Dynamic Transfer Learning for One-class Classification : a Multi-task Learning Approach

Yongjian Xue

Thèse
de doctorat
de l'UTT

Yongjian XUE

Dynamic Transfer Learning for One-class Classification: a Multi-task Learning Approach

Spécialité :
Optimisation et Sûreté des Systèmes

2018TROY0006

Année 2018

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Yongjian XUE

le 4 avril 2018

Dynamic Transfer Learning for One-class Classification: a Multi-task Learning Approach

JURY

M. G. GASSO	PROFESSEUR DES UNIVERSITES	Président (Rapporteur)
M. P. BEAUSEROY	PROFESSEUR DES UNIVERSITES	Directeur de thèse
Mme É. GRALL	MAITRE DE CONFERENCES - HDR	Examinateur
M. Y. GRANDVALET	DIRECTEUR DE RECHERCHE CNRS	Rapporteur
M. H. SAHBI	CHARGE DE RECHERCHE CNRS	Examinateur
M. D. R. TOMASSI	PROFESOR ADJUNTO	Examinateur

Personnalité invitée

M. V. VRABIE	MAITRE DE CONFERENCES - HDR

Université de Technologie de Troyes

Abstract

Institut Charles Delaunay/LM2S

Doctor of Philosophy

Dynamic Transfer Learning for One-class Classification: A Multi-task Learning Approach

by Yongjian XUE

The aim of this thesis is to minimize the performance loss of a one-class detection system when it encounters a data distribution change. The idea is to use transfer learning approach to transfer learned information from related old task to the new one. According to the practical applications, we divide this transfer learning problem into two parts, one part is the transfer learning in homogenous space and the other part is in heterogeneous space.

A multi-task learning model is proposed to solve the above problem, it uses one parameter to balance the amount of information brought by the old task versus the new task. This model is formalized so that it can be solved by classical one-class SVM except with a different kernel matrix. To select the control parameter, a kernel path solution method is proposed. It computes all the solutions along that introduced parameter and corresponding criteria are proposed to choose the optimal solution at given number of new samples. Experiments show that this model can give a smooth transition from the old detection system to the new one whenever it encounters a data distribution change.

Moreover, as the proposed model can be solved by classical one-class SVM, online learning algorithms for one-class SVM are studied later in the purpose of getting a constant false alarm rate. It can be applied to the online learning of the proposed model directly.

Keywords: transfer learning, multi-task learning, one-class classification, outliers detection, one-class SVM.

Acknowledgements

First of all I would like to express my sincere and grateful thanks to my supervisor Pierre Beauseroy, for his excellent guidance, wonderful support and continuous patience during the last three years and a half. I would like to thank him for providing me valuable advices and discussions every week I met him. I am really appreciated for his extensive expertise in machine learning and system detection. It has been a great pleasure working with him for my PhD degree.

Next, I would like to thank the committee members of my defense. Especially thank the reviewers, Gilles GASSO and Yves GRANDVALET, who have accepted and spent their time to read the manuscript and give me valuable reports. I also would like to thank the examiners Edith GRALL, Hichem SAHBI, Diego TOMASSI and Valeriu VRABIE, for accepting to be my defense committee and give me helpful and insightful feedback.

A particular thanks to Regis LENGELLE and Khemais SAANOUNI, the former director and the present director of the UTT doctoral school. And thank the secretaries of doctoral school, they are Isabelle LECLERCQ, Pascale DENIS and Thérèse KAZARIAN. I also would like to thank all the members of LM2S for their help and friendship. Thank the secretaries, Bernadette ANDRE and Veronique BANSE for their kindly help and availability support. Thank my colleagues Nassara ELHADJI, Fei ZHU, Irce FERNANDES GOMES GUIMARAES, Sofia MARINO, Xiaoyi CHEN, Nan ZHANG. I also like to take this opportunity to thank all my friends, with whom I shared wonderful and unforgettable moments in UTT and in France.

I greatly thank my mother and my father for their unconditional love and support, which encourage me to work hard and purse what I want.

Contents

List of Figures

List of Tables

1 Introduction

In this chapter, we will introduce the research problem of this thesis. More specifically, In section 1.1, we define what is one-class classification, what is transfer learning, and then we introduce the research problem of this thesis: transfer learning for one-class classification. Following that research problem statement, we give a general thesis contributions in section 1.2 and list the publications resulted in by this thesis in section 1.3. At last, we outline the whole thesis in section 1.4.

1.1 Problem statement

1.1.1 One-class classification

In many applications, we need to build corresponding monitoring systems to know if they are under good conditions and if they are working normally or if there are some abnormal behaviors that we need to be alerted. For example, in the case of fraud detection, we need detection system to ensure the security of online payment, that is to say for every transaction, the detection system will either classify it as legal to support the transaction or illegal to prevent the behavior. Generally, we can define these behaviors as inliers (normal events) or outliers (abnormal events).

Definition 1.1 (**Outlier**, **inlier**)**.** An outlier (abnormal event) is an observation that is different from the majority observations (normal events), accordingly these majority observations are defined as inliers.

In many such situations, building an artificial model of the system behaviour is either too complex or too costly. In such cases, detection system can be designed based on these observations. This kind of system to detect the potential outliers is named as **outlier detection system**, which is also referred to as **novelty detection** or

faulty detection system in previous literatures [Khan and Madden, 2014; Pimentel et al., 2014; Tarassenko et al., 2009].

Usually, this kind of dataset is imbalanced. For specific applications, the normal data are easy to access while the abnormal data are either inaccessible or rarely sampled. Take the machine monitoring system for an example, observations under normal state of the machine are easy to obtain while the breakdown observations are rare or impossible to get. Traditional two-class classifiers will not work well in such case because of the less representation of the abnormal behaviors. Similar examples like the jet engine monitoring system, nuclear power plant monitoring system, human patients disease detection etc. Instead of describing all kinds of unpredictable abnormal behaviors, decision rules can be established only using these normal class data, which leads to the one-class classification problem [Moya and Hush, 1996].

Definition 1.2 (**One-class classification**)**.** One-class classification tries to learn a classifier only based on the normal class (target class) samples to identify new samples belonging to this class or as outliers.

When we conduct an one-class classification, four kinds of situations (see table 1.1) may happen:

(1) A true inlier sample is **correctly accepted** as inlier by the algorithm.

(2) A true inlier sample is wrongly classified as outlier, which is named as **false alarm** (**type I error**) because the alarm is not a true one.

(3) A true outlier is wrongly accepted as inlier, which is named as **miss alarm** (**type II error**) because we should be alerted in that situation.

(4) A true outlier is **correctly rejected** by the algorithm, which is named as **hit** or **true alarm**.

The fraction of the total number of inliers classified as outliers is defined as **false alarm rate**, and the fraction of the total number of outliers classified as inliers is defined as **miss alarm rate**. During the classification, what we should do is to control the false alarm rate (usually it is a user given level) and minimize the miss alarm rate [Tong, Feng, and Zhao, 2016].

TABLE 1.1: Confusion matrix for one-class classification

	true classes	
	inlier (normal event)	outlier (abnormal event)
classified as inlier (normal event)	correct accept	miss alarm (type II error)
classified as outlier (abnormal event)	false alarm (type I error)	correct reject (hit, true alarm)

There are a mount of approaches for one-class classification. According to [Pimentel et al., 2014], these approaches can be categorized as probabilistic based approach (such as mixture models, kernel density estimators), distance based approach (such as nearest neighbour, clustering based approach), reconstruction based approach (such as neural network based approach), domain based approach (such as support vector description and one class support vector machine) and information theoretic approach. Among all of them, in this thesis we constraint the research to one-class support vector machines approach, which does not need distribution assumption in advance, or tons of parameters to tune as with neural network, and it works relatively well with small number of samples.

1.1.2 Transfer learning

One implicit assumption of most machine learning models is that the training data and the testing data are subject to the same distribution. However, this is not always true in the real applications. Imaging that a nuclear power plant detection system which is trained based on data from plant A, which maybe will not work well on a new built plant B because new technical updates may be introduced and the testing data may be subject to different distribution. In order to get a trustful model, when the distribution changes, most of the machine learning models need to be retrained from scratch by using the new collected data [Cao, Zhang, and Yang, 2011]. Doing this is costly and consumes time to collect enough new data to train the new model.

Inspired by the human learning activities, transfer learning appeared to solve the above problems. The intuition is that related learning skills will be helpful for people to pick up new skills. A lot of examples support this idea, such as people with skill

of riding a bicycle will learn the motorcycle more quickly than people without such skill; people with experience of playing guitar may learn the piano in a more efficient manner than people without any music instrument experience; people with Matlab programming experience can learn Python more quickly than people without any programming language skills. The common part of these examples is that people can use information learned before (here the information or skill is: the ability to keep balance for riding a two wheels bicycle, the master of music knowledge for playing instruments, the understanding of computer programming languages) to help the new related learning tasks.

The same thing can happen to the machine learning model, let us define transfer learning as follows:

Definition 1.3 (**Transfer learning**). Transfer learning uses information learned from source task T_S, with source data $\boldsymbol{X}_S = \{\boldsymbol{x}_{S1}, ..., \boldsymbol{x}_{Sn}\}$ which come from source feature space $\mathcal{X}_S$, to help the learning performance of target task T_T with target data $\boldsymbol{X}_T = \{\boldsymbol{x}_{T1}, ..., \boldsymbol{x}_{Tn}\}$ from target feature space $\mathcal{X}_T$.

As we know, deep learning is very "data-hungry", usually it needs millions of samples to train a fine model. When data is insufficient or data collecting is expensive and slow, it will be hard to train a shallow model let alone the deep model. Instead transfer learning can be used to leverage the information learned from related source tasks to improve the target task performance, which is very useful when one learning model is at the small scale data period. An illustration of the performance vs. data scale for different methods (transfer learning, deep learning and classical machine learning) is shown in figure 1.1.

1.1.3 Research problem: transfer learning for one-class classification

In the case of one-class classification, we consider the following transfer learning problems:

(1) Transfer learning for one-class classification in the homogenous spaces. Where the source data and target data are from the same feature space $\mathcal{X}_S = \mathcal{X}_T \subseteq \mathbb{R}^d$,

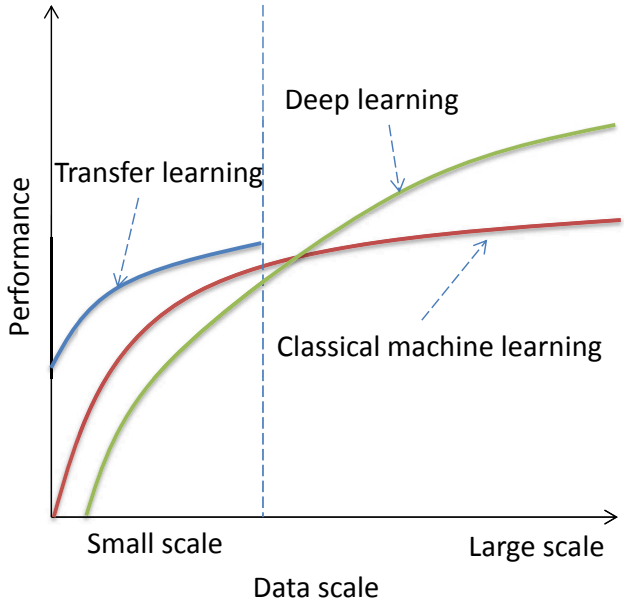


FIGURE 1.1: An illustration of the performance for transfer learning, deep learning and classical machine learning at different data scale. Figure reproduced from source [Ng, 2017; Tommasi, Orabona, and Caputo, 2014]

.

but subject to different distributions $P(\boldsymbol{X}_S) \neq P(\boldsymbol{X}_T)$. Two kinds of examples are listed:

a) Situation happens for the detection system that we need to introduce a new detection system but we do not want to wait long time until we get enough normal data to train a new model, then related trained model can be used to transfer knowledge to the new one.

b) For specific detection system, due to practical reasons we may want to change or update some of the sensors to get an updated system, then new data samples may have different distribution. Again we want to adapt the detection system smoothly from the old detection system to the new updated one without significant performance loss.

(2) Transfer learning for one-class classification in the heterogeneous spaces. Example of this situation is when we want to add some new sensors to the detection system according to the domain expert suggestions, then the source data and target are from different feature spaces $\mathcal{X}_S \subseteq \mathbb{X}^d, \mathcal{X}_T \subseteq \mathbb{R}^q$, where $d \neq q$.

In these situations, we only care about the performances of the new system or the

updated system, and the former system is there to compensate for the lack of data coming for the new system. We will exploit these two categories of transfer learning problems separately in this thesis.

1.2 Thesis contributions

In this thesis we study the transfer learning problems for one-class classification. We solve the above possible problems for detection system from the homogenous space side and heterogeneous space side. The main contributions of this thesis are:

(1) A multi-task learning model for one-class SVM:

This model uses one parameter $\mu \in [0, 1]$ to balance information for one big task training and multiple independent tasks training. It can be solved by classical one-class SVM with a different kernel matrix. When apply to transfer learning case, this model can leverage information from the source detection task to the target detection task by tuning the parameter μ properly. It will give a smooth transition from the source detection system to the new one without significant performance loss.

(2) A kernel path solution method:

As the kernel matrix for the dual optimization problem of the proposed model is parameterized by μ, by using this method the whole path solution can be obtained, which means we can search all the solution space to tune a good parameter μ.

(3) Parameter tuning criteria for multi-task learning:

As the transfer learning is focused on the performance of target task, when we apply this multi-task learning model to the transfer learning case, the parameter tuning criteria guarantee the proper amount of information needed from the source task as the number of samples for target task increases.

(4) Apply the multi-task learning model to the case when new features are added:

Due to practical reasons, when new features are added to the old detection system, by using a variation of the Gaussian kernel parameter, a smooth transition can be obtained from the old detection system to the new one as the number of samples increases.

(5) A constant false alarm rate algorithm for one-class SVM:

As the proposed model can be solved by classical one-class SVM, the online learning problems of one-class SVM are studied and a constant false alarm rate algorithm is developed.

1.3 Publications

The scientific work of this thesis resulted in the following publications:

Journal articles:

- [Xue and Beauseroy, 2017]: Xue, Yongjian, and Pierre Beauseroy. "Transfer learning for one class SVM adaptation to limited data distribution change." Pattern Recognition Letters 100 (2017): 117-123.

Conference articles:

- [Xue and Beauseroy, 2016]: Xue, Yongjian, and Pierre Beauseroy. "Multi-task learning for one-class SVM with additional new features." In Pattern Recognition (ICPR), 2016 23rd International Conference on, pp. 1571-1576. IEEE, 2016.
- [Xue and Beauseroy, 2018b]: Xue, Yongjian and Pierre Beauseroy. "Transfer Learning to Adapt One Class SVM Detection to Additional Features". In:ICPRAM, pp. 78-85. 2018.
- [Xue and Beauseroy, 2018a]: Xue, Yongjian and Pierre Beauseroy (2018a). "Constant false alarm rate for online one-class SVM learning". In: Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on. IEEE, accepted.

1.4 Outline of the thesis

The main structure of this thesis is as follows. Chapter 2 will give a review on related work about one-class SVM and transfer learning. Chapter 3 will propose the multi-task learning model to solve the transfer learning problem with the same feature spaces, a kernel path solution and parameter tuning criteria are also proposed accordingly. In chapter 4, we will solve the transfer learning problem for one-class SVM with new added features. Following that, in chapter 5 we will study the online one-class SVM learning problem and propose a constant false alarm rate online learning approach. At last, we will conclude and discuss possible future work in chapter 6.

2 Related Work

Contents

2.1 Introduction

In this chapter, we will review related work on typical one-class SVM algorithms as well as related literature on multi-task learning and transfer learning for one-class SVM. More precisely, the chapter is organized as follows: section 2.2 gives a general ideal of kernel method used for classification; section 2.3 analyzes two typical one-class SVM algorithms (parameter ν-based and parameter C-based) and their relationships; section 2.4 provides the different highlights of multi-task learning and transfer learning in literature, and we will give an analytical review of the multi-task learning approaches which are used in one-class SVM situation. Finally, we conclude this chapter in section 2.5.

