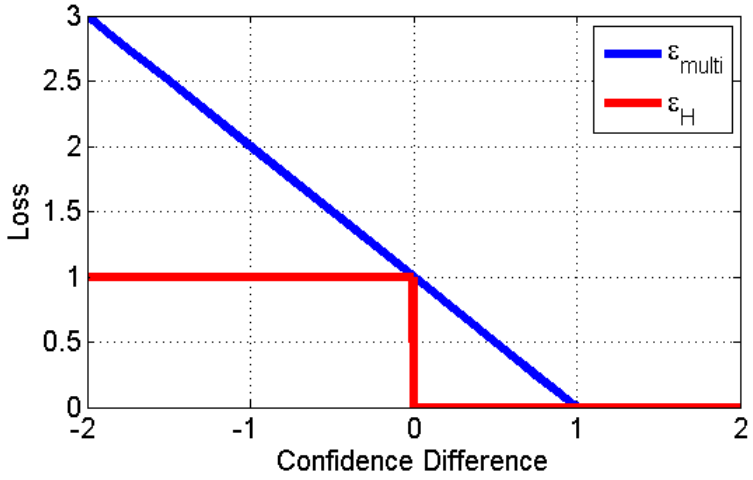


Figure 3.5: Loss function comparision for multi-class hinge loss ϵ_{multi} and classical zero-one loss ϵ_H

Let $[\![y]\!]$ be 1 if the predicate y holds and 0 otherwise. The empirical error for a multi-class problem is given by:

$$\epsilon_H = \frac{1}{l}\sum_{1}^{l}[\![H_{\mathcal{F}}(x_i) \neq y_i]\!] \tag{3.27}$$

Our goal is to find a group of function $\mathcal{F}$ that has a small empirical error on the training set as well as on an unforeseen test set. Because function (3.27) takes discrete value on the predicate, directly approaches which tries to minimize the empirical error are computational expensive [16]. Crammer et al. [15] proposed an piecewise linear bound to replace it:

$$\epsilon_{multi} = \frac{1}{l}\sum_{i}^{l}\epsilon_{multi}^{(i)} = \frac{1}{l}\sum_{i}^{l}\max_{n\in\{1,\ldots,N\}}\left[1 - \varepsilon_{ny_i} + f_n(x_i) - f_{y_i}(x_i)\right] \tag{3.28}$$

where ε_{ny_i} is equal 1 if $n = y_i$ and 0 otherwise.

According to the multi-class loss (3.28), its bound is 0 if the confidence score for the correct label is larger by at least 1 than the confidence scores assigned to the other labels. Otherwise, we suffer a linear loss that is proportional to the difference between the confidence of the correct label and the maximum among the confidence of the other labels (see Figure 3.6).

Let $\hat{y}_{ij}$ denotes the LOO-CV estimation of the example x_i for binary model f_i. From Eq. (3.25) we can see that:

$$\hat{y}_{ij} = y_{ij} - \frac{\alpha_{ji}}{S_{ii}^{-1}} \tag{3.29}$$

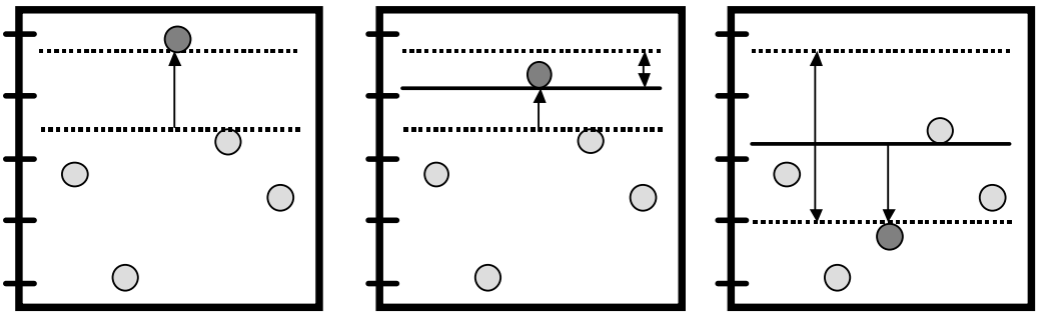


Figure 3.6: Illustration of the margin bound for a single example in 3 scenarios. The cirles denote different confidences and the correct label is ploted in dark grey. The height of each lable is the confidence score. In the left figure,$\epsilon(x) = 0$. In the middle figure, even though the confidence of the correct label is the largest, it fails to be larger by 1 than the confidence of the runner-up and have a small loss. In the right figure, the confidence of the correct label is not the largest one and have a very large loss.

where the matrix $\boldsymbol{\alpha} = [\alpha_{ji}|i = 1, ..., l; j = 1, ..., N]$ should satisfy:

$$\mathcal{S}\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} Y - D(\gamma)X(W')^T \\ 0 \end{bmatrix} \tag{3.30}$$

Let:

$$\begin{aligned} \mathcal{S}\begin{bmatrix} \boldsymbol{\alpha}' \\ \boldsymbol{b}' \end{bmatrix} &= \begin{bmatrix} Y \\ 0 \end{bmatrix} \\ \mathcal{S}\begin{bmatrix} \boldsymbol{\alpha}'' \\ \boldsymbol{b}'' \end{bmatrix} &= \begin{bmatrix} X(W')^T \\ 0 \end{bmatrix} \end{aligned} \tag{3.31}$$

Thus we have:

$$\alpha_{ji} = \alpha'_{ji} - \gamma_j \alpha''_{ji} \tag{3.32}$$

The multi-class LOO-CV loss (3.28) can be re-written as:

$$\begin{aligned} \epsilon^{(i)}_{multi} &= \max_{n\in\{1,...,N\}} \left[1 - \varepsilon_{ny_i} + \hat{y}_{in} - \hat{y}_{iy_i}\right] \\ &= \max_{n\in\{1,...,N\}} \left[1 - \varepsilon_{ny_i} + y_{in} - \frac{\alpha'_{ni} - \gamma_n \alpha''_{ni}}{\mathcal{S}^{-1}_{ii}} - y_{iy_i} + \frac{\alpha'_{y_i i} - \gamma_{y_i} \alpha''_{y_i i}}{\mathcal{S}^{-1}_{ii}}\right] \end{aligned} \tag{3.33}$$

With the label encoding strategy in (3.23), we the multi-class LOO error for example (x_i, y_i) can be written as:

$$\epsilon^{(i)}_{multi} = \max\left(\max_{n\neq y_i}\left[\frac{(\alpha'_{iy_i} - \alpha'_{in}) + (\gamma_n \alpha''_{ni} - \gamma_{y_i}\alpha''_{y_i i})}{\mathcal{S}^{-1}_{ii}} - 2\right], 0\right) \tag{3.34}$$

Thus the optimal transfer parameter $\gamma*$ should be the one that minimize the LOO-CV error ϵ_{multi} which is the average of the error for $\epsilon^{(i)}_{multi}$.

In summary, in this section, we introduce the multi-class loss for LOO-CV. We show that the multi-class LOO-CV loss is a function of the transfer parameter. Therefore, by finding the minimum of the multi-class LOO-CV loss, we can get the optimal transfer parameter.