2.2 Kernel methods

Kernel methods are well known for the usage on support vector machines [Vapnik, 1995] for two-class classification problem, the idea is to construct a linear separating hyperplane in a mapped higher dimensional feature space rather than in the original space where the observations are nonlinear separable (figure 2.1). Consider training dataset

$$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}, \tag{2.1}$$

where n is the number of samples, and $\boldsymbol{x}_j \in \mathcal{X}$, $\mathcal{X}$ is the input feature space. Define feature map function ϕ from the input space $\mathcal{X}$ to a higher dimensional feature space $\mathcal{H}$ as:

$$\begin{aligned} \phi \quad : \quad & \mathcal{X} \to \mathcal{H} \\ & \boldsymbol{x} \to \phi(\boldsymbol{x}), \end{aligned} \tag{2.2}$$

To avoid finding the explicit mapping ϕ, kernel trick can be used when linear classifier can be expressed in terms of inner products. More precisely, define kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}. \tag{2.3}$$

Then all inner products terms for input space can be replaced by (2.3) in feature space. As a result, problem changes to define a kernel function which maps the inner product in original space $\mathcal{X}$ to the inner product in higher dimensional feature space $\mathcal{H}$. In order to construct a valid kernel, the Mercer's theorem should be satisfied:

(1) the symmetric $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_j, \boldsymbol{x}_i)$,

(2) the positive definite, for any $c \in \mathbb{R}$ if and only if

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0. \tag{2.4}$$

Many kernels have been proposed for different situations in literature [Scholkopf

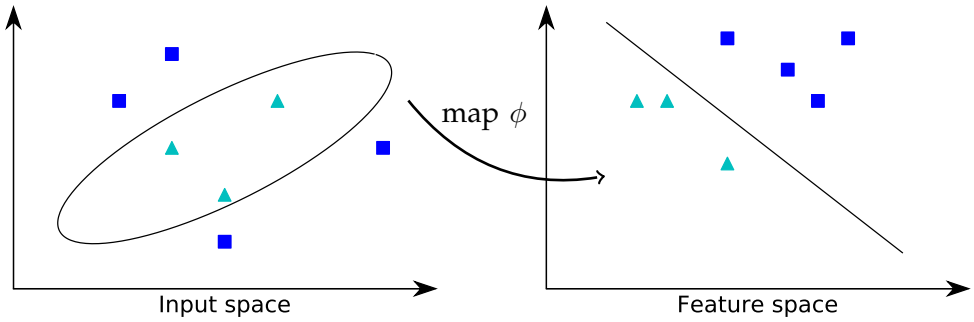


FIGURE 2.1: The idea of mapping the nonlinear separable training data into a feature space where a linear separable hyperplane can be constructed.

and Smola, 2001]. Among them, the polynomial kernels, the Gaussian radial basis kernels and the sigmoid kernels are the most common and important kernels. According to the research of [Schölkopf et al., 2001; Tax, 2001; Tax and Duin, 2004], the Gaussian kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\| \boldsymbol{x}_i - \boldsymbol{x}_j \|^2}{2\sigma^2}\right), \tag{2.5}$$

where $\sigma > 0$, is the most favorable and suitable kernel in the domain of one-class classification, so we use it in all situations of this thesis.

2.3 One-class SVM

There exists two typical kinds of one-class support vector machines (OCSVM) for classification or outliers detection. One is proposed by [Schölkopf et al., 1999, 2001], known as ν one-class support vector machines (ν-OCSVM). It finds an optimal hyperplane in feature space to separate the majority proportion of the data samples from the origin, and the proportion of outliers is controlled by a parameter ν which gives an upper bound on the fraction of outliers and lower bound on the fractions of support vectors for training data.

The other one is introduced by [Tax and Duin, 1999a,b, 2004], named as support vector domain description (SVDD), which aims at finding a hypersphere with minimal volume to enclose the data samples in feature space, the amount of data within the hypersphere is tuned by a parameter C. It is proved that these two approaches lead to the same solution according to [Chang and Lin, 2001; Schölkopf et al., 2001;

Tax, 2001], under build condition that satisfies the Gaussian kernel. Relation can be found in such case between parameters ν and C.

2.3.1 Parameter ν-based one-class SVM

The ν one-class support vector machines (ν-OCSVM) is a kind of new support vector machines proposed by [Schölkopf et al., 1999, 2001] which can be used for novelty detection and quantile estimation. The main idea of ν-OCSVM is to find a hyperplane with maximum margin to separate most of samples from the origin in the feature space (figure 2.2). The proportion of training samples classified as outliers by this hyperplane is controlled by a parameter ν, which gives an upper bound on the fraction of outliers and lower bound on the fraction of support vectors.

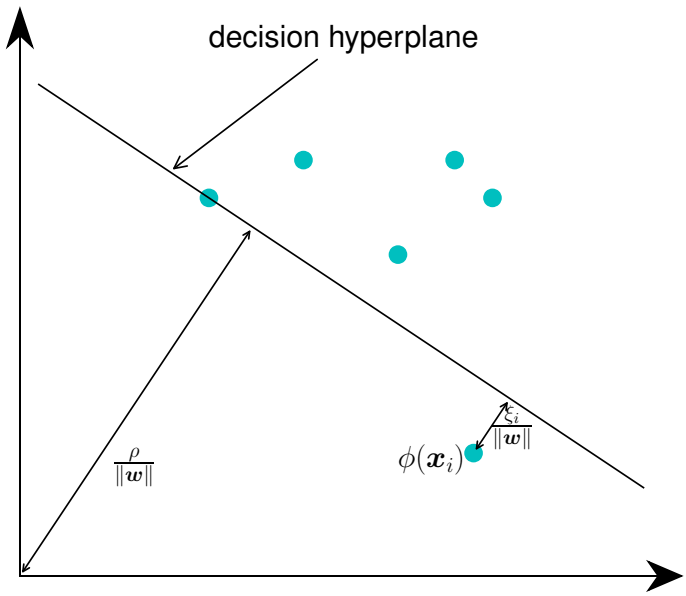


FIGURE 2.2: An illustration of ν-OCSVM, the hyperplane $\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle = \rho$ separates most of the data from the origin except $\phi(\boldsymbol{x}_i)$, where the distance from that point to the hyperplane is $\frac{\xi_i}{\|\boldsymbol{w}\|}$ and the distance from the hyperplane to the origin is $\frac{\rho}{\|\boldsymbol{w}\|}$.

Thus, the main problem of ν-OCSVM is formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\xi},\rho} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n\nu}\sum_{i=1}^{n}\xi_i - \rho \\ &s.t.\ \ \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle \geq \rho - \xi_i,\ \xi_i \geq 0, i = 1, 2, ..., n, \end{aligned} \tag{2.6}$$

where ξ_i is slack variable in order to get soft margins and ρ is an offset which can be recovered later.

Introducing multipliers α_i, $\beta_i \geq 0$, the Lagrangian optimization problem can be written as:

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\xi}, \rho) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n\nu}\sum_{i=1}^{n}\xi_i - \rho \\ &- \sum_{i=1}^{n}\alpha_i\left(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle - \rho + \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i. \end{aligned} \tag{2.7}$$

Setting the partial derivatives respect to $\boldsymbol{w}$, $\boldsymbol{\xi}$ and ρ equal to 0, which gives:

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i), \tag{2.8}$$

$$0 \leq \alpha_i = \frac{1}{n\nu} - \beta_i \leq \frac{1}{n\nu}, \tag{2.9}$$

$$\sum_{i=1}^{n}\alpha_i = 1. \tag{2.10}$$

Substituting equations (2.8), (2.9) and (2.10) into (2.7), then the dual problem is:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \sum_{i,j=1}^{n}\alpha_i\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &s.t.\ 0 \leq \alpha_i \leq \frac{1}{n\nu},\ \sum_{i=1}^{n}\alpha_i = 1, i = 1, 2, ..., n. \end{aligned} \tag{2.11}$$

Again it can be solved by quadratic programming algorithms. According to the KKT conditions, the offset ρ can be recovered by any support vector $\phi(\boldsymbol{x}_k)$ on the hyperplane (the corresponding α_i satisfies $0 < \alpha_k < \frac{1}{n\nu}$):

$$\rho = \langle \boldsymbol{w}, \phi(\boldsymbol{x}_k)\rangle = \sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_k). \tag{2.12}$$

For new data point $\boldsymbol{x}$, the decision function is defined as:

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{\alpha}, \rho) &= sign\left(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle - \rho\right) \\ &= sign\left(\sum_{i=1}^{n}\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - \rho\right) \end{aligned} \tag{2.13}$$

It returns +1 for points classified as normal and -1 for those classified as outliers. Assuming k is the number of training points classified by (2.13) as outliers (the corresponding α_i satisfies $\alpha_i = \frac{1}{n\nu}$), m is the number of support vectors (the corresponding α_i satisfies $0 < \alpha_i \leq \frac{1}{n\nu}$), according to (2.9) and (2.10):

$$\sum_{i=1}^{k} \alpha_i = k\frac{1}{n\nu} \leq 1 \Rightarrow \frac{k}{n} \leq \nu \tag{2.14}$$

$$\sum_{i=1}^{m} \alpha_i = 1 \leq m\frac{1}{n\nu} \Rightarrow \nu \leq \frac{m}{n} \tag{2.15}$$

From (2.14) and (2.15), it is clear that ν gives an upper bound on the fraction of outliers and lower bound on the fraction of support vectors for training data.

2.3.2 Parameter C-based one-class SVM

For C-based one-class SVM, the well known method is SVDD proposed by [Tax and Duin, 1999a]. For the convenient derivations of the proposed multi-task learning approach in chapter 3, another one-class SVM (named as C-OCSVM) derived from the formulation of two-class SVM will also be introduced. Both of them are regularized by parameter C, so we name them as parameter C-based one-class SVM.

2.3.2.1 SVDD

Support vector domain description (SVDD) was introduced by [Tax and Duin, 1999a], which is inspired by the support vector machines [Vapnik, 1995] for two-class classification problem. The idea of SVDD is to enclose most of the training data within a minimum volume hypersphere (figure 2.3). Samples enclosed in this spherically shaped decision boundary are accepted as normal observations (inliers) while outside of this boundary are rejected as abnormal events (outliers).

In order to get more flexible descriptions, similar to SVM [Vapnik, 1995] , kernel tricks can be used to implicitly transform (map ϕ) the data to new feature spaces $\mathcal{H}$ without much extra computation cost. Besides that, slack variables $\boldsymbol{\xi}$, $\xi_i \geq 0$ with penalization factor C are introduced to get soft margins to balance the structural error

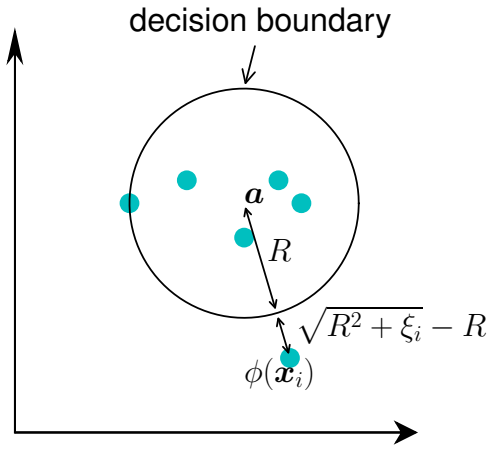


FIGURE 2.3: An illustration of SVDD. Most of the data points are enclosed by a hypersphere (with center $\boldsymbol{a}$ and radius R) as inliers except that $\boldsymbol{x}_i$ is outlier with the corresponding slack variable $\xi_i > 0$. According to the constraint of (2.16), the nearest distance from that point to the boundary is $\sqrt{R^2+\xi_i} - R$.

and the empirical error. Let $\boldsymbol{a}$ be the center and R be the radius of the hypersphere, then the problem of SVDD can be formulated as:

$$\begin{aligned} &\min_{R,\boldsymbol{\xi},\boldsymbol{a}} R^2 + C\sum_{i=1}^{n}\xi_i \\ &s.t.\ \ \|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|^2 \leq R^2 + \xi_i,\ \xi_i \geq 0, i = 1, 2, ..., n. \end{aligned} \tag{2.16}$$

Introducing Lagrange multipliers $\alpha_i, \beta_i \geq 0$, then:

$$\begin{aligned} L(R, \boldsymbol{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= R^2 + C\sum_{i=1}^{n}\xi_i \\ &- \sum_{i=1}^{n}\alpha_i\left(R^2 + \xi_i - \|\phi(\boldsymbol{x}_i) - \boldsymbol{a}\|^2\right) - \sum_{i=1}^{n}\beta_i\xi_i. \end{aligned} \tag{2.17}$$

Setting the partial derivatives respect to R, $\boldsymbol{a}$ and $\boldsymbol{\xi}$ to 0, which leads to:

$$\sum_{i=1}^{n}\alpha_i = 1, \tag{2.18}$$

$$\boldsymbol{a} = \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i), \tag{2.19}$$

$$0 \leq \alpha_i = C - \beta_i \leq C. \tag{2.20}$$

The radius R can be computed by the distance from the points $\phi(\boldsymbol{x}_k)$ on the decision boundary to the center $\boldsymbol{a}$, notice that due to KKT conditions, the corresponding Lagrangian multipliers must satisfy $0 < \alpha_k < C$. Then:

$$\begin{aligned} R^2 &= \|\phi(\boldsymbol{x}_k) - \boldsymbol{a}\|^2 \\ &= k(\boldsymbol{x}_k, \boldsymbol{x}_k) - 2\sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_k, \boldsymbol{x}_i) + \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{2.21}$$

Substituting equations (2.18), (2.19) and (2.20) into (2.17), then the dual problem is:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_i) \\ &s.t.\ 0 \le \alpha_i \le C,\ \sum_{i=1}^{n} \alpha_i = 1, i = 1, 2, ..., n. \end{aligned} \tag{2.22}$$

The above problem is a well known quadratic programming (QP) problem, which can be solved by standard QP solver. Then the decision function of SVDD for data point $\boldsymbol{x}$ is defined as:

$$f(\boldsymbol{x}, \boldsymbol{\alpha}, R) = sign\left(R^2 - k(\boldsymbol{x}, \boldsymbol{x}) + 2\sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) - \sum_{i,j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right). \tag{2.23}$$

It returns +1 for points enclosed by the boundary and -1 for outliers.

2.3.2.2 C-OCSVM

Derived from two-class SVM, another formulation of one-class SVM [Lee and Scott, 2007] constrained by parameter C (C-OCSVM) can be written as:

$$\begin{aligned} &\min_{\boldsymbol{w}, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i \\ &s.t.\ \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle \ge 1 - \xi_i,\ \xi_i \ge 0, i = 1, 2, ..., n. \end{aligned} \tag{2.24}$$

Introducing Lagrangian multipliers $\alpha_i, \beta_i \geq 0$, then the optimization function is formulated as:

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &- \sum_{i=1}^{n}\alpha_i\left(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle - 1 + \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i. \end{aligned} \tag{2.25}$$

Setting the partial derivatives respect to $\boldsymbol{w}$, $\boldsymbol{\xi}$ to 0, relations can be obtained as follows:

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i), \tag{2.26}$$

$$0 \leq \alpha_i = C - \beta_i \leq C. \tag{2.27}$$

In order to get the dual optimization problem, substituting (2.26) and (2.27) into (2.25), then:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \sum_{i,j=1}^{n}\alpha_i\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i \\ &s.t.\ 0 \leq \alpha_i \leq C, i = 1, 2, ..., n, \end{aligned} \tag{2.28}$$

which is also quadratic programming problem. Accordingly, the decision function for new data $\boldsymbol{x}$ is defined as:

$$f(\boldsymbol{x}, \boldsymbol{\alpha}) = sign\left(\sum_{i=1}^{n}\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) - 1\right). \tag{2.29}$$

It returns +1 if samples are classified as normal points and -1 for outliers.

2.3.3 Relation of ν-based and C-based one-class SVM

In this section, we will show the equivalent relationships of these two parameters ν and C-based methods.

2.3.3.1 ν-OCSVM vs. SVDD

For kernels satisfying $k(\boldsymbol{x}, \boldsymbol{x}) = const$, if we set

$$C = \frac{1}{n\nu} \tag{2.30}$$

then the problem for SVDD (2.22) and ν-OCSVM (2.11) will be equivalent. Especially, if Gaussian kernel (2.5) is used, then for any point $\boldsymbol{x}$ we have $k(\boldsymbol{x}, \boldsymbol{x}) = 1$, which means that we have unit norm in the feature space and all the mapped points lie on a hypersphere with unit radius. A geometric view is shown in figure 2.4.

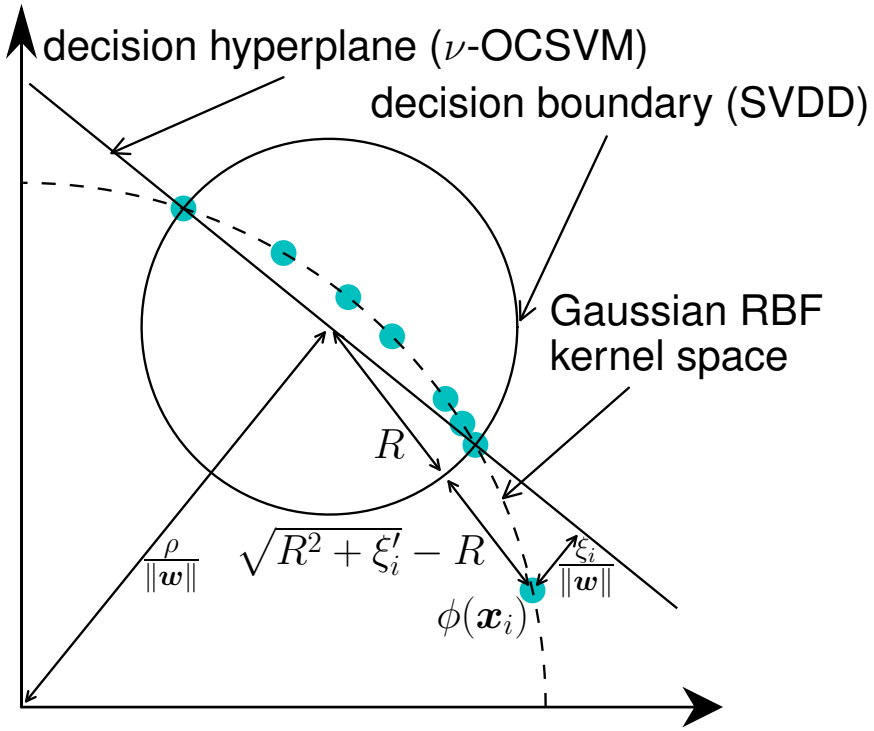


FIGURE 2.4: In the Gaussian RBF kernel space, the idea of SVDD (with slack variable ξ_i') for finding the smallest sphere is equivalent to that of ν-OCSVM (with slack variable ξ_i) for finding a hyperplane with maximum margin to the origin, where $\xi_i' = 2\xi_i$.

2.3.3.2 ν-OCSVM vs. C-OCSVM

Note $\boldsymbol{w}^\nu$, $\boldsymbol{\xi}^\nu$, $\boldsymbol{\alpha}^\nu$, ρ as the solution of ν-OCSVM, and $\boldsymbol{w}^C$, $\boldsymbol{\xi}^C$, $\boldsymbol{\alpha}^C$ as the solution of C-OCSVM. Let

$$\boldsymbol{w}^\nu = \rho \boldsymbol{w}^C \tag{2.31}$$

$$\xi_i^\nu = \rho \xi_i^C \tag{2.32}$$

then (2.6) could be rewritten as:

$$\begin{aligned} &\min_{\boldsymbol{w}^C, \boldsymbol{\xi}^C, \rho} \rho^2 \left(\frac{1}{2}\|\boldsymbol{w}^C\|^2 + \frac{1}{n\nu\rho}(\sum_{i=1}^{n} \xi_i^C - 1) \right) \\ &s.t.\ \langle \boldsymbol{w}^C, \phi(\boldsymbol{x}_i) \rangle \geq 1 - \xi_i^C,\ \xi_i^C \geq 0, \rho \geq 0, i = 1, 2, ..., n \end{aligned} \tag{2.33}$$

Notice that we have fixed solution of ρ for given ν, if we set

$$C = \frac{1}{n\nu\rho} \tag{2.34}$$

then the ν-OCSVM (2.6) and C-OCSVM (2.24) will lead to the same solution. Then from (2.8), (2.26) and (2.31), we can conclude that:

$$\alpha_i^\nu = \rho\alpha_i^C \tag{2.35}$$

Since $\sum_i \alpha_i^\nu = 1$, then:

$$\rho = \frac{1}{\sum_i \alpha_i^C} \tag{2.36}$$

So the equivalent relationships of ν-OCSVM and C-OCSVM can be expressed by (2.35) and (2.36) if condition (2.34) is satisfied.

2.4 Multi-task learning vs. transfer learning

For some specific applications, such as medical image analysis, industrial detection system, data collecting and data labelling are costly as it is time consuming and domain specific experts are required. As a result, a trustful model is hard to build by using traditional machine learning algorithms when the data are insufficient. In this case, multi-task learning (MTL) [Zhang and Yang, 2017] and transfer learning (TL) [Pan and Yang, 2010; Weiss, Khoshgoftaar, and Wang, 2016] are good ways to tackle this problem.

Both of them are inspired by the human learning activities where new skill can be learned more efficiently by the help of related experiences or skills, such as the learning experience of riding the bicycle will be helpful for the learning process of driving a motorcycle and vice versa. According to [Pan and Yang, 2010], multi-task

learning is mainly classified as inductive transfer learning. The difference between them is that multi-task learning (figure 2.5) is focused on improving the multiple related tasks simultaneously while transfer learning (figure 2.6) only cares about the performance of the target task by using the source tasks as auxiliary task.

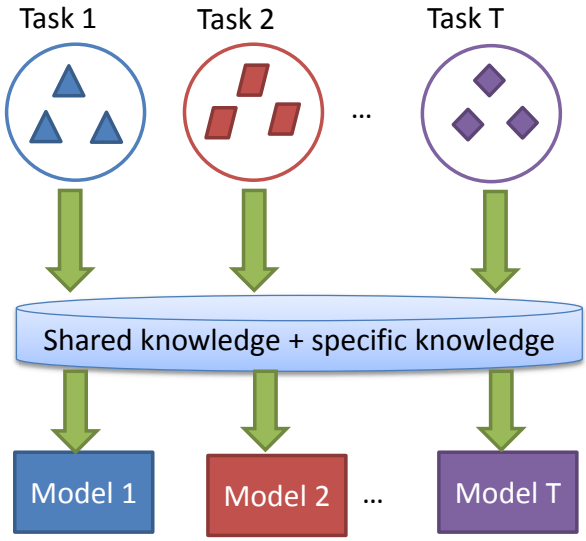


FIGURE 2.5: Multi-task learning.

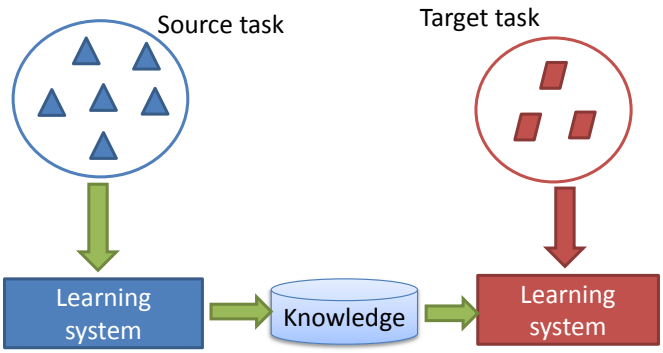


FIGURE 2.6: Transfer learning, source [Pan and Yang, 2010].

2.4.1 Multi-task learning

In order to take advantage of the information or knowledge shared in multiple tasks, according to the specific areas, different definitions of task relatedness produce different multi-task learning methods. For example, [Caruana, 1997] assumes that multiple related tasks share related hidden units in artificial neural nets; For two-class SVM, [Evgeniou and Pontil, 2004] assume that the hyperplane for decision function

is constructed by a mean part and a specific part which follows the intuition of Hierarchical Bayes. However, rare literature covers the area of one-class classification. Following the same idea of [Evgeniou and Pontil, 2004], [He et al., 2011, 2014] proposed a multi-task learning algorithm for one-class SVM. As our proposed method in chapter 3 is derived from that one, here we give a detail review on it.

Consider the case of T learning tasks in the same feature space $\mathcal{X}$, where $\mathcal{X} \in \mathbb{R}^d$. We have n_t samples for each task t, $\boldsymbol{X}_t = \{\boldsymbol{x}_{1t}, \boldsymbol{x}_{2t}, ..., \boldsymbol{x}_{n_t t}\} \in \mathcal{X}$. For standard ν one-class SVM, a hyperplane in the mapped feature space for task t can be defined as:

$$f_t(\boldsymbol{x}) = sign(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}) \rangle - \rho_t), \tag{2.37}$$

where $\boldsymbol{w}_t$ is a vector orthogonal to the hyperplane and ρ_t is an offset. By using the assumption that the parameterized vector $\boldsymbol{w}_t$ can be divided as a mean vector $\boldsymbol{w}_0$ and a specific vector $\boldsymbol{v}_t$ for each task:

$$\boldsymbol{w}_t = \boldsymbol{w}_0 + \boldsymbol{v}_t \tag{2.38}$$

Then the primal problem for the multi-task learning situation can be formulated as:

$$\begin{aligned} \min_{\boldsymbol{w}_0, \boldsymbol{v}_t, \xi_{it}, \rho_t} & \frac{1}{2} \sum_{t=1}^{T} \|\boldsymbol{v}_t\|^2 + \frac{\mu}{2} \|\boldsymbol{w}_0\|^2 + \sum_{t=1}^{T} \sum_{i=1}^{n_t} \frac{\xi_{it}}{n_t \nu_t} - \sum_{t=1}^{T} \rho_t \\ s.t. \ & \langle (\boldsymbol{w}_0 + \boldsymbol{v}_t), \phi(\boldsymbol{x}_{it}) \rangle \geq \rho_t - \xi_{it}, \ \xi_{it} \geq 0, \end{aligned} \tag{2.39}$$

for $t \in \{1, 2, ..., T\}$, $i \in \{1, 2, ..., n_t\}$, ξ_{it} is slack variable corresponding to sample $\boldsymbol{x}_{it}$ and the regularization parameter μ is introduced to control the importance of the task similarity.

Introducing the multipliers $\alpha_{it}, \beta_{it} \geq 0$, then the Lagrangian function is written as:

$$\begin{aligned} L(\boldsymbol{w}_0, \boldsymbol{v}_t, \boldsymbol{\xi}, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}) = & \frac{1}{2} \sum_{t=1}^{T} \|\boldsymbol{v}_t\|^2 + \frac{\mu}{2} \|\boldsymbol{w}_0\|^2 + \sum_{t=1}^{T} \sum_{i=1}^{n_t} \frac{\xi_{it}}{n_t \nu_t} - \sum_{t=1}^{T} \rho_t \\ & - \sum_{t=1}^{T} \sum_{i=1}^{n_t} \alpha_{it} \left(\langle (\boldsymbol{w}_0 + \boldsymbol{v}_t), \phi(\boldsymbol{x}_{it}) \rangle - \rho_t + \xi_{it} \right) - \sum_{t=1}^{T} \sum_{i=1}^{n_t} \beta_{it} \xi_{it} \end{aligned} \tag{2.40}$$

Setting the partial derivatives respect to $\boldsymbol{w}_0$, $\boldsymbol{v}_t$, $\boldsymbol{\xi}$ and ρ equal to zero, then we have the following relations:

$$\boldsymbol{w}_0 = \frac{1}{\mu}\sum_{t=1}^{T}\sum_{i=1}^{n_t}\alpha_{it}\phi(\boldsymbol{x}_{it}), \tag{2.41}$$

$$\boldsymbol{v}_t = \sum_{i=1}^{n_t}\alpha_{it}\phi(\boldsymbol{x}_{it}), \tag{2.42}$$

$$0 \leq \alpha_{it} = \frac{1}{n_t\nu_t} - \beta_{it} \leq \frac{1}{n_t\nu_t}, \tag{2.43}$$

$$\sum_{i=1}^{n_t}\alpha_{it} = 1. \tag{2.44}$$

The dual problem can be obtained by substituting (2.41), (2.42), (2.43) and (2.44) into (2.40):

$$\begin{aligned}\min_{\alpha_{it}} \sum_{t=1}^{T}\sum_{r=1}^{T}\sum_{i=1}^{n_t}\sum_{j=1}^{n_r}\alpha_{it}\alpha_{jr}\left(\frac{1}{\mu}+\delta_{rt}\right)k(\boldsymbol{x}_{it},\boldsymbol{x}_{jr}) \\ s.t.\ 0 \leq \alpha_{it} \leq \frac{1}{n_t\nu_t},\ \sum_{i=1}^{n_t}\alpha_{it} = 1,\end{aligned} \tag{2.45}$$

where $k(\boldsymbol{x}_{it},\boldsymbol{x}_{jr}) = \langle\phi(\boldsymbol{x}_{it}),\phi(\boldsymbol{x}_{jr})\rangle$ and δ_{rt} is defined as:

$$\delta_{rt} = \begin{cases} 1, & \text{if } r = t, \\ 0, & \text{otherwise.}\end{cases} \tag{2.46}$$

Notice that the optimization term of problem (2.45) is very similar to that of (2.11) except with a different valid kernel (with a factor term $\frac{1}{\mu}+\delta_{rt}$). The intention of [He et al., 2014] is to solve problem (2.45) by using the standard one-class SVM optimization approach with the purpose of getting balanced training results between T independent tasks learning (treating the T tasks as separated tasks, noted as T-OCSVM) and a big union task learning (treating all the tasks as one big task, noted as I-OCSVM). More precisely, this is hopefully achieved by setting the value of parameter μ. That is to say when μ is small enough, the multi-task learning will hopefully get identical solutions with the big union task learning and when μ is large enough it will hopefully be identical with the T separated tasks learning.

However, problem (2.45) can not be solved by means of standard one-class SVM

even when we have the same ν and n for every task because the equality constraint $\sum_{i=1}^{n_t} \alpha_{it} = 1$ for MTL is the sum of Lagrangian multipliers over task t. Even though we have solved the problem (2.45), the solution of MTL will never be identical with that of I-OCSVM no matter how small μ is. The proof is that when μ is small enough, the optimization term of (2.45) can be regarded as the same as that of I-OCSVM, but the constraint for I-OCSVM satisfies: $\sum_{t=1}^{T} \sum_{i=1}^{n_t} \alpha_{it} = 1$ while the constraint for MTL satisfies: $\sum_{i=1}^{n_t} \alpha_{it} = 1$ for all t.

2.4.2 Transfer learning

An increasing number of research papers about transfer learning appeared during the last decade. From different point of view, those transfer learning approaches can be categorized differently.

Based on the availability of labels, transfer learning can be categorized as inductive transfer learning, transductive transfer learning and unsupervised transfer learning [Pan and Yang, 2010]. Where inductive transfer learning refers to the case of having labeled data in the target domain, such as the multi-task learning for supervised learning [Evgeniou and Pontil, 2004; Maurer, Pontil, and Romera-Paredes, 2016; Zhang, 2015] and self-taught learning [Raina et al., 2007]. Transductive transfer learning refers to the case of only having labeled data in the source domain, such as domain adaptation [Daume III and Marcu, 2006] and covariance shift [Shimodaira, 2000]. Finally, unsupervised learning refers to the case of no labeled data both in source domain and target domain, such as transfer learning for dimensionality reduction [Pan, Kwok, and Yang, 2008].

Based on the difference of probability distribution, three kinds of transfer learning situations happen. The first one is when the marginal distribution of source data $\boldsymbol{X}_S$ is different from that of target data $\boldsymbol{X}_T$, that is $P(\boldsymbol{X}_S) \neq P(\boldsymbol{X}_T)$. The second one happens when the conditional distribution of source data and target data is different for supervised learning, that means $P(\boldsymbol{Y}_S \mid \boldsymbol{X}_S) \neq P(\boldsymbol{Y}_T \mid \boldsymbol{X}_T)$, where $\boldsymbol{Y}_S$ and $\boldsymbol{Y}_T$ are labels for source dataset and target dataset. The last situation is that both of the two cases happen at the same time. As a result, methods [Daume III and Marcu,

2006; Kulis, Saenko, and Darrell, 2011; Long et al., 2014] based on the correction of these mismatched distribution are mainly proposed as domain adaptation.

Based on the difference between input feature spaces, transfer learning can be classified as homogenous learning and heterogeneous learning. Homogenous transfer learning is the mostly studied case, where input feature space of source data $\mathcal{X}_S$ is the same as that of target data $\mathcal{X}_T$, that is $\mathcal{X}_S = \mathcal{X}_T$. While heterogeneous transfer learning happens when the feature space is different for source data and target data, that means $\mathcal{X}_S \neq \mathcal{X}_T$, which is less studied in literatures compared to homogenous learning. Possible methods [Duan, Xu, and Tsang, 2012; Kulis, Saenko, and Darrell, 2011; Li et al., 2014] are to transform both of the input spaces into a same feature space where the corresponding learning tasks can be performed.