3.6 Transfer Parameter Optimization

In last section, we introduced the multi-class LOO-CV loss. We can see that it is a function of the transfer parameter. Therefore, in this section, we propose a novel objective function based on the multi-class LOO-CV loss. We use sub-gradient descent to optimize this objective function to get the optimal solution.

As we discussed in the last section, the optimal transfer parameter is the one that minimize the LOO-CV error. We can see that the multi-class LOO-CV loss is an extension of binary hinge loss and therefore, it is convex. We propose the following objective function based on the multi-class LOO-CV loss:

$$\begin{aligned} \textbf{min} \quad & L(\gamma) = \frac{\lambda}{2}\sum_{n=1}^{N}\|\gamma_n\|^2 + \sum_{i=1}^{l}\epsilon_i \\ \textbf{s.t.} \quad & 1 - \varepsilon_{ny_i} + \hat{y}_{in}(\gamma) - \hat{y}_{iy_i}(\gamma) \leq \epsilon_i; \\ & \lambda \geq 0 \end{aligned} \tag{3.35}$$

Besides the multi-class LOO-CV loss, we add an extra L2 regularization term in objective function (3.35). Compared to other methods such as [59] and [37], which optimize the multi-class loss directly with certain constraint for the transfer parameters, the L2 regularization term has the following advantages:

- The regularization term can limit the value of the transfer parameter. Our objective function try to make a balance between the knowledge transferred from the source and the LOO-CV error made on the target data. For example, if increase the amount of the source knowledge can improve the performance of our model, we can expect a decrease on the LOO-CV loss. However, because we the LOO-CV error is estimated from a small dataset, it is more likely to prune to overfitting. Overfitting on a small dataset would lead to poor generalization ability. As a result, it would lead to decreased performance. In [36], it shows that the upper bound of the generalization error of a transfer model can be written as:

$$Err \leq Err_{loo} + O(\|\gamma\|^2) \tag{3.36}$$

Therefore, limit the value of the transfer parameter can improve the generalization ability of our transfer model.

- Because of the L2 regularization term, our objective function is a strongly convex function. Due to its strong convexity, we can use sub-gradient descent [11] to search for the optimal transfer parameter. We show that we can find the ϵ-accurate solution by using only $O\left(\frac{\ln \epsilon}{\epsilon}\right)$ iterations. In each iteration, we only have to solve

Theorem 3.6.1 (Converge Property) *Let $L(\gamma)$ be a λ-strongly convex function in* (3.35) *and γ^* be its optimal solution. Let $\gamma_1, ..., \gamma_{T+1}$ be a sequence such that $\gamma_1 \in B$ and for $t > 1$, we have $\gamma_{t+1} = \gamma_t - \eta_t \Delta_t$, where Δ_t is the sub-gradient of $L(\gamma_t)$ and $\eta_t = 1/(\lambda t)$. Assume we have $||\Delta_t|| \leq G$ for all t. Then we have:*

$$L(\gamma_{T+1}) \leq L(\gamma^*) + \frac{G^2(1 + \ln(T))}{2\lambda T} \tag{3.37}$$

We show the detail proof in Appendix A.3.

By adding a dual set of variables, one for each constraint, we get the Lagrangian of the optimization problem (3.35):

$$\begin{aligned} \textbf{max} \qquad & L(\gamma, \epsilon, \eta) = \frac{\lambda}{2} \sum_{n=1}^{N} \|\gamma_n\|^2 + \sum_{i=1}^{l} \epsilon_i \\ & + \sum_{i,n} \eta_{i,n} \left[1 - \varepsilon_{ny_i} + \hat{Y}_{in}(\gamma) - \hat{Y}_{iy_i}(\gamma) - \epsilon_i\right] \\ \textbf{s.t.} \qquad & \forall i, n \quad \eta_{i,n} \geq 0 \end{aligned} \tag{3.38}$$

The sub-gradient of can be written as:

$$\Delta_\gamma = \begin{cases} \mathbf{0} & y_i = n \\ \left[0, .., \frac{\alpha''_{in}}{S_{ii}^{-1}}, .., -\frac{\alpha''_{iy_i}}{S_{ii}^{-1}}, .., 0\right] & y_i, n = 1, ...N \end{cases}$$

We call this algorithm Safety MultI-class Transfer Learning (SMITLe) and show the algorithm details in Alg. 1.

In summary, in this section, we proposed a novel objective function based on LOO-CV error. By adding the L2 regularization terms we show that we can use sub-gradient descent to find the ϵ-accurate solution by using only $O\left(\frac{\ln \epsilon}{\epsilon}\right)$ iterations. In the next section, we show some empirical results on the benchmark dataset.

Algorithm 1 SMITLe

Input: $\lambda, \mathcal{S}, \alpha', \alpha'', T,$

Output: $\gamma = \left\{\gamma^1, ..., \gamma^n\right\}$

$\gamma^0 \leftarrow 1$
for $t = 1$ to T **do**
  $\hat{Y} \leftarrow Y - (\mathcal{S} \circ I)^{-1} (\alpha' - \alpha'' D(\gamma))$
  $\Delta_\gamma = 0, \Delta_\beta = 0$
  for $i = 1$ to l **do**
    $\Delta_\gamma \leftarrow \Delta_\gamma + \lambda_1 \gamma$
    for $r = 1$ to $N + 1$ **do**
      $l_{ir} = 1 - \varepsilon_{y_i r} + \hat{Y}_{ir} - \hat{Y}_{iy_i}$
      if $l_{ir} > 0$ **then**
        $\Delta_\gamma^{y_i} \leftarrow \Delta_\gamma^{y_i} - \frac{\alpha''_{iy_i}}{\mathcal{S}_{ii}^{-1}}$
      end if
    end for
  end for
  $\gamma^t \leftarrow \gamma^{(t-1)} - \frac{\Delta_\gamma}{\lambda \times t}$
end for

3.7 Experiment

In this section, we show the empirical result on some benchmarks.to write there.

3.7.1 Dataset

In this subsection, we introduce the datasets we use in this chapter. to write there.

MNIST

The MNIST database [39] (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by ”re-mixing” the samples from NIST’s original datasets. The creators felt that since NIST’s training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, NIST’s complete dataset was too hard. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale

(a) MNIST (b) USPS

Figure 3.7: Some examples of MNIST & USPS.

levels.

In our experiment, we use a sub-set of MNIST dataset, containing 6,000 examples for 10 classes (from digit 0 to 9). We show the class distribution

Table 3.2: Data distribution of MNIST subset

digit	# examples	percentage	digit	# examples	percentage
1	583	9.7%	6	511	8.5%
2	686	11.4%	7	602	10.0%
3	590	9.8%	8	609	10.2%
4	652	10.9%	9	607	10.1%
5	564	9.4%	10	596	9.9%

USPS

We also use another handwritten digital dataset USPS [28]. USPS contains 11,000 images and the data is evenly distributed among 10 classes, i.e. 1,100 examples for each digit. Each digit is represented as a 16x16 greyscale image.