Based on the knowledge to transfer, a few subcategories can be defined, such as instance based transfer learning, parameter based transfer learning, feature based transfer learning and so on [Pan and Yang, 2010; Weiss, Khoshgoftaar, and Wang, 2016]. Instance based transfer learning is motivated by importance sampling and accordingly reweighted source domain instances are used to improve the performance of the target domain task [Gretton et al., 2009; Quionero-Candela et al., 2009]. Parameter based transfer learning is based on the assumption that related tasks share similar parameters, such the hyperplane information for SVM and the similar layers for neural network etc [Caruana, 1997; Evgeniou and Pontil, 2004]. Feature based transfer learning is under the assumption that related tasks have some common feature representations [Blitzer, McDonald, and Pereira, 2006] or common latent input feature space [Shi et al., 2010].

Above all, the unsupervised learning area for transfer learning is not well studied by the literatures, where labeled data are unavailable for both domains. One class classification (outliers detection) is a kind of unsupervised learning in most cases as outliers or abnormal events detection are conducted only on normal data during the training process. The method of multi-task learning for one-class SVM approach (2.39) can be cast into the framework of parameter based learning as the vector to define the hyperplane can be divided as common part and specific part.

The most important difference between transfer learning and multi-task learning is the different learning objective: transfer learning is focused on the results of target task learning while multi-task learning gives all the learning tasks the same importance. If we use multi-task learning for the purpose of transfer learning side, potential negative transfer problem (that is to say the performance by training a model based only on the target dataset is better than that of the transfer learning model) should be addressed properly.

2.5 Related work summary

In this chapter, we first introduced basic kernel mapping idea for classification and followed that we reviewed two typical kinds of one-class SVM methods, parameter ν-based and C-based algorithms. Then we showed their equivalent relationship under the use of Gaussian kernel and parameter connection between C and ν.

If the dataset for training and testing are sampled from different distribution or when we encounter the situation of insufficient data for training a new model, transfer learning or multi-task learning can be used accordingly. We gave general idea on how they work and detailed a multi-task learning method for one-class SVM, but the disadvantage of that method is that we can not solve the problem by classical one-class SVM solver and it offers non-identical solution for multi-task learning and one big total task learning as we want (by tuning parameter μ).

Furthermore, we gave the relationship between multi-task learning and transfer learning, then based on the literature reviews, different methods are categorized. We cast the transfer learning for one class SVM into the unsupervised learning, which is a rarely studied area in transfer learning literatures. As the different purpose for multi-task learning and transfer learning, if we want to use multi-task learning for the transfer learning side, then negative transfer problem should be tackled accordingly.

In the next chapter, we will propose a new multi-task learning model for one-class SVM to solve the problems mentioned above.

3 Transfer Learning for One-class SVM with the Same Features

Contents

In this chapter, we will propose a new multi-task learning model in section 3.2, in order to tackle the transfer learning problem for one-class SVM with a data distribution change in the same feature space. It corresponds to the first research problem that we listed in chapter 1. This proposed multi-task learning approach uses one parameter μ to balance the data training from the source detection system to the target detection system. In section 3.3, we propose a kernel path method which enables to compute all solutions of μ at once. Then criteria to choose the optimal μ are discussed in section 3.4. Experiments on toy data and real data in section 3.5 assess the performance of the proposed model. At last, we give this chapter's summary in section 3.6.

3.1 Introduction

Data based one class classification rules are widely used in system monitoring. Due to practical reasons, we may come across a change of data distribution with respect to training data. For example, the update or change of the detection sensors in an existing detection system, the introduction of a new related detection system and so on. In such situation, we would like to reduce the performance loss and avoid waiting for long time to get enough new data samples to train a trustful model.

While lacking of representative samples for the new dataset, one can try to adapt the former learned detection rule to the new dataset instead of retraining a new rule which implies to gather a significant amount of data. In order to solve the problems mentioned above, multi-task learning seems to be an ideal mean. It uses the idea that related tasks share some useful information, such as common structure, similar model parameters and common representative features etc.

Previous related research shows that learning multiple related tasks simultaneously leads to better performance than learning them independently [Evgeniou and Pontil, 2004; He et al., 2014; Yang, King, and Lyu, 2010]. For example, [Yang, King, and Lyu, 2010] proposed a multi-task learning framework for ν-SVM by upper-bounding the L_2 difference between each pair parameters in order to have similar solutions for related tasks. Later, motivated by [Evgeniou and Pontil, 2004], [He et al., 2014] proposed another multi-task learning method for one-class SVM under the assumption that related tasks' model or model parameters are close to a certain mean function. However, by changing the balance parameter, this model can not provide identical solution between multi-task learning and separate independent learning.

In the next section, we will propose a new multi-task learning model parameterized by $\mu \in [0, 1]$ which enable to move the solution from treating all tasks as one big task ($\mu = 1$) to treating all tasks as separate task learning ($\mu = 0$). As the proposed method can be solved by classical one-class SVM except with a different kernel matrix parameterized by μ, a new version of kernel adaptation method [Le and Beauseroy, 2015] is proposed which enables to compute all the solutions at once for any value of μ. It facilitates the parameter choice process without computing all the candidate

solution independently. Then criteria to select μ are proposed to find a reasonable solution for a given number of samples from the new system.

3.2 Multi-task learning model

In this section, we first introduce the proposed multi-task learning model and then we apply this model to transfer learning problem when the data distribution experiences a change.

Consider the case of T learning tasks in the same feature space $\mathbb{R}^d$. For each task t, we have n_t samples $\boldsymbol{X}_t = \{\boldsymbol{x}_{1t}, \boldsymbol{x}_{2t}, ..., \boldsymbol{x}_{n_t t}\}$, where $\boldsymbol{x}_{jt} \in \mathbb{R}^d$. Intuitively, we may either try to solve the problem by T independent separated tasks or treat them together as one single learning task. Here the idea is trying to balance between the two extreme cases by introducing a parameter μ. The decision function for each task t is:

$$f_t(\boldsymbol{x}) = sign(\langle \boldsymbol{w}_t, \boldsymbol{\phi}(\boldsymbol{x}) \rangle - 1), \tag{3.1}$$

where $\boldsymbol{w}_t$ is the normal vector and $\boldsymbol{\phi}(\boldsymbol{x})$ is the non-linear feature mapping. In the chosen multi-task learning approach, the needed vector of each task $\boldsymbol{w}_t$ could be divided into two part, one part is the common mean vector $\boldsymbol{w}_0$ shared among all the learning tasks and the other part is the specific vector $\boldsymbol{v}_t$ for a specific task.

$$\boldsymbol{w}_t = \mu \boldsymbol{w}_0 + (1 - \mu)\boldsymbol{v}_t, \tag{3.2}$$

where $\mu \in [0, 1]$, when $\mu = 0$, then $\boldsymbol{w}_t = \boldsymbol{v}_t$, which corresponds to T separated tasks learning. While $\mu = 1$, then $\boldsymbol{w}_t = \boldsymbol{w}_0$, which corresponds to one single global task.

3.2.1 Primal problem

Based on this setting, the primal one class problem could be formulated as:

$$\min_{\boldsymbol{w}_0, \boldsymbol{v}_t, \xi_{it}} \frac{1}{2}\mu \parallel \boldsymbol{w}_0 \parallel^2 + \frac{1}{2}(1 - \mu) \sum_{t=1}^{T} \parallel \boldsymbol{v}_t \parallel^2 + C \sum_{t=1}^{T} \sum_{i=1}^{n_t} \xi_{it} \tag{3.3}$$

for $t \in \{1, 2, ..., T\}$ and $i \in \{1, 2, ..., n_t\}$ in each task, subject to the constraints:

$$\langle \mu \boldsymbol{w}_0 + (1-\mu)\boldsymbol{v}_t, \phi(\boldsymbol{x}_{it}) \rangle \geq 1 - \xi_{it},\ \xi_{it} \geq 0, \tag{3.4}$$

where ξ_{it} is slack variable for each sample and C is penalty parameter.

3.2.2 Dual problem

Introducing the Lagrange multipliers $\alpha_{it}, \beta_{it} \geq 0$, then the Lagrangian of this problem could be expressed as:

$$\begin{aligned} L(\boldsymbol{w}_0, \boldsymbol{v}_t, \xi_{it}, \alpha_{it}, \beta_{it}) &= \frac{1}{2}\mu \parallel \boldsymbol{w}_0 \parallel^2 + \frac{1}{2}\sum_{t=1}^{T}(1-\mu) \parallel \boldsymbol{v}_t \parallel^2 + C\sum_{t=1}^{T}\sum_{i=1}^{n_t}\xi_{it} \\ &- \sum_{t=1}^{T}\sum_{i=1}^{n_t}\alpha_{it}\Big(\langle \mu \boldsymbol{w}_0 + (1-\mu)\boldsymbol{v}_t, \phi(\boldsymbol{x}_{it})\rangle - 1 + \xi_{it}\Big) \\ &- \sum_{t=1}^{T}\sum_{i=1}^{n_t}\beta_{it}\xi_{it}. \end{aligned} \tag{3.5}$$

Then setting the partial derivatives of the Lagrangian to zero, which leads to the following relations:

$$\boldsymbol{w}_0 = \sum_{t=1}^{T}\sum_{i=1}^{n_t}\alpha_{it}\phi(\boldsymbol{x}_{it}), \tag{3.6}$$

$$\boldsymbol{v}_t = \sum_{i=1}^{n_t}\alpha_{it}\phi(\boldsymbol{x}_{it}), \tag{3.7}$$

$$\alpha_{it} \in [0, C]. \tag{3.8}$$

Substituting (3.6), (3.7) and (3.8) into (3.5), the Lagrangian dual form could be given as:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^T K^{\mu}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{1} \\ &s.t.\ \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \end{aligned} \tag{3.9}$$

where $\boldsymbol{\alpha}^T = [\alpha_{11}, ..., \alpha_{n_1 1}, ..., \alpha_{1T}, ..., \alpha_{n_T T}]$, and K^{μ} is a block matrix with $T \times T$ blocks corresponding to all task pairs. Let K^{μ}_{rt} denote the block corresponding to task r and t, which is defined as:

$$K^{\mu}_{rt} = (\mu + (1-\mu)\delta_{rt})\langle \phi(\boldsymbol{X}_t), \phi(\boldsymbol{X}_r)\rangle, \tag{3.10}$$

and δ_{rt} is the Kronecker delta:

$$\delta_{rt} = \begin{cases} 1, & \text{if } r = t, \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

Notice that (3.9) could be solved by the classical one class SVM (C-OCSVM) with the specific modified Gram matrix K^{μ}. Now the decision function for task t is:

$$f_t(\boldsymbol{x}) = sign\Big\{ \sum_{r=1}^{T} \sum_{j=1}^{n_r} \alpha_{jr} \Big(\mu + (1-\mu)\delta_{rt}\Big) \langle \phi(\boldsymbol{x}_{jr}), \phi(\boldsymbol{x}) \rangle - 1 \Big\}. \tag{3.12}$$

Compared to the multi-task learning proposed by [He et al., 2014], this approach has two advantages:

- It can be solved by C-OCSVM, while the method of [He et al., 2014] cannot be solved by ν-OCSVM because the constraint there satisfies $\sum_{i=1}^{n_t} \alpha_{it} = 1$ for all t, while for ν-OCSVM, we need $\sum_{t=1}^{T} \sum_{i=1}^{n_t} \alpha_{it} = 1$.
- The meaning of parameter μ is more significant here. When $\mu = 1$, we have solution corresponding to one union big task learning, when $\mu = 0$, we have solution corresponding to T separate independent tasks learning. As for the method of [He et al., 2014], the initial purpose is to get T separate tasks learning when μ is large enough and one union task solution when μ is small enough, however the solution will not be identical with T separated tasks learning because of the constraint.

3.2.3 Application to transfer learning

Remember that the problem we want to solve here is a transfer learning problem: we want to get a smooth transition from the old system T_1 to the new one T_2. When we need to update some sensors in an exiting system or introduce a new related detection system, there may be a data mismatch problem for T_1 and T_2. During a period of time, we just have limited amount of data for the new detection system T_2, instead of waiting long time to get enough data to train a trustful model, we can

make full use of T_1 to get a reasonable transition to the new one. That means we want to use T_1 to avoid a strong change and to alleviate the lack of data in T_2. The idea is to compute a solution for the new system T_2 but using the old T_1 to limit the solution space to be reasonable according to T_1.

Here we consider the source task T_1 with dataset $\boldsymbol{X}_1$, and the target task T_2 related to the changed dataset $\boldsymbol{X}_2$ ($t \in \{1, 2\}$). According to (3.12), the decision function $f_1(\boldsymbol{x})$ corresponding to T_1 is:

$$f_1(\boldsymbol{x}) = sign(g_1(\boldsymbol{x}) - 1), \tag{3.13}$$

where the SVM function $g_1(\boldsymbol{x})$ is defined as:

$$g_1(\boldsymbol{x}) = \boldsymbol{\alpha}^T \begin{bmatrix} k(\boldsymbol{X}_1, \boldsymbol{x}) \\ \mu k(\boldsymbol{X}_2, \boldsymbol{x}) \end{bmatrix}, \tag{3.14}$$

where $k(\cdot)$ is the kernel function: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ (we use Gaussian kernel in this thesis). If the test data $\boldsymbol{x}$ come from T_2, the decision function $f_2(\boldsymbol{x})$ corresponding to T_2 is:

$$f_2(\boldsymbol{x}) = sign(g_2(\boldsymbol{x}) - 1), \tag{3.15}$$

where the SVM function $g_2(\boldsymbol{x})$ is defined as:

$$g_2(\boldsymbol{x}) = \boldsymbol{\alpha}^T \begin{bmatrix} \mu k(\boldsymbol{X}_1, \boldsymbol{x}) \\ k(\boldsymbol{X}_2, \boldsymbol{x}) \end{bmatrix}. \tag{3.16}$$

If we want to use the proposed multi-task learning model to tackle our transfer learning problem, one important issue need to be solved: the choice of parameter μ by given different number of samples n_2 in target task. In the following sections, we will develop a method to find the path solutions along with μ by one time computation and propose a heuristic method to choose the proper value of μ.

3.3 Kernel path solution

To solve the problem (3.9), the parameter μ must be chosen in order to get a kernel matrix. However, we do not know exactly which μ is better to the current situation. Moreover, it is time consuming to search all the solution space by solving the one class SVM model for a significant set of possible values of μ.

One appealing approach is to determine the entire solution path of the Lagrange multipliers along μ from 0 to 1, then we can choose one solution as we want. A kernel adaptive ν one-class SVM method is proposed in [Le and Beauseroy, 2015] to deduce one solution from another when the kernel is changing from one to another which is not too different. Following similar ideas, here a C version of kernel adaptive one class SVM is developed with an improved search method.

The aim of the kernel adaptive method here is to find the path solutions of the Lagrange multipliers $\boldsymbol{\alpha}^{\mu}$ over the parameter μ. Define

$$K^0 = \begin{bmatrix} K_{ss} & \\ & K_{tt} \end{bmatrix}, \tag{3.17}$$

and

$$K^1 = \begin{bmatrix} K_{ss} & K_{st} \\ K_{st}^T & K_{tt} \end{bmatrix}, \tag{3.18}$$

where $K_{ss} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_1) \rangle$, $K_{st} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_2) \rangle$, $K_{tt} = \langle \phi(\boldsymbol{X}_2), \phi(\boldsymbol{X}_2) \rangle$. Thus, we want to find the solution path from K^0 to K^1. Let:

$$K^{\mu} = (1-\mu)K^0 + \mu K^1, \tag{3.19}$$

which will be the same kernel matrix to (3.10). Consider any value of μ, $f(\boldsymbol{x})$ is the decision function and the following KKT conditions must be satisfied:

- $f^{\mu}(\boldsymbol{x_i}) < 0 \Rightarrow \alpha_i^{\mu} = C$,
- $f^{\mu}(\boldsymbol{x_i}) > 0 \Rightarrow \alpha_i^{\mu} = 0$,
- $f^{\mu}(\boldsymbol{x_i}) = 0 \Rightarrow \alpha_i^{\mu} \in [0, C]$.