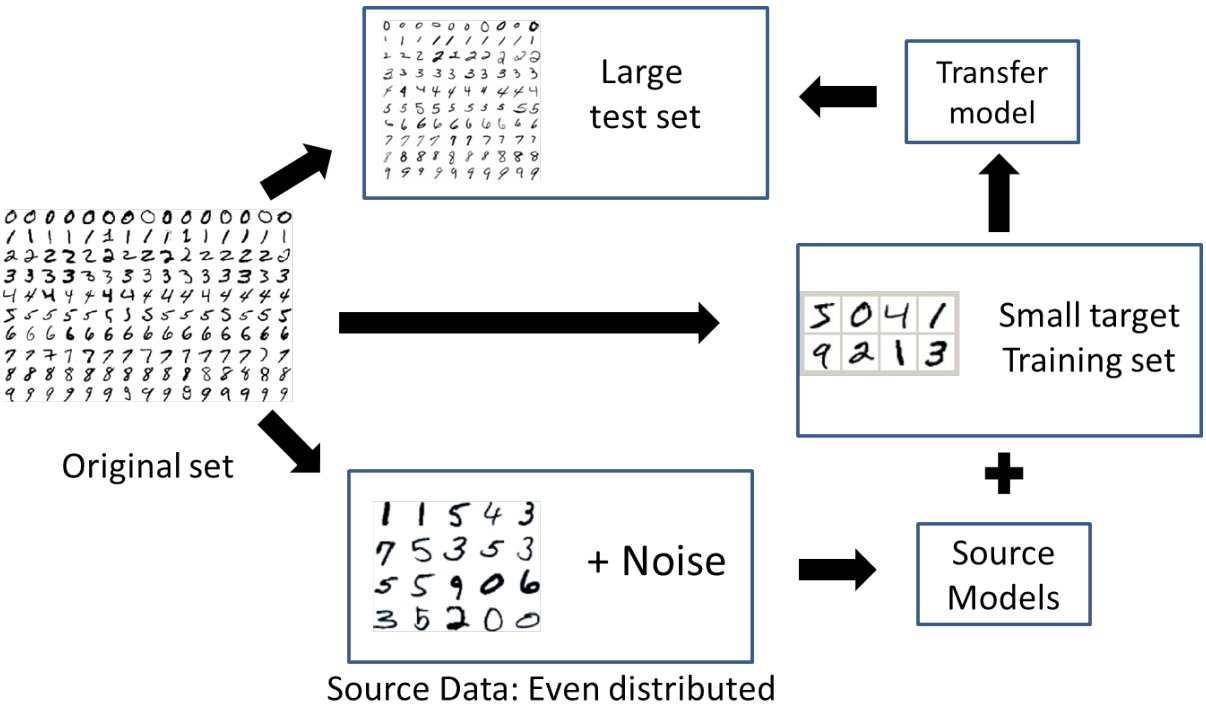


Figure 3.8: Illustration of our training procedure. The data is splitted into 3 sets: a source training set, a small target training set and a large test set. Salt & pepper noise is added into the source training set in order to generate different source models.

3.7.2 Experiment Setup

In this part, we introduce the experiment set up on the two datasets. We perform experiment on different settings to simulate different scenarios for transfer learning.

As we discussed in Section 3.3, previous work [5] show that the more similar of the source model(s) and the target task are, the more improvement the target model can get from the source model(s). When the source model and the target task are unrelated, negative transfer could happen. In order to test the robustness of our algorithm, we should assign the source model with different $\mathcal{D}$-relationship to the target task and test how our algorithm performs, especially when negative transfer could happen. To generate the source model(s) with different $\mathcal{D}$-relationship to the target task, we first split the dataset into 3 sub-sets: source training set, target training set and test set. Then we add the noise to the source training set and train the source model(s) from it. Therefore, as we add more noise to the source training set, the source models are more unrelated to the target task.

(a) Noise Rate = 0

(b) Noise Rate = 0.3

(c) Noise Rate = 0.5

(d) Noise Rate = 0.8

Figure 3.9: Images with different level of noise rate. By adding more slat&pepper noise, the images become less clear.

Table 3.3: Setups for our experiment on two datasets

	MNIST	USPS
noise rate	0,0.2,0.3,0.5,0.8,0.9	
source training set size	100	
target training set size	5,10,15,20,25	
test set size	4700	9700

Bibliography

[1] MA Aizerman, E Braverman, and LI Rozonoer. Probability problem of pattern recognition learning and potential functions method. *Avtomat. i Telemekh*, 25(9):1307–1323, 1964.

[2] Yusuf Aytar and Andrew Zisserman. Tabula rasa: Model transfer for object category detection. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2252–2259. IEEE, 2011.

[3] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *The Journal of Machine Learning Research*, 4:83–99, 2003.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[5] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[6] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[7] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 232–237. IEEE, 1998.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[9] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[12] Gavin C Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1661–1668. IEEE, 2006.

[13] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[14] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223, 2011.

[15] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[16] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine learning*, 47(2-3):201–233, 2002.

[17] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[18] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[20] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[21] Jesse Davis and Pedro Domingos. Deep transfer via second-order markov logic. In *Proceedings of the 26th annual international conference on machine learning*, pages 217–224. ACM, 2009.

[22] A Evgeniou and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007.

[23] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[24] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):594–611, 2006.

[25] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[26] Judy Hoffman, Eric Tzeng, Jeff Donahue, Yangqing Jia, Kate Saenko, and Trevor Darrell. One-shot adaptation of supervised deep convolutional models. *arXiv preprint arXiv:1312.6204*, 2013.

[27] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[28] Jonathan J Hull. A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(5):550–554, 1994.

[29] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pages 264–271, 2007.

[30] Luo Jie, Tatiana Tommasi, and Barbara Caputo. Multiclass transfer learning from unconstrained priors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1863–1870. IEEE, 2011.

[31] Søren Johansen and Katarina Juselius. Maximum likelihood estimation and inference on cointegrationwith applications to the demand for money. *Oxford Bulletin of Economics and statistics*, 52(2):169–210, 1990.

[32] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[33] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[35] Gregory Kuhlmann and Peter Stone. Graph-based domain mapping for transfer learning in general games. In *Machine Learning: ECML 2007*, pages 188–200. Springer, 2007.

[36] Ilja Kuzborskij and Francesco Orabona. Stability and hypothesis transfer learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 942–950, 2013.

[37] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. From n to n+ 1: Multiclass transfer incremental learning. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3358–3365. IEEE, 2013.

[38] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] Xuejun Liao, Ya Xue, and Lawrence Carin. Logistic regression with an auxiliary data source. In *Proceedings of the 22nd international conference on Machine learning*, pages 505–512. ACM, 2005.

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

[42] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France*, pages 97–105, 2015.

[43] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[44] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: a survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[45] Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. *arXiv preprint arXiv:1503.01558*, 2015.

[46] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[47] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[48] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[49] John Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[50] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[51] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

[52] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Transfer Learning*, volume 898, 2005.

[53] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE, 2004.

[54] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[55] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[56] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[58] Erik Talvitie and Satinder P Singh. An experts algorithm for transfer learning. In *IJCAI*, pages 1065–1070, 2007.

[59] Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3081–3088. IEEE, 2010.

[60] Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. Learning categories from few examples with multi model knowledge transfer. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(5):928–941, 2014.

[61] Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[62] Lisa Torrey, Trevor Walker, Jude Shavlik, and Richard Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Machine Learning: ECML 2005*, pages 412–424. Springer, 2005.

[63] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[64] Joost Van De Weijer and Cordelia Schmid. Coloring local feature extraction. In *Computer Vision–ECCV 2006*, pages 334–348. Springer, 2006.

[65] Vladimir Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. *Neural computation*, 12(9):2013–2036, 2000.

[66] Xuezhi Wang, Tzu-Kuo Huang, and Jeff Schneider. Active transfer learning under model shift. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1305–1313, 2014.

[67] Jianchao Yang, Kai Yu, Yihong Gong, and Tingwen Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.

[68] Jun Yang, Rong Yan, and Alexander G Hauptmann. Adapting svm classifiers to data with shifted distributions. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 69–76. IEEE, 2007.

[69] Jun Yang, Rong Yan, and Alexander G Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th international conference on Multimedia*, pages 188–197. ACM, 2007.

[70] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[71] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[72] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014.

[73] Tianyi Zhou and Dacheng Tao. Multi-task copula by sparse graph regression. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 771–780. ACM, 2014.

Appendix A

Proofs of Theorems

A.1 Proof of Theorem1

Proof For simplification, let $\delta_i = 1$ if $i = N + 1$ and 0 otherwise, and $\theta_{ij} = \alpha''_{ij}\left(1 - \delta_j\right)/\psi_{ii}^{-1}$. Eq. (**??**) can be written as:

$$\xi_i(\gamma,\beta) = \max_n \left\{ \varepsilon_{ny_i} - 1 + \frac{\left(\alpha'_{iy_i} - \alpha'_{in}\right)}{\psi_{ii}^{-1}} + \theta_{in}\gamma_n - \theta_{iy_i}\gamma_{y_i} + \left(\delta_n - \delta_{y_i}\right)\sum_k \frac{\alpha''_{ik}\beta_k}{\psi_{ii}^{-1}} \right\} \tag{A.1}$$

When $\gamma = \beta = \mathbf{0}$, from Eq. (A.1) we can get:

$$\bar{\xi}_i = \max_n \left[\varepsilon_{ny_i} - 1 + \frac{\left(\alpha'_{iy_i} - \alpha'_{in}\right)}{\psi_{ii}^{-1}} \right]$$

To obtain the optimal value of γ and β, we have to seek the saddle point of the Lagrangian problem in (**??**) by finding the minimum for the prime variables $\{\gamma, \beta, \xi\}$ and the maximum for the dual variables η. To find the minimum of the primal problem, we require:

$$\frac{\partial L}{\partial \xi_i} = 1 - \sum_n \eta_{in} = 0 \rightarrow \sum_n \eta_{in} = 1$$

Similarly, for γ and β, we require:

$$\begin{aligned} \frac{\partial L}{\partial \gamma_n} &= \lambda_1\gamma_n + \sum_i \eta_{in}\theta_{in} - \sum_{i,n=y_i}\left(\sum_q \eta_{iq}\right)\theta_{in}\gamma_n \\ &=_1 \lambda_1\gamma_n + \sum_i \eta_{in}\theta_{in} - \sum_i \varepsilon_{ny_i}\theta_{in} = 0 \\ &\Rightarrow \gamma_n^* = \frac{1}{\lambda_1}\sum_i \left(\varepsilon_{ny_i} - \eta_{in}\right)\theta_{in} \end{aligned} \tag{A.2}$$

In $=_1$ we use the facts that $\sum_n \eta_{in} = 1$ and use ε_{ny_i} to replace it.

$$\begin{aligned} \frac{\partial L}{\partial \beta_n} &= \lambda_2 \beta_n + \left[\sum_{i,n} \frac{\eta_{in} \alpha''_{in}}{\psi_{ii}^{-1}} \left(\delta_n - \delta_{y_i} \right) \right] = 0 \\ &\Rightarrow \quad \beta_n^* = \frac{1}{\lambda_2} \sum_{i,n} \frac{\eta_{in} \alpha''_{in}}{\psi_{ii}^{-1}} \left(\delta_{y_i} - \delta_n \right) \end{aligned} \tag{A.3}$$

As the strong duality holds,the primal and dual objectives coincide. Plug Eq (A.2) and (A.3) into Eq. (**??**), we have:

$$\sum_{i,n} \eta_{in} \left[1 - \varepsilon_{ny_i} + \hat{Y}_{in} (\gamma^*, \beta^*) - \hat{Y}_{iy_i} (\gamma^*, \beta^*) - \xi_i^* \right] = 0$$

Expand the equation above, we have:

$$\sum_{i,n} \eta_{in} \left[\varepsilon_{n,y_i} - 1 + \frac{\left(\alpha'_{iy_i} - \alpha'_{in} \right)}{\psi_{ii}^{-1}} - \xi_i \right] = \lambda_1 \sum_r \left\| \gamma_r^* \right\|^2 + \lambda_2 \sum_r \left\| \beta_r^* \right\|^2 \geq 0$$

Rearranging the above, we obtain:

$$\sum_{i,n} \eta_{in} \left[\varepsilon_{n,y_i} - 1 + \frac{\left(\alpha'_{iy_i} - \alpha'_{in} \right)}{\psi_{ii}^{-1}} \right] \geq \sum_{i,n} \eta_{in} \xi_i = \sum_i \xi_i \tag{A.4}$$

The left-hand side of Inequation (A.4) can be bounded by:

$$\begin{aligned} &\sum_{i,n} \eta_{in} \left[\varepsilon_{ny_i} - 1 + \frac{\left(\alpha'_{iy_i} - \alpha'_{in} \right)}{\psi_{ii}^{-1}} \right] \\ &\leq \sum_i \left(\sum_n \eta_{in} \max_r \left\{ \varepsilon_{ry_i} - 1 + \frac{\left(\alpha'_{iy_i} - \alpha'_{ir} \right)}{\psi_{ii}^{-1}} \right\} \right) \\ &= \sum_i \left(\sum_n \eta_{in} \bar{\xi}_i \right) = \sum_i \bar{\xi}_i \end{aligned} \tag{A.5}$$

A.2 Proof of Theorem 3.4.1

Theorem 3.4.1 *Given a dataset $D = \{(x_i, y_i) | i = 1, ..., l\}$, the solution of a LS-SVM on D can be written as:*

$$\begin{bmatrix} K + \frac{1}{C} \mathrm{I} & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{A.6}$$

Assume that $D^{(n)} = \{(x_i, y_i) | i = 1, ..., n\}$ is a subset of D and $D \backslash D^{(n)}$ is the complement of $D^{(n)}$ in D , The unbiased leave out error of a LS-SVM trained from $D \backslash D^{(n)}$ on $D^{(n)}$ can be estimated as:

$$ERR_{leave-out} = \left(S_n - sS_{(l-n+1)}^{-1}s^T\right)[\alpha_1, ..., \alpha_n]^T$$

Where $[\alpha_1, ..., \alpha_n]$ *is the first n rows of* α *in* (A.6). S_n, s *and* $S_{(l-n+1)}$ *are the square blocks of matrix:*

$$\left[\begin{array}{c|c} S_n & s \\ \hline s^T & S_{(l-n+1)} \end{array}\right] = \begin{bmatrix} K + \frac{1}{C}\mathrm{I} & 1 \\ 1^T & 0 \end{bmatrix}$$

Proof Following the result of Eq. (A.6) and noticing that the matrix of the left hand in Eq. (A.6) is symmetrical, it can be written as follow:

$$\begin{bmatrix} K + \frac{1}{C}\mathrm{I} & 1 \\ 1^T & 0 \end{bmatrix} = \left[\begin{array}{c|c} S_n & s \\ \hline s^T & S_{(l-n+1)} \end{array}\right] \tag{A.7}$$

Where $S_n \in R^{n\times n}$, $s \in R^{n\times(l-n+1)}$ and $S_{(l-n+1)} \in R^{(l-n+1)\times(l-n+1)}$.