Then based on the above conditions, 3 groups of Lagrangian $\mathcal{M}$, $\mathcal{E}$ and $\mathcal{C}$ could be defined as:

- $\mathcal{M}^{\mu} = \{i : f^{\mu}(\boldsymbol{x_i}) = 0,\ \alpha_i^{\mu} \in [0, C]\}$ for samples on the margin,
- $\mathcal{E}^{\mu} = \{i : f^{\mu}(\boldsymbol{x_i}) < 0,\ \alpha_i^{\mu} = C\}$ for samples outside the decision region
- $\mathcal{C}^{\mu} = \{i : f^{\mu}(\boldsymbol{x_i}) > 0,\ \alpha_i^{\mu} = 0\}$ for samples inside the decision region

Assuming a solution α^{μ^-}, $\mathcal{M}^{\mu^-}$, $\mathcal{E}^{\mu^-}$ and $\mathcal{C}^{\mu^-}$ is known for a given parameter μ^-, we want to find another solution α^{μ} where μ is close enough to μ^- such that the group sets do not change. That means:

$$\alpha^{\mu}_{\mathcal{E}^{\mu}} = \alpha^{\mu^-}_{\mathcal{E}^{\mu^-}} = C, \tag{3.20}$$

and

$$\alpha^{\mu}_{\mathcal{C}^{\mu}} = \alpha^{\mu^-}_{\mathcal{C}^{\mu^-}} = 0. \tag{3.21}$$

$\alpha^{\mu}_{\mathcal{M}^{\mu^-}}$ is changing with μ such that all $\boldsymbol{x}_i \in \mathcal{M}^{\mu^-}$ satisfy:

$$f^{\mu}(\boldsymbol{x}_i) = f^{\mu^-}(\boldsymbol{x}_i) = 0. \tag{3.22}$$

From equation (3.20), (3.21) and (3.22), we can get:

$$\boldsymbol{\alpha}^{\mu}_{\mathcal{M}^{\mu^-}} = \left(K^{\mu}_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})}\right)^{-1} \left(K^{\mu^-}_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})} \boldsymbol{\alpha}^{\mu^-}_{\mathcal{M}^{\mu^-}} + C(\mu - \mu^-) \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{E}^{\mu^-})} \mathbf{1}\right), \tag{3.23}$$

where $K^{\mu}_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})}$ is the kernel matrix with entries (l, j) of K^{μ} such that the index $l \in \mathcal{M}^{\mu^-}$ and $j \in \mathcal{M}^{\mu^-}$, and

$$\Delta K = K^0 - K^1. \tag{3.24}$$

Following events will affect the composition of the 3 groups $\mathcal{M}, \mathcal{E}, \mathcal{C}$ as μ is increasing from one value μ^- to another one:

1. $\boldsymbol{x}_i$ leaves the border $\mathcal{M}$ to $\mathcal{E}$: $\alpha_i^{\mu^-} \in [0, C] \rightarrow \alpha_i^{\mu} = C$.
2. $\boldsymbol{x}_i$ changes from $\mathcal{M}$ to $\mathcal{C}$: $\alpha_i^{\mu^-} \in [0, C] \rightarrow \alpha_i^{\mu} = 0$.
3. $\boldsymbol{x}_i$ leaves $\mathcal{C}$ to $\mathcal{M}$: $f^{\mu^-}(\boldsymbol{x}_i) > 0 \rightarrow f^{\mu}(\boldsymbol{x}_i) = 0$.

4. $\boldsymbol{x}_i$ leaves the outlier set $\mathcal{E}$ to join the border $\mathcal{M}$: $f^{\mu^-}(\boldsymbol{x}_i) < 0 \rightarrow f^{\mu}(\boldsymbol{x}_i) = 0$.

Considering that we study the value $\boldsymbol{\alpha}^{\mu}$ around a chosen value μ_c ($\mu_c \geq \mu^-$)

$$\mu = \mu_c + \Delta\mu_c. \tag{3.25}$$

What we want to find is $\Delta\mu_c$ such that events 1-4 happen, which means that the partition of $\mathcal{M}, \mathcal{E}, \mathcal{C}$ will change, according to (3.19), (3.24) and (3.25):

$$K^{\mu} = K^{\mu_c} - \Delta\mu_c \Delta K. \tag{3.26}$$

For events 1 and 2, we just need to monitor the value change of α_i^{μ}, which may either reach 0 or C. From (3.23), all the terms are known except the inverse of $K^{\mu}{}_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})}$. As the 2nd order Taylor expansion is:

$$(X+Y)^{-1} = X^{-1} - X^{-1}YX^{-1} + X^{-1}YX^{-1}YX^{-1} + \varepsilon, \tag{3.27}$$

where ϵ is the error term, let $Z = K_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})}$, and $X = Z^{\mu_c}$, $Y = -\Delta\mu_c\Delta K$, then according to (3.27), the inverse of $K^{\mu}{}_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})}$ could be written as:

$$\begin{aligned}\left(Z^{\mu}\right)^{-1} =& \left(Z^{\mu_c}\right)^{-1} + \Delta\mu_c \left(Z^{\mu_c}\right)^{-1} \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})} \left(Z^{\mu_c}\right)^{-1} \\ &+ \Delta\mu_c{}^2 \left(\left(Z^{\mu_c}\right)^{-1} \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})}\right)^2 \left(Z^{\mu_c}\right)^{-1} + \epsilon.\end{aligned} \tag{3.28}$$

Neglecting ϵ, as a result, an approximation of equation (3.23) could be written as:

$$\tilde{\boldsymbol{\alpha}}^{\mu}_{\mathcal{M}^{\mu^-}} = C_0 + \Delta\mu_c C_1 + \Delta\mu_c^2 C_2, \tag{3.29}$$

where:

$$C_0 = \left(Z^{\mu_c}\right)^{-1} Z^{\mu^-} \boldsymbol{\alpha}^{\mu^-}_{\mathcal{M}^{\mu^-}} + C(\mu_c - \mu^-) \left(Z^{\mu_c}\right)^{-1} \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{E}^{\mu^-})} \mathbf{1}, \tag{3.30}$$

$$\begin{aligned}C_1 =& \left(Z^{\mu_c}\right)^{-1} \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})} \left(Z^{\mu_c}\right)^{-1} Z^{\mu^-} \boldsymbol{\alpha}^{\mu^-} \\ &+ C(\mu_c - \mu^-) \left(Z^{\mu_c}\right)^{-1} \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{M}^{\mu^-})} \left(Z^{\mu_c}\right)^{-1} \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{E}^{\mu^-})} \mathbf{1} \\ &+ C \left(Z^{\mu_c}\right)^{-1} \Delta K_{(\mathcal{M}^{\mu^-}, \mathcal{E}^{\mu^-})} \mathbf{1},\end{aligned} \tag{3.31}$$

$$C_2 = C\left(\left(Z^{\mu_c}\right)^{-1}\Delta K_{(\mathcal{M}^{\mu^-},\mathcal{M}^{\mu^-})}\right)^2 + \left(\left(Z^{\mu_c}\right)^{-1}\Delta K_{(\mathcal{M}^{\mu^-},\mathcal{M}^{\mu^-})}\right)^2\left(Z^{\mu_c}\right)^{-1}Z^{\mu^-}\boldsymbol{\alpha}^{\mu^-}_{\mathcal{M}^{\mu^-}}$$
$$+C(\mu_c-\mu^-)\left(\left(Z^{\mu_c}\right)^{-1}\Delta K_{(\mathcal{M}^{\mu^-},\mathcal{M}^{\mu^-})}\right)^2\left(Z^{\mu_c}\right)^{-1}\Delta K_{(\mathcal{M}^{\mu^-},\mathcal{E}^{\mu^-})}\mathbf{1}. \tag{3.32}$$

Then the events of 1 and 2 could be approximately solved by 2nd order polynomial with (3.29).

For samples $\boldsymbol{x}_i \in \mathcal{E}\bigcup\mathcal{C}$, values μ such that $f^{\mu}(\boldsymbol{x}_i) = 0$ need to be detected:

$$K^{\mu}_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{M}^{\mu^-})}\boldsymbol{\alpha}^{\mu}_{\mathcal{M}^{\mu^-}} + CK^{\mu}_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{M}^{\mu^-})}\mathbf{1} - \mathbf{1} = 0. \tag{3.33}$$

Substituting the approximation (3.29) into (3.33), then it comes to:

$$D_0 + \Delta\mu_c D_1 + \Delta\mu_c^2 D_2 = 0, \tag{3.34}$$

where:

$$D_0 = -\mathbf{1} + CK^{\mu_c}_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{E}^{\mu^-})}\mathbf{1} + K^{\mu_c}_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{M}^{\mu^-})}C_0, \tag{3.35}$$

$$D_1 = -C\Delta K_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{E}^{\mu^-})}\mathbf{1} - \Delta K_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{M}^{\mu^-})}C_0 + K^{\mu_c}_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{M}^{\mu^-})}C_1, \tag{3.36}$$

$$D_2 = -\Delta K_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{M}^{\mu^-})}C_1 + K^{\mu_c}_{(\mathcal{E}^{\mu^-}\bigcup\mathcal{C}^{\mu^-},\mathcal{M}^{\mu^-})}C_2. \tag{3.37}$$

Again, the 2nd order polynomial could be solved to find the breakpoints corresponding to events 3 and 4.

The process of the kernel adaptation for C-one class SVM is shown in algorithm 1. We set $\mu^- = \mu_c = 0$ for the initial point, and then α^{μ^-}, f^{μ^-}, $\mathcal{M}$, $\mathcal{E}$, $\mathcal{C}$ and ΔK can be obtained accordingly. For the next breakpoint, first we compute $\mu = \mu^- + \min\Delta\mu_c$ which is a first approximation of the next break point, where $\min\Delta\mu_c$ is the minimum value of $\Delta\mu_c$ such that events 1,2,3,4 happen according to (3.29) and (3.34). Next in order to find a more precise value of the break point which is near to μ_c, we compute the minimum $\Delta\mu_c$ such that events 1,2,3,4 happen based on μ_c and μ^- according to (3.29) and (3.34), then let $\mu = \mu_c + \min\Delta\mu_c$, and $\mu_c = \mu$. We iterate the steps until the change of μ is smaller than a given threshold ϵ which means we find the next breakpoint in a given error. Finally set $\mu^- = \mu$, and update α^{μ^-}, f^{μ^-}, $\mathcal{M}$, $\mathcal{E}$, $\mathcal{C}$ and ΔK accordingly. Repeat the upper steps until $\mu = 1$. One example of

the solution path is shown in figure 3.1, where the Lagrangian multipliers $\boldsymbol{\alpha}$ change as the kernel matrix K^{μ} varies from K^0 to K^1.

Then for a given value of μ, we can find the two nearest breakpoints μ_L and μ_R such that $\mu_L \leq \mu \leq \mu_R$, then α^{μ} could be obtained by (3.23), where $\mu^- = \mu_L$.

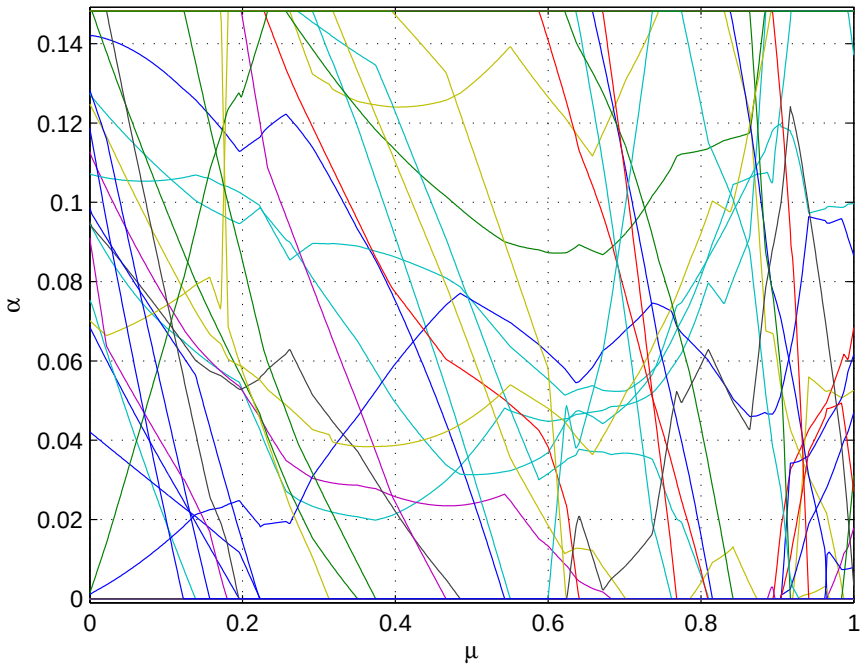


FIGURE 3.1: One example of solution path, where the Lagrangian multipliers $\boldsymbol{\alpha}$ change as the kernel matrix K^{μ} varies from K^0 to K^1.

Algorithm 1 Kernel adaptation for C-one class SVM.

Initialisation: let $\mu^- = \mu_c = 0$, then compute α^{μ^-}, f^{μ^-}, $\mathcal{M}$, $\mathcal{E}$, $\mathcal{C}$ and ΔK.
while $\mu < 1$ **do**
  compute $\min \Delta\mu_c$ such that events 1,2,3,4 happen.
  then $\mu_c = \mu^- + \min \Delta\mu_c$.
  while convergence = False **do**
    compute $\min \Delta\mu_c$ such that events 1,2,3,4 happen.
    then $\mu = \mu_c + \min \Delta\mu_c$.
    if $|\mu - \mu_c| < \epsilon$ **then**
      convergence = True.
    end if
    $\mu_c = \mu$.
  end while
  $\mu^- = \mu$.
  update α^{μ^-}, f^{μ^-}, $\mathcal{M}$, $\mathcal{E}$, $\mathcal{C}$ and ΔK.
end while

3.4 Parameter μ tuning criteria

As we get all the path solutions corresponding to μ, problem arises to choose a good value of μ which makes a good decision function for a given number of samples n_2 to the target task. Besides that, the chosen procedure should be done based on the training dataset $\boldsymbol{X_1}$ and the available data $\boldsymbol{X_2}$ alone. Typical parameter tuning approach like cross validation can not be used in this situation, because for one thing we do not have labeled data for one-class classification in the training phrase, and for the other thing we do not have enough data for the target task at the beginning.

Intuitively, a good solution for the proposed model should have the following properties:

- It could adapt to the change of the decision function due to the data distribution change.
- At the same time, it should keep the false alarm rate as close as possible to the desired one (usually it is a user given level which indicates the proportion of rejected instances).

Based on the above properties, we apply sequential different criteria to reduce the possible domain of μ.

First we can choose a domain for μ corresponding to the maximum number of margin support vectors which come from $\boldsymbol{X}_2$, noted as $\max(\#\text{SV}_2)$. This means that the decision function takes into account the new data when the distribution of training data experiences a change. But for practice results, large number of margin support vectors might have a chance to overestimate the model. As we can measure the number of support vectors from T_2 as a function of μ and restrict the search of μ domain to the corresponding solutions, among all solutions, we choose the Nth largest number of support vectors from T_2, noted as $\max(\#\text{SV}_2)^{Nth}$, which produces a large valid search domain in the first step.

Next, we want to reduce that initial domain so that the false alarm rate is as close as possible to the desired one. Related to empirical estimation of false alarm rate, we want the decision boundary for multi-task model of T_2 to enclose a given proportion

of samples from $\boldsymbol{X}_2$, so we select μ such that:

$$\mu = \arg\min |\#\{f_2(\boldsymbol{X_2}, \mu) > 0\} - n_2(1-p)|. \tag{3.38}$$

For example, we set $p = 0.1$ to the desired false alarm rate, which means we want to keep the proportion of outliers for $\boldsymbol{X_2}$ to be close to 0.1.

Besides that, for multi-task learning point of view, among remaining possible values of μ, we want to select the one which preserves the detection for the initial task T_1 (solution when $\mu = 0$), noted as $\arg\min A(\mu)$, where:

$$A(\mu) = \|g_1(\boldsymbol{X}_1, \mu) - g_1(\boldsymbol{X}_1, \mu = 0)\|. \tag{3.39}$$

Thus, the proposed criteria for choosing the optimal value μ^* are summarized in algorithm 2.

Algorithm 2 Choosing the optimal μ^*.

1: Choose a list L_1 of μ_a,
s.t. $\#\text{SV}_2(\mu_a) \geq \max(\#\text{SV}_2)^{Nth}$, $\mu_a \in [0, 1]$,
where $\max(\#\text{SV}_2)^{Nth}$ is the Nth largest of $\#SV_2$.
2: Choose a list L_2 of μ_b from L_1,
s.t. $\mu_b = \arg\min_{\mu \in L_1} |\#\{f_2(\boldsymbol{X_2}, \mu) > 0\} - n_2(1-p)|$.
3: Choose μ^* from L_1,
s.t. $\mu^* = \arg\min_{\mu \in L_2} A(\mu)$.

3.5 Experiments

Experiments are conducted on toy dataset and wine quality dataset to assess the performance of the proposed model.