For the first round of the cross validation situation, assume the first n examples are used as the validation set. In this case, let $\left[\alpha^{-1}, b^{-1}\right] \in R^{l-n+1}$ denote the optimal parameters for a LS-SVM f^{-1} trained on the rest of the samples and they can be found by:

$$\begin{bmatrix} \alpha^{-1} \\ b^{-1} \end{bmatrix} = \mathrm{S}_{(l-n+1)}^{-1}[y_{n+1}, ..., y_l, 0]^T \tag{A.8}$$

The prediction of f^{-1} on the validation set $\hat{Y} = [\hat{y}_1, ..., \hat{y}_n]$ is given by:

$$[\hat{y}_1, ..., \hat{y}_n]^T = s\begin{bmatrix} \alpha^{-1} \\ b^{-1} \end{bmatrix} = sS_{(l-n+1)}^{-1}[y_{n+1}, ..., y_l, 0]^T \tag{A.9}$$

Moreover, the last $l - n + 1$ rows in Eq. (A.6) can be represented as $\begin{bmatrix} s^T & S_{l-n+1} \end{bmatrix}[\alpha, b]^T = [y_{n+1}, ..., y_l, 0]^T$. So

$$\begin{aligned} \hat{Y} = [\hat{y}_1, ..., \hat{y}_n]^T &= sS_{(l-n+1)}^{-1}\begin{bmatrix} s^T & S_{l-n+1} \end{bmatrix}\begin{bmatrix} \alpha \\ b \end{bmatrix} \\ &= sS_{(l-n+1)}^{-1}s^T[\alpha_1, ..., \alpha_n]^T + s[\alpha_{n+1}, ..., \alpha_l, b]^T \end{aligned} \tag{A.10}$$

Then first n rows in Eq. (A.6) can be represented as:

$$Y = [y_1, ...y_n]^T = \begin{bmatrix} S_n & s \end{bmatrix}\begin{bmatrix} \alpha \\ b \end{bmatrix} = S_n[\alpha_1, ..., \alpha_n]^T + s[\alpha_{n+1}, ..., \alpha_l, b]^T \tag{A.11}$$

Thus, combining (A.10) and (A.11), we have the following equation:

$$\begin{aligned}\hat{Y} &= Y - S_n[\alpha_1, ..., \alpha_n]^T + sS^{-1}_{(l-n+1)}s^T[\alpha_1, ..., \alpha_n]^T \\ &= Y - \left(S_n - sS^{-1}_{(l-n+1)}s^T\right)[\alpha_1, ..., \alpha_n]^T\end{aligned} \tag{A.12}$$

According to block matrix inversion lemma

$$\begin{bmatrix} S_n & s \\ s^T & S_{(l-n+1)} \end{bmatrix}^{-1} = \begin{bmatrix} \kappa^{-1} & -\kappa^{-1}sS^{-1}_{(l-n+1)} \\ -S_{(l-n+1)}s^T\kappa^{-1} & S^{-1}_{(l-n+1)} + S^{-1}_{(l-n+1)}s^T\kappa^{-1}sS^{-1}_{(l-n+1)} \end{bmatrix} \tag{A.13}$$

Where $\kappa = \left(S_n - sS^{-1}_{(l-n+1)}s^T\right)$. We have:

$$Y - \hat{Y} = \kappa[\alpha_1, ..., \alpha_n]^T \tag{A.14}$$

A.3 Proof of Theorem 3.6.1

Theorem 3.6.1 *Let $L(\gamma)$ be a λ-strongly convex function in* (3.35) *and γ^* be its optimal solution.Let $\gamma_1, ..., \gamma_{T+1}$ be a sequence such that $\gamma_1 \in B$ and for $t > 1$, we have $\gamma_{t+1} = \gamma_t - \eta_t\Delta_t$, where Δ_t is the sub-gradient of $L(\gamma_t)$ and $\eta_t = 1/(\lambda t)$. Assume we have $\|\Delta_t\| \leq G$ for all t. Then we have:*

$$L(\gamma_{T+1}) \leq L(\gamma^*) + \frac{G^2(1 + \ln(T))}{2\lambda T} \tag{A.15}$$

Proof As $L(\gamma)$ is strongly convex and Δ_t is in its sub-gradient set at γ_t, according to the definition of λ-strong convexity [51], the following inequality holds:

$$\langle \gamma_t - \gamma^*, \Delta_t \rangle \geq L(\gamma_t) - L(\gamma^*) + \frac{\lambda}{2}\|\gamma_t - \gamma^*\|^2 \tag{A.16}$$

For the term $\left\langle \gamma_t - \gamma^*, \Delta_y \right\rangle$, it can be written as:

$$\begin{aligned}\langle \gamma_t - \gamma^*, \Delta_t \rangle &= \left\langle \gamma_t - \frac{1}{2}\eta_t\Delta_t + \frac{1}{2}\eta_t\Delta_t - \gamma^*, \Delta_t \right\rangle \\ &= \frac{1}{2}\left\langle [(\gamma_t - \eta_t\Delta_t) - \gamma^*] + (\gamma_t - \gamma^*) + \eta_t\Delta_t, \Delta_t \right\rangle \\ &= \frac{1}{2}\left\langle (\gamma_{t+1} - \gamma^*) + (\gamma_t - \gamma^*), \Delta_t \right\rangle + \frac{1}{2}\eta_t\Delta_t^2 \\ &= \frac{1}{2}\left\langle \gamma_{t+1} + \gamma_t - 2\gamma^*, \Delta_t \right\rangle + \frac{1}{2}\eta_t\Delta_t^2\end{aligned} \tag{A.17}$$

Then we have:

$$\begin{aligned}\|\gamma_t - \gamma^*\|^2 - \|\gamma_{t+1} - \gamma^*\|^2 &= (\gamma_t - \gamma_{t+1})(\gamma_t + \gamma_{t+1} - 2\gamma^*) \\ &= \langle \gamma_{t+1} + \gamma_t - 2\gamma^*, \eta_t\Delta_t \rangle\end{aligned} \tag{A.18}$$

Using the assumption $\|\Delta_t\| \leq G$, we can rearrange (A.16) and plug (A.17) and (A.18) into it, we have:

$$\begin{aligned} Diff_t = L(\gamma_t) - L(\gamma^*) &\leq \frac{\|\gamma_t - \gamma^*\|^2 - \|\gamma_{t+1} - \gamma^*\|^2}{2\eta_t} - \frac{\lambda}{2}\|\gamma_t - \gamma^*\|^2 + \frac{1}{2}\eta_t\Delta_t^2 \\ &\leq \frac{\|\gamma_t - \gamma^*\|^2 - \|\gamma_{t+1} - \gamma^*\|^2}{2\eta_t} - \frac{\lambda}{2}\|\gamma_t - \gamma^*\|^2 + \frac{1}{2}\eta_t G^2 \\ &= \frac{\lambda(t-1)}{2}\|\gamma_t - \gamma^*\|^2 - \frac{\lambda t}{2}\|\gamma_{t+1} - \gamma^*\|^2 + \frac{1}{2}\eta_t G^2 \end{aligned} \tag{A.19}$$