3.5.1 Toy dataset

3.5.1.1 Data description

The proposed test dataset is named as banana dataset, it is defined as:

$$x^{(1)} = 2sin\theta + N(0, 0.25); \tag{3.40}$$

$$x^{(2)} = 2cos\theta + N(0, 0.25); \tag{3.41}$$

where θ subjects to uniform distribution $U(\frac{1}{8}\pi, \frac{11}{8}\pi)$ and $N(0, 0.25)$ is Gaussian distribution. The source dataset is $\boldsymbol{X}_1 = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [x^{(1)}, x^{(2)}], \boldsymbol{x} \in \mathbb{R}^2\}$, and the target dataset is $\boldsymbol{X}_2$, which subjects to the same relationship but with a rotation and a translation.

A two dimensional view of 400 source samples and 1,000 target samples with $\pi/12$ rotation and $(-0.3, 0.5)$ translation is shown in figure 3.2. We use source samples $n_1 = 400$, and we add n_2 target samples (from 10 to 1,000). For test, 10,000 target data samples are generated as positive samples and 10,000 negative samples subjecting to uniform distribution which covers the maximum axis of the test dataset are also generated for testing the probability of miss alarm. Results are averaged over 10 repetitions.

3.5.1.2 Experimental settings

Since we have enough training data samples in the source task, the Gaussian kernel parameter σ and the constraint parameter C could be tuned based on the source training samples. According to [Lee and Scott, 2007], if $C = 1/(n\nu\rho)$, the same decision function could be get for C-OCSVM and ν-OCSVM. Besides that, the performance is relatively stable for ν-OCSVM as the number of samples increases if we keep the same value of ν, so we use the ν-OCSVM for the initialisation of the kernel adaptation method each time. Here we choose $\sigma = 1.26$ and $\nu = 0.1$, which makes the false alarm rate around 0.1 in source task.

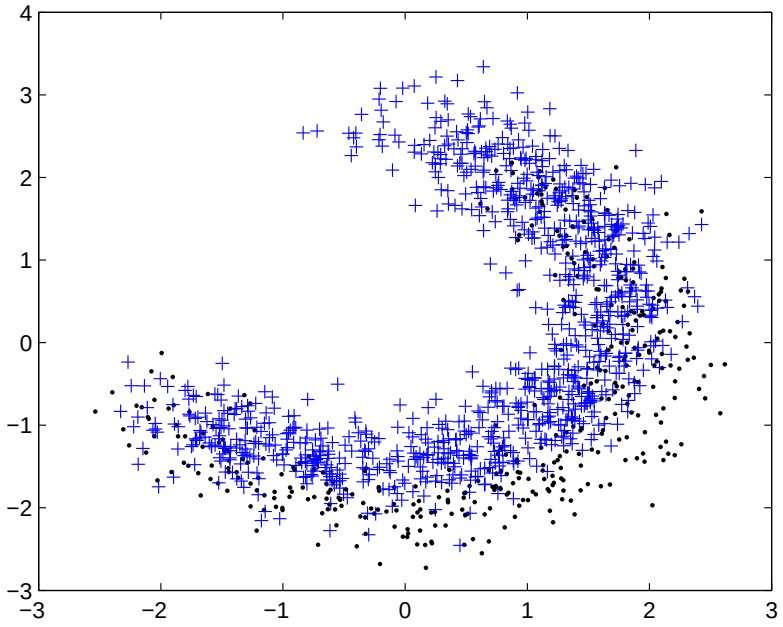


FIGURE 3.2: View of the banana data, the black dot and the blue plus correspond to the source and target samples.

With the kernel adaptation algorithm, all the path solutions could be obtained while μ varies from 0 to 1. For example, figure 3.1 shows the solution path of α with $n_1 = 400$ and $n_2 = 50$.

After that, the search strategy is used to choose a good value for μ. In order to see if the selected μ^* is a proper one, we minimize a reference function $G_{\text{ref}}(\mu)$

$$\min_{\mu \in [0,1]} G_{\text{ref}}(\mu) \tag{3.42}$$

to define a reference value μ_{ref}, where

$$G_{\text{ref}}(\mu) = \|g_2(\boldsymbol{x}, \mu) - g_{2,\text{ref}}(\boldsymbol{x}, \mu = 0)\|, \tag{3.43}$$

and

$$g_{2,\text{ref}}(\boldsymbol{x}, \mu = 0) = \frac{1}{10} \sum_{i=1}^{10} \boldsymbol{\alpha}_i^T \begin{bmatrix} \mathbf{0} \\ k(\boldsymbol{X}_2, \boldsymbol{x}) \end{bmatrix} \tag{3.44}$$

is a reference border defined by averaging 10 realisations of the SVM function for training T_2 alone ($\mu = 0$), with 400 samples for each realisation. By doing this, from a statistic point of view, we can get an averaged boundary which the new system will end up with. Thus, the one μ_{ref} from equation (3.42) will give us a reference

value to check if the one μ^* selected by the criteria is a proper one.

The results of the multi-task learning method noted as MTL(T_2, μ^*) are compared with:

- the independent training of $\boldsymbol{X}_1$ (noted as MTL$(T_1, \mu = 0)$, the same solution as T_1),
- the method of combining $\boldsymbol{X}_1$, $\boldsymbol{X}_2$ together as a single big task (T_{Big}),
- the method of training target data $\boldsymbol{X}_2$ alone (T_2),
- multi-task learning with reference value MTL(T_2, μ_{ref}).

3.5.1.3 Parameter μ tuning

Figure 3.3 shows the results of one realisation for the proposed three steps to tune parameter μ as well as the reference function for different values of μ as n_2 increases from 10 to 400.

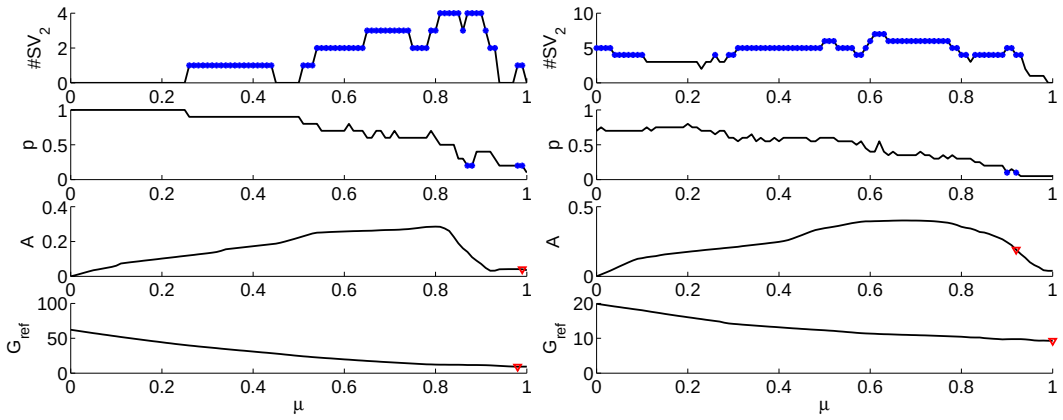
- $\#SV2$ is the number of margin vectors (candidate value μ_a corresponding to the blue star points, here we choose the 4th largest $\#SV2$).
- p is the proportion of outliers from dataset $\boldsymbol{X_2}$ (candidate value μ_b corresponding to the blue star points).
- A is the distance between SVM function $g_1(\boldsymbol{X_1}, \mu)$ and $g_1(\boldsymbol{X_1}, \mu = 0)$, which is define in equation (3.39) (the one corresponding to the red triangle is the final selected μ^*).
- G_{ref} is the value of the reference function which is defined in equation (3.42) (the one corresponding to the red triangle is the reference value μ_{ref}).

Results (figure 3.3) show that the final chosen value μ^* has the same trend with the reference value μ_{ref} as n_2 increases. More specifically, both of them decreases from a large value (close to 1) to a very small value (close to 0) as n_2 changes from 10 to 400, which means that the multi-task learning model parameterized by μ tends to move from one big task learning (as the number of samples from target task is small) to

two separate tasks learning (as more representative samples are collected from the target task).

This property facilitates the practical applications. As we know, when the detection system experiences a change, we care about the performance of the new task and not about the performance of the former one. Then, after the system change, for a certain time period we just have a small number of new samples. The proposed method tends to select μ which enables to transfer as much information as needed from the old task to the new one to keep approximately stable performances even if the amount of new data is limited. When enough new samples are collected, the model tends to choose small mu which indicates that the former solution and data are not useful anymore. To some extend, negative transfer learning is also avoided by tuning μ dynamically.

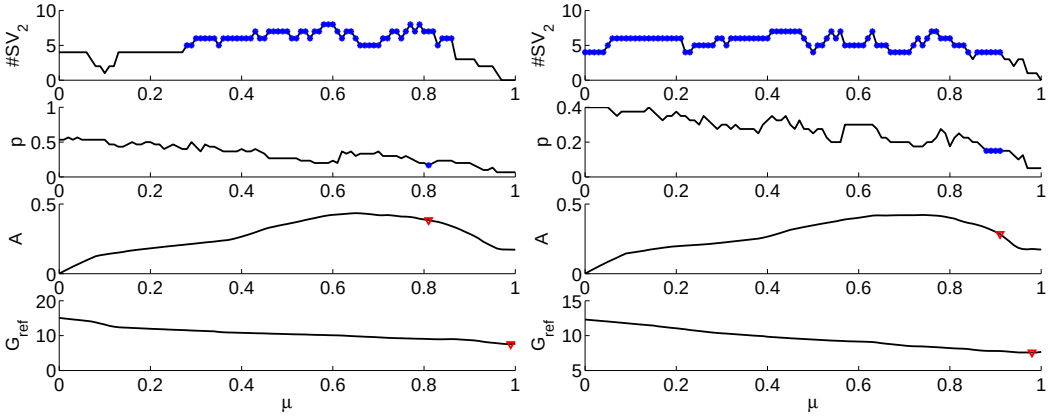
Figure 3.4 shows the averaged comparison results by 10 realisations of μ^* and μ_{ref} as n_2 changes. Obviously, they have the same trend as n_2 increases from 10 to 1000. From the above, the proposed criteria can tune the parameter μ properly as n_2 increases.



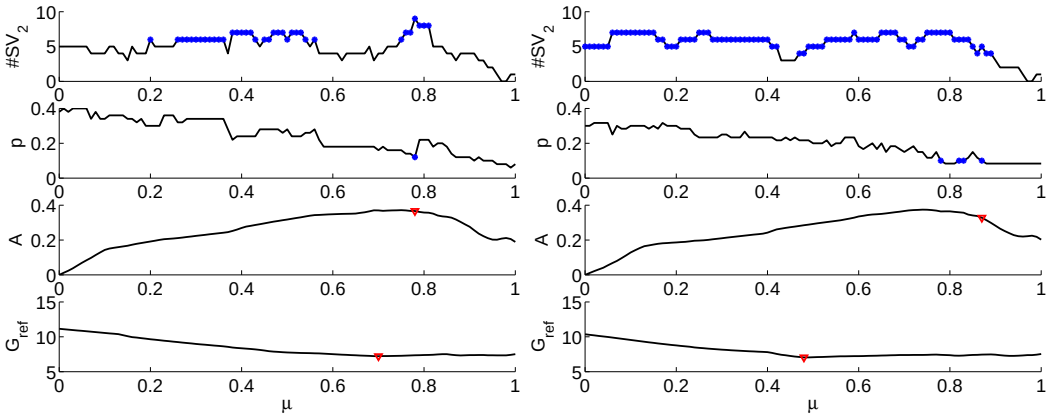
(a) $n_2 = 10$ (b) $n_2 = 20$

3.5.1.4 Decision boundaries change

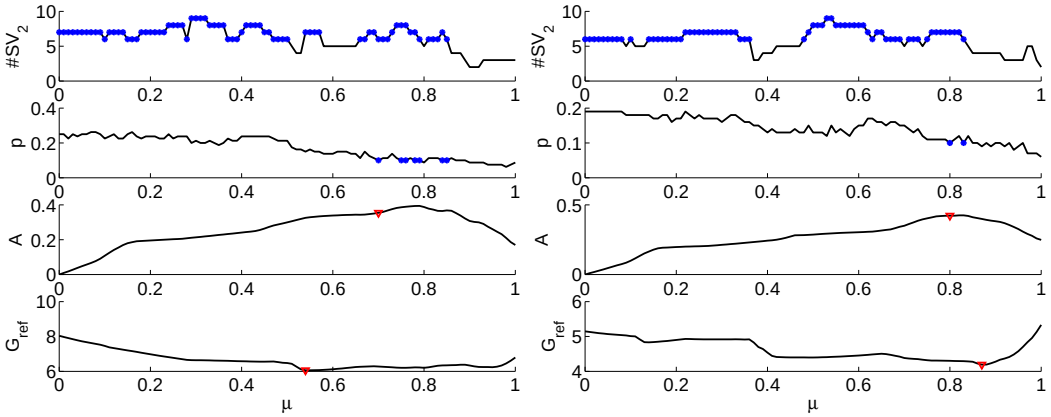
Figure 3.5 shows the boundaries of the different decision function as n_2 increases from 10 to 400. The blue one ($T_2(n_2 = 400)$) is the boundary of the targeted decision function (trained with $\boldsymbol{X}_2$ alone and $n_2 = 400$). When n_2 is small, the decision boundaries of T_2 are very sensitive to the parameter C, here we draw a multi boundaries



(c) $n_2 = 30$ (d) $n_2 = 40$



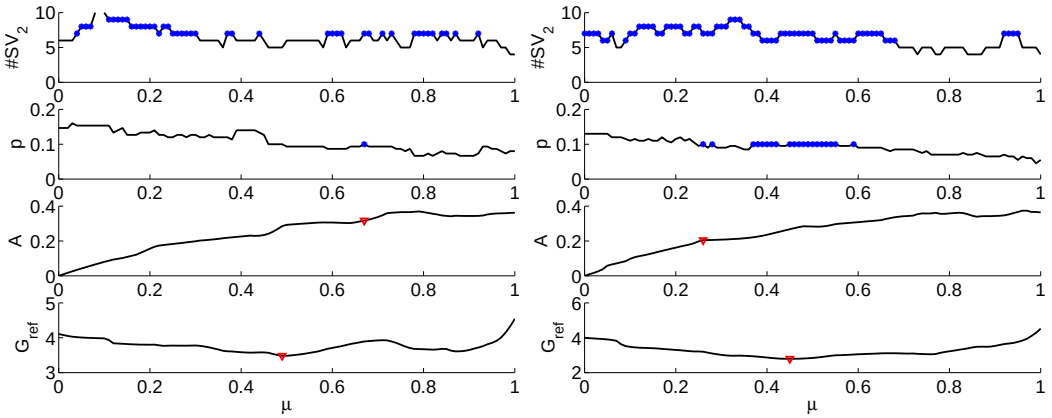
(e) $n_2 = 50$ (f) $n_2 = 60$



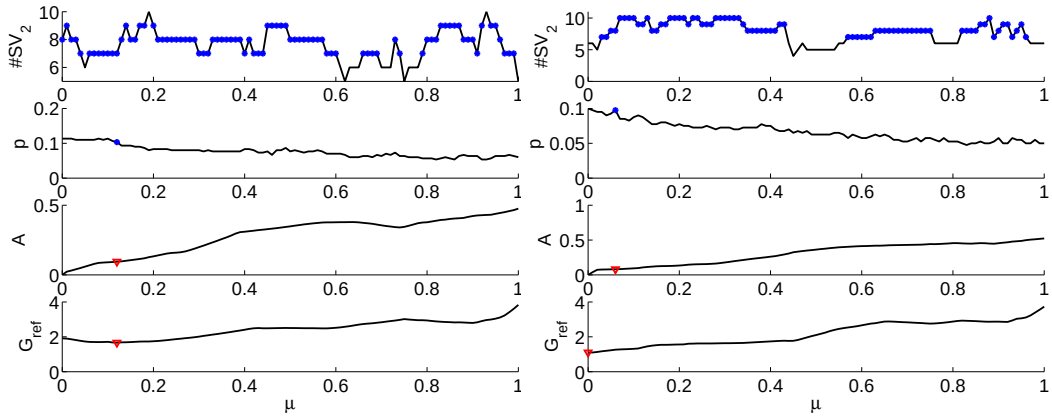
(g) $n_2 = 80$ (h) $n_2 = 100$

of T_2 with $C \in [0.02, 1]$, which changes from small value with exclusive boundaries to large value with inclusive boundaries of all the dataset $\boldsymbol{X}_2$.

When $n_2 = 10$, the decision boundary of MTL(T_2) is almost the same as that of T_{Big}, because here we choose $\mu^* = 0.99$ according to the criteria. As n_2 increases, μ^* tends to decrease a little, which is coherent with the increase of the information weight from $\boldsymbol{X_2}$ and the decrease of that from $\boldsymbol{X_1}$. The MTL(T_2) could detect the boundaries of the new dataset for all values of n_2 as n_2 increases. While the MTL(T_1)



(i) $n_2 = 150$ (j) $n_2 = 200$



(k) $n_2 = 300$ (l) $n_2 = 400$

FIGURE 3.3: Criteria as n_2 increases: values corresponding to the blue star points are candidates μ_a and μ_b in step 1 and 2, values corresponding to the red triangle points in step 3 are μ^* (the final one chosen by the criteria in A), μ_{ref} is the reference value (the red triangle point in G_{ref}).

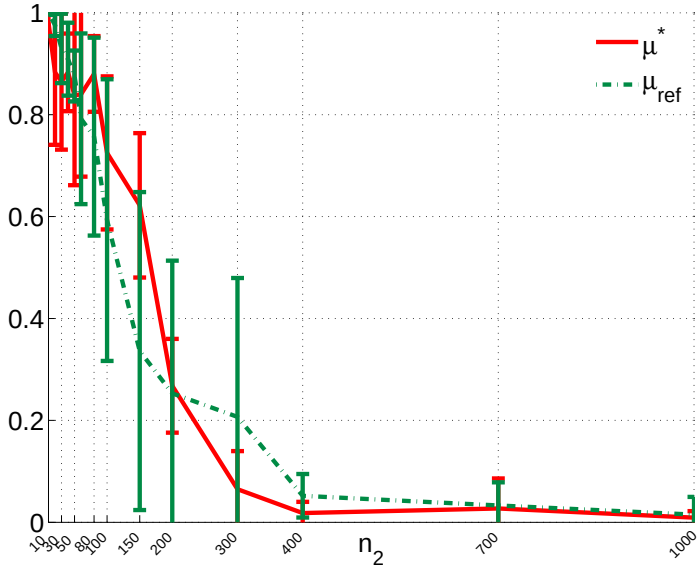


FIGURE 3.4: Selected μ^* vs. reference value μ_{ref}.

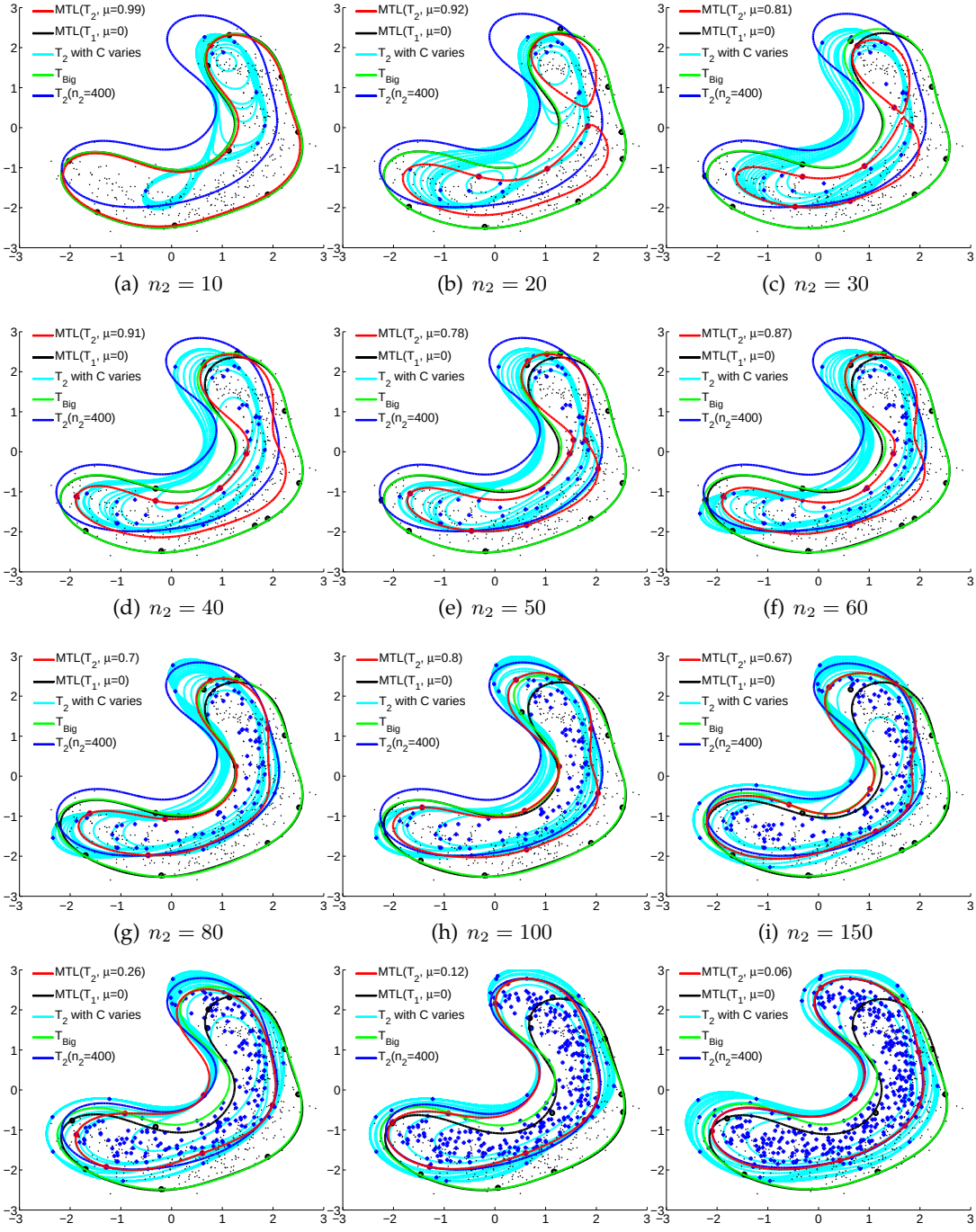


FIGURE 3.5: Decision boundaries for different methods as n_2 increases.

with $\mu = 0$ is the same as the model just based on $\boldsymbol{X}_1$, which cannot detect the new boundaries of the dataset. The T_{Big} method tends to inclose all the dataset of $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, which means it increases the probability of miss alarm. For the method of T_2 with C varies, it tends to cover all the dataset $\boldsymbol{X}_2$ without considering the known structure or the former dataset information when n_2 is small. Besides that it is hard

to tune parameter C for T_2 at that time, while with the proposed approach it defines a good boundary.

3.5.1.5 Performance

The corresponding false alarm rate (figure 5.8(a)) and miss alarm rate (figure 5.8(b)) for MTL(T_2, μ^*) are close to that of MTL(T_2, μ_{ref}). If we use the old detection system (MTL$(T_2, \mu = 0)$, which is independent training based on X_1), it can not follow the distribution change. When $n_2 \geq 300$, the two type errors of MTL(T_2, μ^*) and MTL(T_2, μ_{ref}) are close to that of T_2, which means that when n_2 is large, more representative samples are introduced, a good decision boundary could be obtained based on the new dataset only.

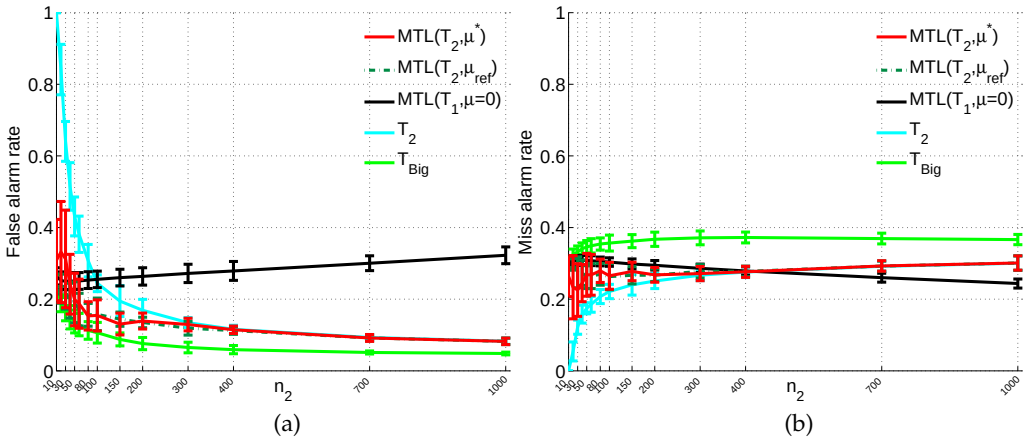


FIGURE 3.6: Two type errors on banana dataset: (a) false alarm rate, (b) miss alarm rate.

From the above analysis, we could say that the proposed heuristic approach for choosing μ is reasonable. And compared to T_{Big} and T_2 alone, the MTL(T_2, μ^*) gives a good transition from the old detection system to the new one when n_2 changes from 10 to 300. When $n_2 \geq 300$, the selected μ^* is very small, less information is needed from the former task, that means we can abandon the old system and just use the new one instead.

3.5.2 Wine quality dataset

Experiments are also conducted on the real dataset: the wine quality dataset by [Cortez et al., 2009].

3.5.2.1 Experimental settings

We consider the dataset restricted to red wine as $\boldsymbol{X_1}$ and the added samples are white wine dataset as $\boldsymbol{X_2}$. We choose the 6th most importance features for red wine to train a SVM model, they are sulphates, pH, total sulfur dioxide, alcohol, volatile acidity and free sulfur dioxide, which are different for white wine to simulate a detection system with changed distribution data samples. For the purpose of estimating performances, we use the wine that classified by wine experts as 3,4 as negative samples and 6,7,8 as positive samples. We set parameter $\sigma = 1.75$, and C is chosen by fixing $\nu = 0.1$ for every initialisation of the kernel adaptation algorithm.

The index of the dataset is shuffled, then we use all the positive red wine samples to simulate data from the old detection system, and white wine data are used to simulate the new detection system. Where $n_1 = 855$, and n_2 goes from 10 to 1,000, the remaining samples are hold for testing, which is shown in table 3.1. Experiments are conducted until all the positives in white wine are taken part in as training data, then results are averaged.

TABLE 3.1: Wine quality dataset

Red wine (old system)	White wine (new system)
855 Positives	3253 Positives (3-fold cross validation)
	183 Negatives

3.5.2.2 Results

Figure 3.7 shows the selected μ^* by the proposed criteria in algorithm (2). It decreases from large value to small value as n_2 increasing from 10 to 1000, which means the multi-task model changes from the one big task model T_{Big} to a much independent model, and the information needed from the former task decreases as n_2 increases.

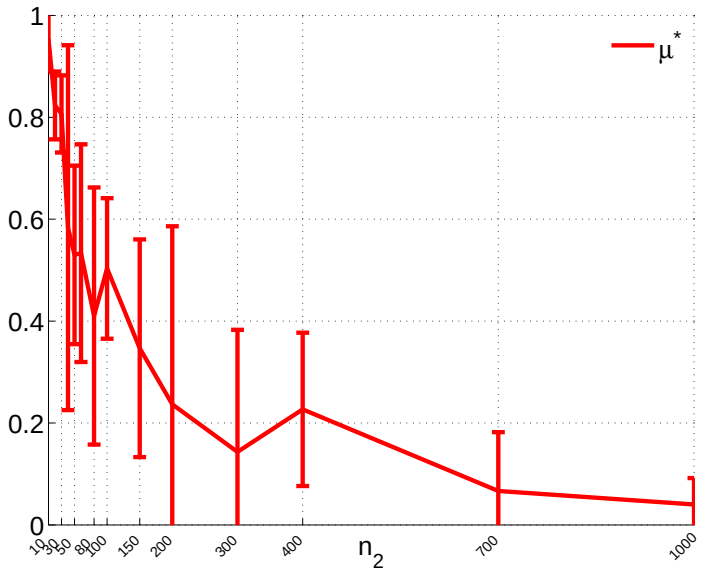


FIGURE 3.7: Selected μ^* on wine quality dataset.

From figure 3.8(a) and figure 3.8(b), we can see that if we use the old detection system MTL($T_2, \mu = 0$), we will have a large proportion of false alarm. If we use T_{Big}, then we have a low false alarm rate, but we increase too much the miss alarm rate, which means we do not detect the change of the dataset. If we use T_2, then we have lower miss alarm rate, but we increase the false alarm rate too much, that means the detection is too restrictive.

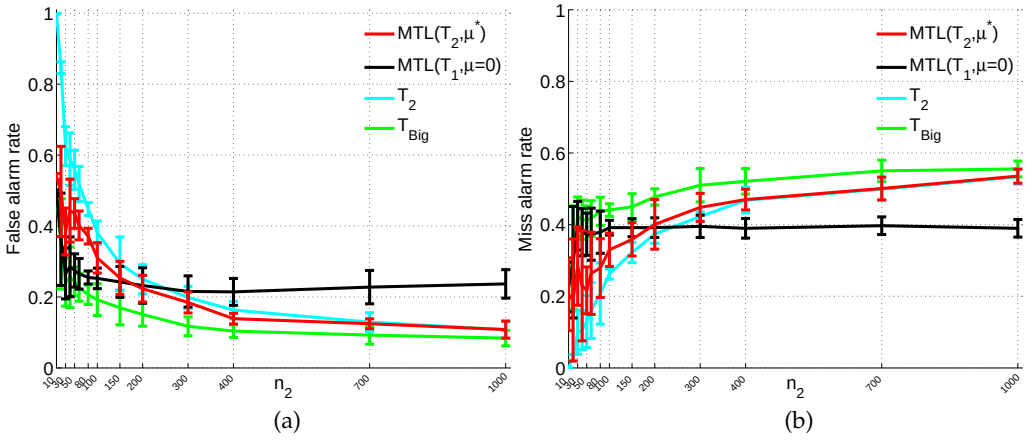


FIGURE 3.8: Two type errors on wine quality dataset: (a) false alarm rate, (b) miss alarm rate.

The MTL(T_2, μ^*) method could reduce the miss alarm rate compared to T_{Big} without increasing the false alarm rate too much compared to that of T_2 when n_2 is small. At last the two type errors of MTL(T_2, μ^*) come close to that of T_2 when we have much more representative samples of $\boldsymbol{X}_2$. So this means that when we do not have

enough samples of the new task T_2, we can have a smooth and acceptable transition from the former detection system to the new one.

3.6 Chapter summary

In this chapter, we proposed a multi-task learning model to solve the one class classification problem to data distribution with a limited change in homogenous space. In this model, a parameter μ is introduced to control the amount of information that is taken into account from the former task T_1. With the method of kernel adaptation for C-one class SVM, we can get all the path solution along μ, then criteria are proposed to choose the proper solution μ^*.

The experiments conducted on toy data and wine quality dataset show that the proposed method could adapt the decision function to the change of dataset and could give a good transition from the former task with dataset $\boldsymbol{X}_1$ to the new task which is just based on the new dataset $\boldsymbol{X}_2$ as n_2 increases gradually. The method could be used to manage a smooth transition between the two tasks and to define the new detection keeping the benefit of a detection system and expanding to limit the performance loss during the transition.

For large-scale problems, one approach could be used to reduce the dimension of the former solution based on approximation approach as in [Wang et al., 2013] before transferring the detection to the new data. But in large scale problems, data gathering is not usually a critical issue. The proposed approach is really useful in case when the sampling rate is slow.

In the next chapter, we will address the transfer learning problem for one-class classification with additional new features, that is to say transfer learning in the heterogenous space situation.

4 Transfer Learning for One-class SVM with Additional New Features

Contents

In this chapter, we will adapt the proposed multi-task learning model in chapter 3 to the situation of transfer learning for one-class SVM with additional new features. It corresponds to the second research problem introduced in chapter 1. In section 5.1, we explain the motivation and present the problem. Then we propose an approach in section 4.2.1 to solve the problem, but that approach comes across some limitation, then an improved approach is proposed in section 4.2.2. Next, experiments and results are commented in section 4.3 to support the method. At last, we give this chapter's conclusion in section 5.6.

4.1 Introduction

From data driven side, the other problem for one-class detection system is the transfer learning problem in heterogeneous space. Here we consider the transfer learning for one-class SVM with additional new features in the target task.

For example, in the application of fault detection for an engine system, there are a few sensors which have already worked on an engine diagnosis system for much time and every sensor gets a few data. Now due to technical or some other practical needs, such as improving detection performances, new sensors are added to this system. As far as we know, this problem has never been tackled in the detection context using one class SVM. After this system change, the object of our work is to keep the false alarm rate to be relatively stable and to reduce the miss alarm rate as much as we can.