Due to the strong convexity, for each pair of $L(\gamma_t)$ and $L(\gamma_{t+1})$ for $t = 1, ..., T$, according to (A.16), we have:

$$\begin{aligned} L(\gamma_{t+1}) - L(\gamma_t) &\leq \langle \gamma_{t+1} - \gamma_t, \Delta_t \rangle - \frac{\lambda}{2}\|\gamma_{t+1} - \gamma_t\|^2 \\ &= -\eta_t\Delta_t^2(1 - \frac{1}{2t}) \leq 0 \end{aligned} \tag{A.20}$$

Therefore, we have the following sequence $L(\gamma^*) \leq L(\gamma_T) \leq L(\gamma_{T-1}) \leq ... \leq L(\gamma_1)$. For the sequence $Diff_t$ for $t = 1, ..., T$, we have:

$$\sum_{t=1}^{T} Diff_t = \sum_{t=1}^{T} L(\gamma_t) - TL(\gamma^*) \geq T\left[L(\gamma_T) - L(\gamma^*)\right] \tag{A.21}$$

Next, we show that

$$\begin{aligned} \sum_{t=1}^{T} Diff_t &= \sum_{t=1}^{T}\left\{\frac{\lambda(t-1)}{2}\|\gamma_t - \gamma^*\|^2 - \frac{\lambda t}{2}\|\gamma_{t+1} - \gamma^*\|^2 + \frac{1}{2}\eta_t G^2\right\} \\ &= -\frac{\lambda T}{2}\|\gamma_{T+1} - \gamma^*\|^2 + \frac{G^2}{2\lambda}\sum_{t=1}^{T}\frac{1}{t} \\ &\leq \frac{G^2}{2\lambda}\sum_{t=1}^{T}\frac{1}{t} \leq \frac{G^2}{2\lambda}(1 + \ln(T)) \end{aligned} \tag{A.22}$$

Combining (A.21) and rearranging the result, we have:

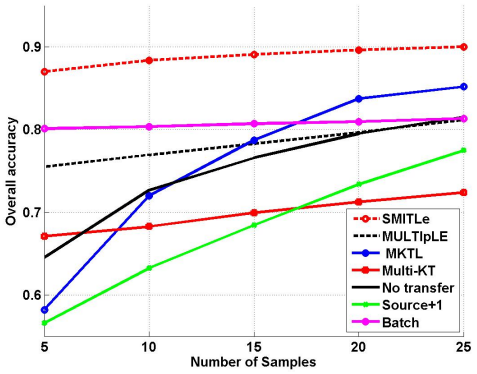
$$L(\gamma_{T+1}) \leq L(\gamma^*) + \frac{G^2(1 + \ln(T))}{2\lambda T}$$

Appendix B

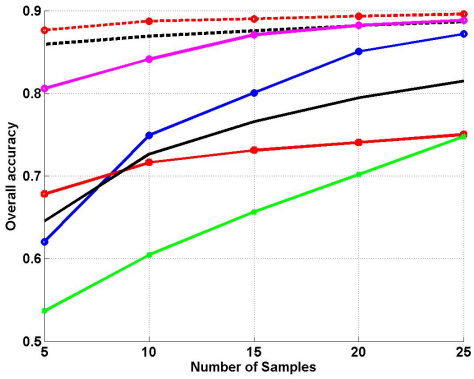
Detailed Experiment Results

B.1 Detailed result on MNIST

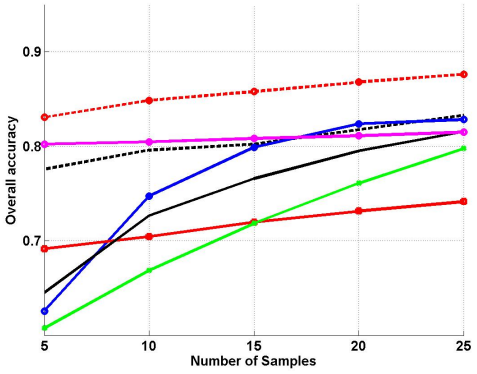
B.1.1 No Noise



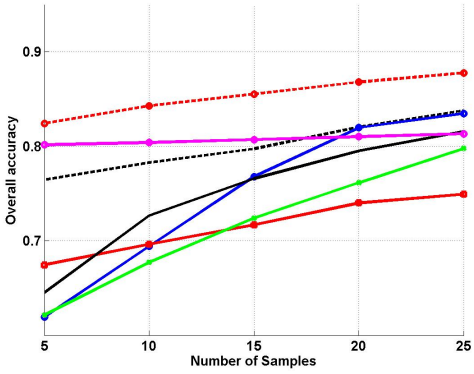
class 1



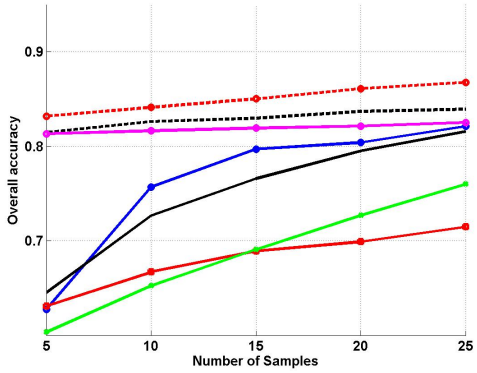
class 2



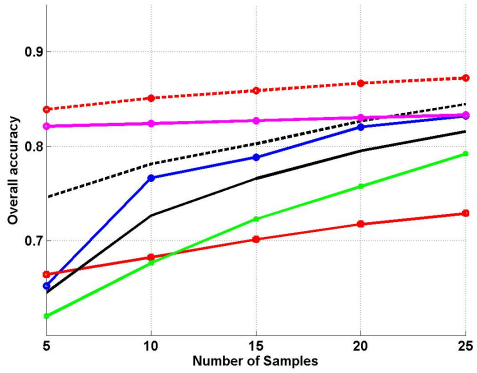
class 3



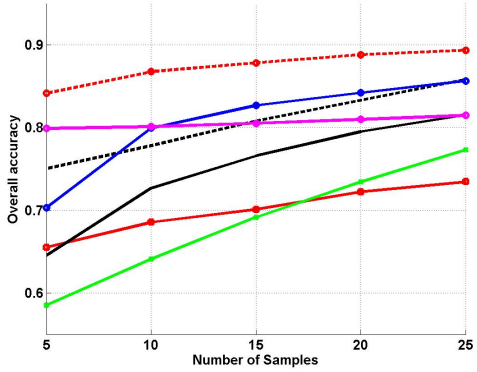
class 4



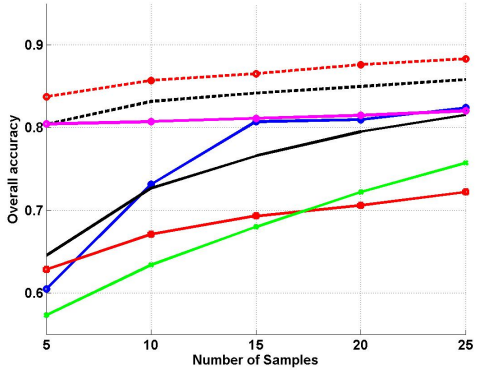
class 5



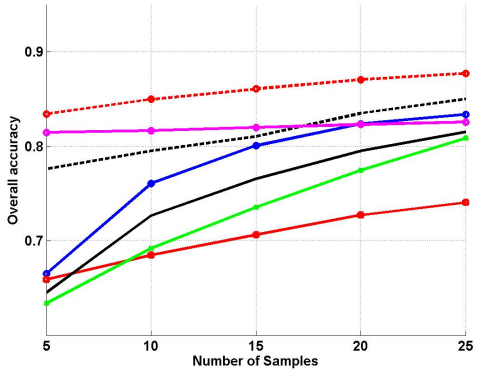
class 6



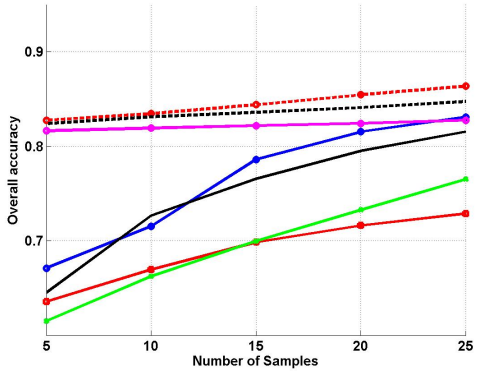
class 7



class 8

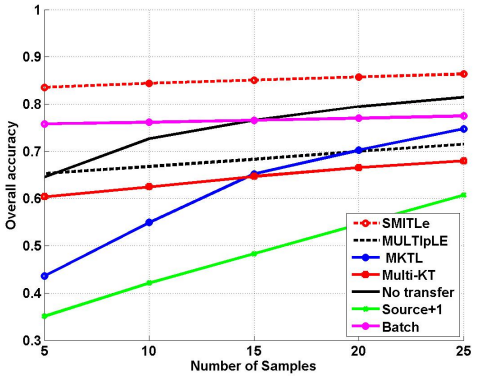


class 9

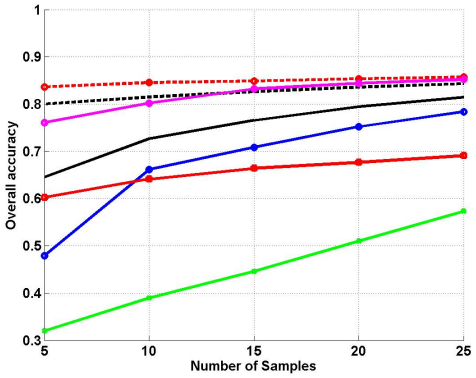


class 10

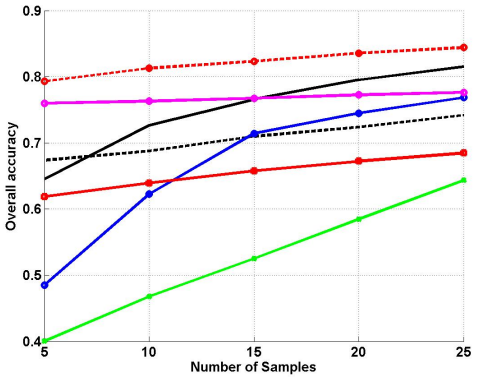
B.1.2 0.3 Noise



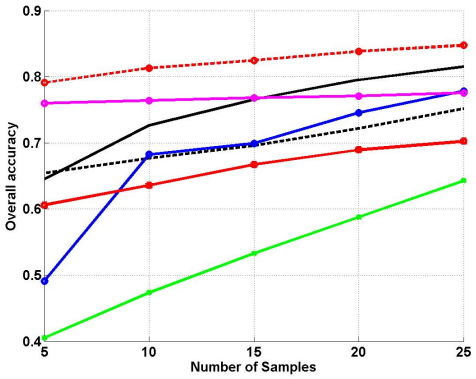
class 1



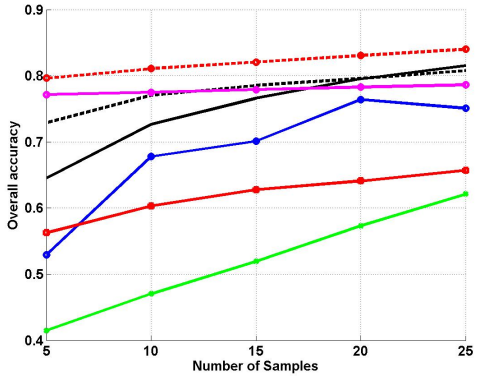
class 2



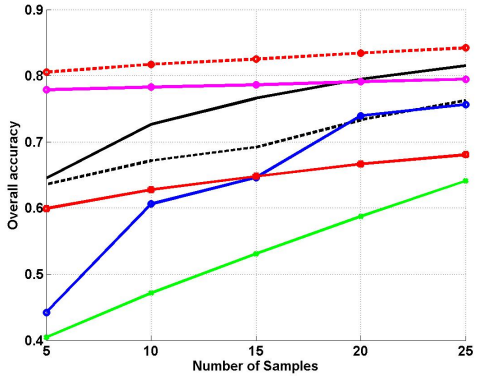
class 3



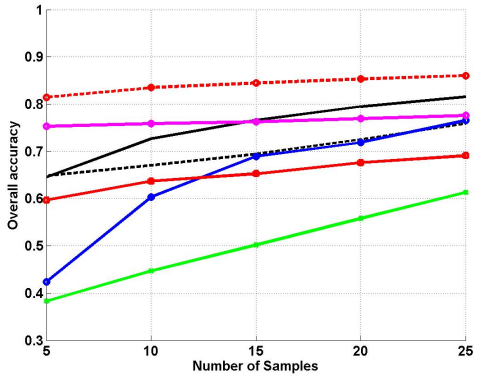
class 4



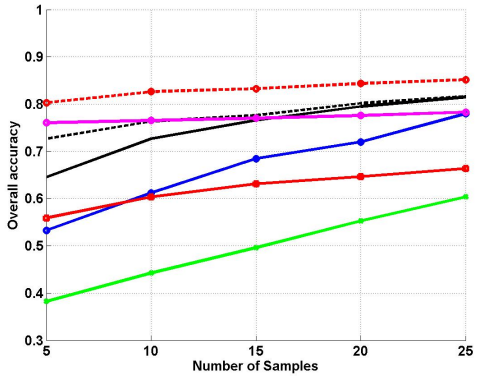
class 5



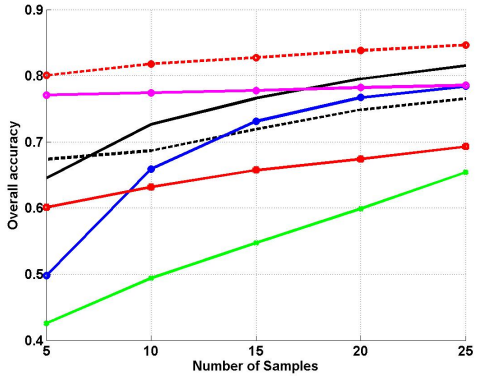
class 6



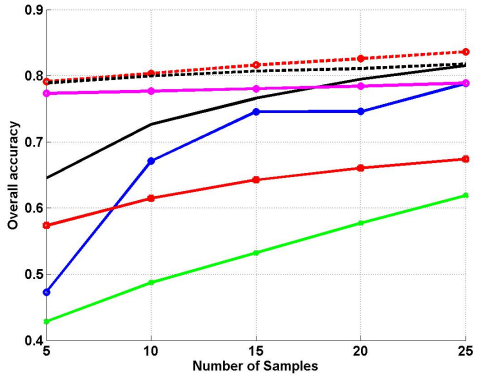
class 7



class 8

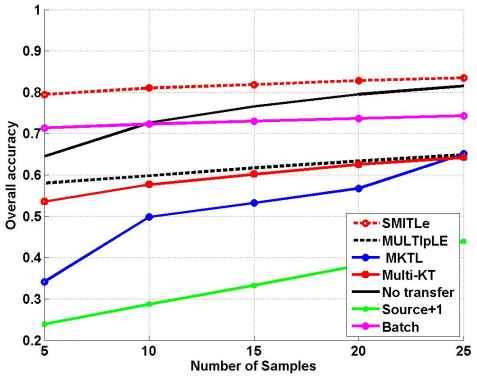


class 9

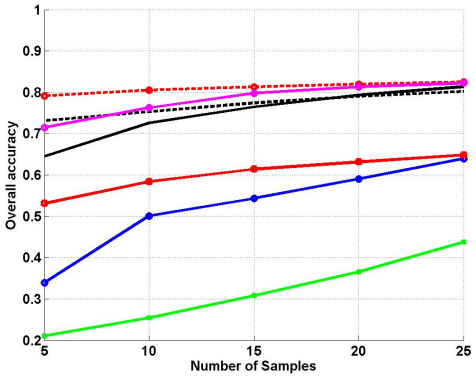


class 10

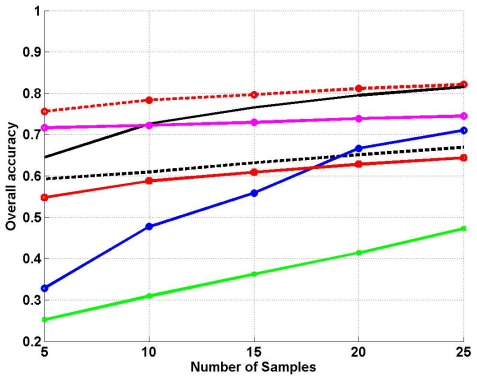
B.1.3 0.5 Noise



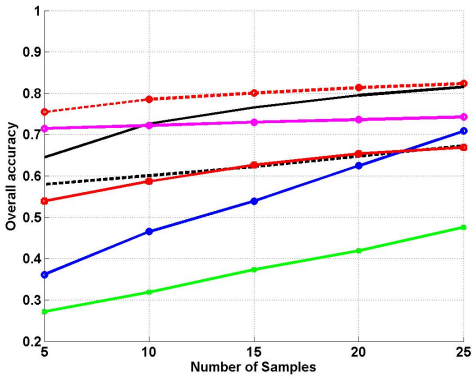
class 1



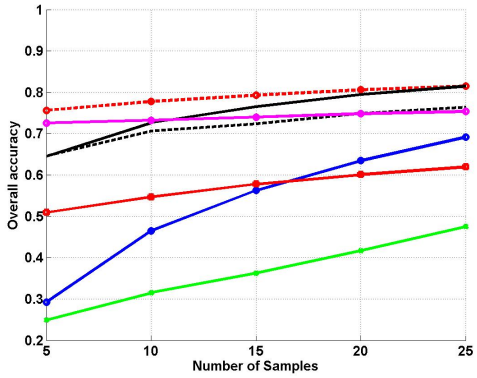
class 2



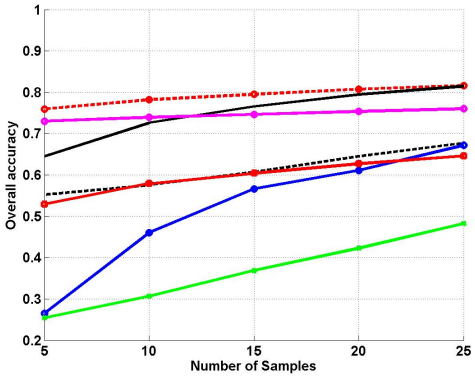
class 3



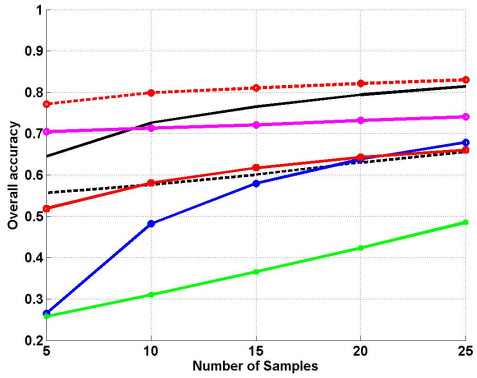
class 4



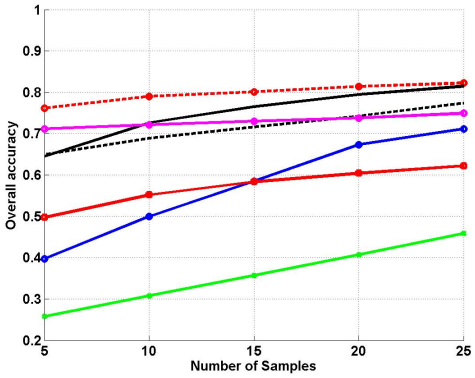
class 5



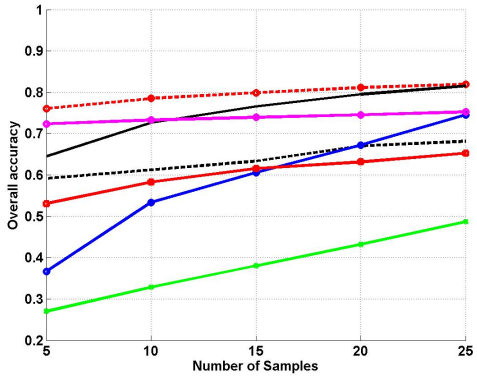
class 6



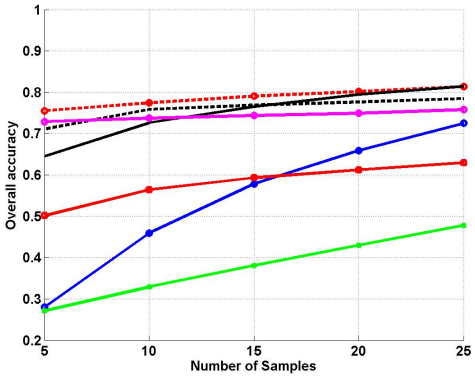
class 7



class 8



class 9



class 10

Curriculum Vitae

Name:	Shuang Ao
Post-Secondary Education and Degrees:	La La School La La Land 1996 - 2000 M.A. University of Western Ontario London, ON 2008 - 2012 Ph.D.
Honours and Awards:	NSERC PGS M 2006-2007
Related Work Experience:	Teaching Assistant The University of Western Ontario 2008 - 2012

Publications:

La La