Instead of training a new detection system from scratch, multi-task learning seems to be an ideal mean to adapt the former detection to an updated system, since it uses the assumption which is satisfied in our context that related tasks share some common structure or similar model parameters [Evgeniou and Pontil, 2004]. Here we assume one task is the former system and the second one is the updated system.

In [Xue and Beauseroy, 2016], a multi-task learning model (we name it as MTL_{I} here, which corresponds to section 4.2.1) is proposed to solve this detection problem with additional new features. It gives a good transition from the old detection system to the new modified one. However, in some cases the kernel matrix in that model is not positive semi-definite which means that some approximation in a semi-definite subspace must be considered to determine the detection.

Later in [Xue and Beauseroy, 2018b], we proposed an improved approach (we name it as MTL_{II}, which corresponds to section 4.2.2) to avoid that issue. As is shown in section 4.2.2, we can divide the kernel matrix into two part, one part is based on the old features and the second part is based on the new added feature. In order to get a positive semi-definite matrix, typical estimation method can be conducted to fill the corresponding new feature in the old detection system, then a specific variable kernel is used in the second kernel matrix (which is base on the new feature) to

control the impact of the new feature over the detection according to the amount of collected new data.

In this chapter, we will exploit these two methods separately. To study the heterogenous transfer learning problem, we consider the situation of adding new feature one by one in target task to simulate the modification or evolution of an existing detection system.

4.2 Multi-task learning with additional new features

Due to practical reasons, when new feature is added to the old detection system, if we continue to use the old detection system we will not be able to take advantage of the new information to improve the detection performances. If we wait until we gather enough new data to train a new detector which means that on one hand we have to delay the benefit of the update of the system, and on the other hand we have to go through all the hyper parameter optimisation process which may be time consuming. On the contrary, the multi-task learning model should be able to take into consideration the information brought by the new feature.

Define the dataset $\boldsymbol{X}_1 = \{\boldsymbol{x}_{11}, \boldsymbol{x}_{21}, ..., \boldsymbol{x}_{n_11}\}$ from the old detection system T_1 as source datset, where n_1 is the number of samples, and $\boldsymbol{x}_{j1} \in \mathbb{R}^d$, $\mathbb{R}^d$ is the source feature space. Define dataset $\boldsymbol{X}_2 = \{\boldsymbol{x}_{12}, \boldsymbol{x}_{22}, ..., \boldsymbol{x}_{n_22}\}$ from the updated system T_2 as target datset, where n_2 is the number of target data samples, and $\boldsymbol{x}_{j2} \in \mathbb{R}^{d+1}$, $\mathbb{R}^{d+1}$ is the target feature space.

4.2.1 $MTL_{\mathbf{I}}$

Recall that the dual problem for multi-task learning in the same feature space is formulated as:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} -\frac{1}{2}\boldsymbol{\alpha}^T K^{\mu}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{1} \\ &s.t.\ \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \end{aligned} \tag{4.1}$$

where $\boldsymbol{\alpha}^T = [\alpha_{11}, ..., \alpha_{n_1 1}, \alpha_{12}, ..., \alpha_{n_2 2}]$ and

$$K^{\mu} = \begin{bmatrix} K_{ss} & \mu K_{st} \\ \mu K_{st}^T & K_{tt} \end{bmatrix} \tag{4.2}$$

is a modified Gram matrix, $K_{ss} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_1) \rangle$, $K_{st} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_2) \rangle$, $K_{tt} = \langle \phi(\boldsymbol{X}_2), \phi(\boldsymbol{X}_2) \rangle$.

If we apply this model to the situation that $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are from different feature space, problem happens when we compute the block matrix K_{st} because of the different features for the source task and the target task. One possible way is to ignore the new feature only for this block computation. Then

$$K_{st} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_2') \rangle, \tag{4.3}$$

where $\boldsymbol{X}_2' = \{\boldsymbol{x} \backslash x^{(d+1)}\}$ coming from $\boldsymbol{X}_2$ restricted to the d initial features.

To some extend, it gives a balance from the old detection system to the new one by tuning the parameter μ as n_2 increases. However, by using this method (named as MTL_I), the modified kernel matrix is not always positive semi-definite which means that a global optimisation solution can not be guaranteed with standard approach.

4.2.2 $MTL_{\mathbf{II}}$

Another way is to fill the corresponding new feature by using some estimation methods like the nearest neighbour, the imputation etc. Accordingly, we can get $\tilde{\boldsymbol{X}}_1 = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [x^{(1)}, ..., x^{(d)}, \tilde{x}^{(d+1)}]\}$, where $\tilde{x}^{(d+1)}$ is the new feature in the old detection system estimated by using information from $\boldsymbol{X}_2$. The drawback of this method is that when the number of samples $\boldsymbol{X}_2$ for target task is small, it is hard to give a good estimation to the new feature $\tilde{x}^{(d+1)}$ in $\tilde{\boldsymbol{X}}_1$.

Once we get $\tilde{\boldsymbol{X}}_1$ and $\boldsymbol{X}_2$, as we use Gaussian kernel,

$$\begin{aligned} k(\boldsymbol{x}_i, \boldsymbol{x}_j) &= exp(\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{-2\sigma^2}) \\ &= \prod_{l=1}^{d+1} exp(\frac{\|\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}\|^2}{-2\sigma^2}), \end{aligned} \tag{4.4}$$

then the kernel matrix in (4.2) can be decomposed into two part:

$$\begin{aligned} K^\mu &= \begin{bmatrix} K_{ss} & \mu K_{st} \\ \mu K_{st}^T & K_{tt} \end{bmatrix}_{\mathbb{R}^{d+1}} \\ &= \underbrace{\begin{bmatrix} K_{ss} & \mu K_{st} \\ \mu K_{st}^T & K_{tt} \end{bmatrix}_{\mathbb{R}^d}}_{A_0} \circ \underbrace{\begin{bmatrix} \tilde{K}_{ss} & \tilde{K}_{st} \\ \tilde{K}_{st}^T & K_{tt} \end{bmatrix}_{\mathbb{R}^1}}_{A_1}, \end{aligned} \tag{4.5}$$

where $\circ$ is element-wise product and A_0 is kernel matrix based on $\mathbb{R}^d$ with the first dth features for $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, A_1 is kernel matrix based on $\mathbb{R}^1$ space with the $d+1$th estimated feature $\tilde{x}^{(d+1)}$ from $\boldsymbol{X}_1$ and $x^{(d+1)}$ from $\boldsymbol{X}_2$. Notice that K^μ is a positive semi-definite matrix when $\mu \in [0, 1]$, even if different kernel parameters are adopted for computing A_0 and A_1. Besides, we can still use the heuristic approach proposed in chapter 3 to tune the parameter μ for different number of samples in target task.

For the Gaussian kernel, when $\sigma \to +\infty$, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) \to 1$, so we propose to use constant σ_0 for the former system which is based on $\mathbb{R}^d$ subspace and to choose a varying $\sigma(n)$ only for the new feature, where n is the number of samples. At a first intuition, we want $\sigma(n_2)$ to be large when n_2 is small and to be close to σ_0 when n_2 is large.

By doing this, all the entries of matrix A_1 will tend to be 1 when n_2 is small, which means that it does not have very important influence to the total kernel matrix when the estimation of the new feature $\tilde{x}^{(d+1)}$ in $\boldsymbol{X}_1$ is not very dependable. As n_2 becomes larger, more information is brought in from the new feature and a better estimation of $\tilde{x}^{(d+1)}$ will be obtained, more consideration should be taken for matrix A_1, so σ decreases and it converges to the same value as σ_0 when n_2 is large enough.

With the above intuition, question arises how to set this variable kernel parameter σ. In kernel density estimation, the optimal window width for a standard distribution

is given by [Silverman, 1986]:

$$h_{opt} = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} n^{-\frac{1}{d+4}}, \tag{4.6}$$

where d is the number of dimensions (here $d+1$ for our case) and n is the number of samples.

Upon above, if we multiple this optimal window width with proper factors, then we can drive it to decrease from a large value to the constant kernel σ_0. Thus, we define the kernel parameter function for A_1 as:

$$\sigma(n) = c_2 \exp(\frac{c_1}{\sqrt[3]{n}}) h_{opt}, \tag{4.7}$$

where the exponent function $\exp(\frac{c_1}{\sqrt[3]{n}})$ decreases from a large value (when n is small) to a small value (close to 1 when n is large), which means that we multiply h_{opt} by a large number at the beginning and we almost keep h_{opt} when n is large enough. The constant c_1 is used to control the value that we want to multiply h_{opt} when n is small and c_2 is a scale factor that makes $\sigma(n)$ converge to σ_0 when n is large.

A few groups of kernel functions $\sigma_1(n)$, $\sigma_2(n)$, $\sigma_3(n)$ are shown in figure 4.2. For application, the group with larger kernel function can be used if the estimation for the potential new feature is very undependable, while the group with smaller kernel function can be used if it is more dependable. We name this multi-task learning method as MTL_{II} in this chapter.

4.3 Experiments

In this section, experiments are conducted on artificial dataset. We compare the proposed method MTL_{I}, MTL_{II}, as well as the other possible solutions: the old detection system T_1 based on the old features, the new detection system T_2 based on data when new feature is added, and the union detection system T_{Big} which is based on the estimated data $\hat{\boldsymbol{X}}_1$ and the new obtained data $\boldsymbol{X}_2$.

4.3.1 Data descriptions

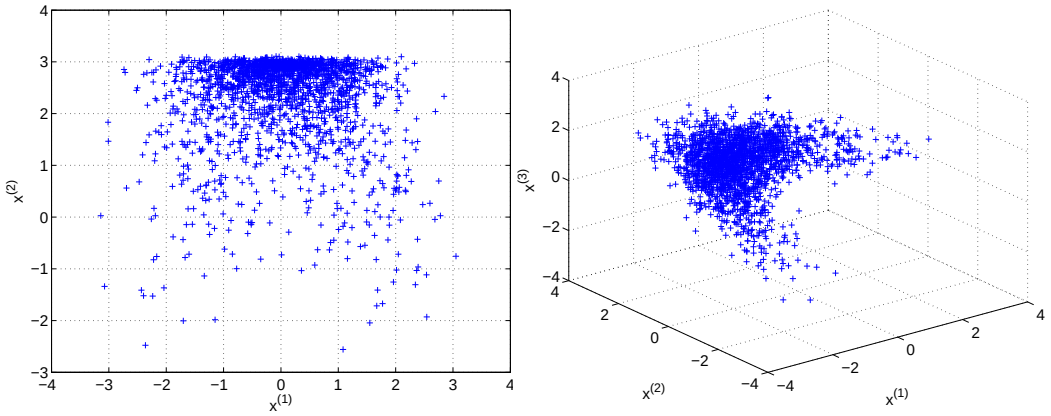
Let $y_1, y_2, y_3, y_4 \sim N(0, 1)$, three features are defined as:

$$x^{(1)} = y_1, \tag{4.8}$$

$$x^{(2)} = 3\cos(\frac{1}{2}y_1 + \frac{1}{2}y_2 + \frac{1}{4}y_3) + N(0, 0.05), \tag{4.9}$$

$$x^{(3)} = y_4, \tag{4.10}$$

where $N(0, 0.05)$ is Gaussian noisy. We use $\boldsymbol{X}_1 = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [x^{(1)}, x^{(2)}]\}$ as the dataset for the old detection system (source task), and $\boldsymbol{X}_2 = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [x^{(1)}, x^{(2)}, x^{(3)}]\}$ as the dataset for the new detection system (target task), here $x^{(3)}$ can be considered as new measurement or new sensor of the detection system. The number of training samples is $n_1 = 200$, and we increase n_2 from 5 to 400 to simulate the data collection of the new detection system. A figure view of the dataset is shown in figure 4.1.



(a) 2d view. (b) 3d view.

FIGURE 4.1: View of the dataset.

To test the performance of the detection system, 20,000 positive samples are generated from X_2 to test the false alarm rate. Besides that, we use 20,000 uniform distribution data which cover the whole test dataset to test the performance of miss alarm rate. Specifically, let $u^{(1)}, u^{(2)}, u^{(3)} \sim U(-4, 4)$, three groups of negative samples are defined as:

1. Uniform distribution for all the features $\boldsymbol{X}_{\text{negI}} = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [u^{(1)}, u^{(2)}, u^{(3)}]\}$.

2. Uniform distribution only for the third dimension $\boldsymbol{X}_{\text{negII}} = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [x^{(1)}, x^{(2)}, u^{(3)}]\}$ to simulate the case where outliers come only from the new added feature.

3. Uniform distribution only for the first two dimensions $\boldsymbol{X}_{\text{negIII}} = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [u^{(1)}, u^{(2)}, x^{(3)}]\}$ to simulate the outliers coming from the old features.

4.3.2 Experimental settings

We choose kernel parameter $\sigma_0 = 1.75$ and $\nu = 0.1$ for ν-OCSVM (it exits a corresponding C for C-OCSVM) which make the proportion of outliers around 0.1 for the old detection system at the beginning. A list of the comparison of different methods is shown in table 4.1. Where $\tilde{\boldsymbol{X}}_1 = \{\boldsymbol{x} \mid \boldsymbol{x}^T = [x^{(1)}, x^{(2)}, \tilde{x}^{(3)}]\}$, $\tilde{x}^{(3)}$ is the estimated feature (we use nearest neighbour method to fill this new feature) and $\boldsymbol{X}_2 \backslash x^{(3)}$ denotes that $\boldsymbol{X}_2$ without the new feature. For T_1, T_2 and T_{Big}, the same kernel parameter σ_0 is used, for MTL_{I} the setting is same as in [Xue and Beauseroy, 2016] and for MTL_{II}, σ_0 is used for the first two features and a variation of $\sigma(n)$ according to (4.7) is used for the third feature. The choice of μ for MTL_{II} is conducted by the criteria proposed in chapter 3. All the results are averaged by 10 times.

TABLE 4.1: Setting of the comparison methods.

Comparison methods	Train datasets
T_1	$\boldsymbol{X}_1$, $\boldsymbol{X}_2 \backslash x^{(3)}$
T_2	$\boldsymbol{X}_2$
T_{Big}	$\tilde{\boldsymbol{X}}_1$, $\boldsymbol{X}_2$
MTL_{I}	$\boldsymbol{X}_1$, $\boldsymbol{X}_2$
MTL_{II}	$\tilde{\boldsymbol{X}}_1$, $\boldsymbol{X}_2$

4.3.3 Performance with different kernel parameters

Three groups of kernel functions $\sigma_1(n_2), \sigma_2(n_2), \sigma_3(n_2)$ are generated to test the performance of MTL_{II}. As shown in figure 4.2, we choose $c_1 = 1, 3, 6$ and then choose corresponding $c_2 = 3.7125, 2.8299, 1.8834$ respectively in (4.7) which makes $\sigma(400) = \sigma_0$ (where $\sigma_0 = 1.75$ is the kernel parameter for the old detection system).

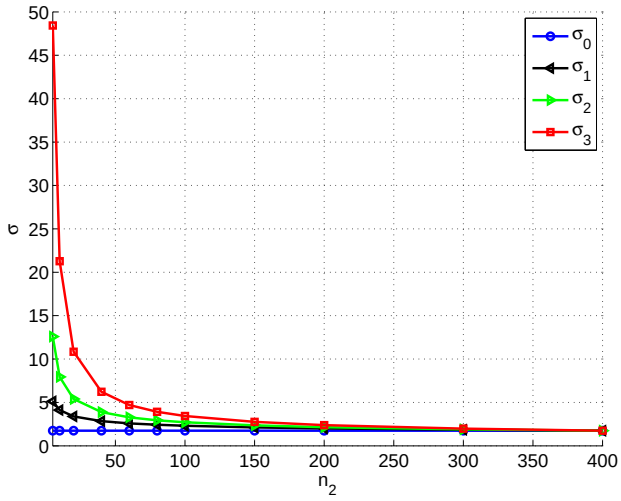


FIGURE 4.2: Different groups of kernel function: (1) the constant one $\sigma_0 = 1.75$; (2) $c_1 = 1$, $c_2 = 3.7125$, which makes $\sigma_1(400) = \sigma_0$; (3) $c_1 = 3$, $c_2 = 2.8299$, which makes $\sigma_2(400) = \sigma_0$; (4) $c_1 = 6$, $c_2 = 1.8834$, which makes $\sigma_3(400) = \sigma_0$.

By using these different kernel functions, how to compute A_0 (which is based on the old features) and A_1 (which is just based on the new feature) is shown in table 4.2.

TABLE 4.2: Kernel functions to compute A_0 and A_1

	A_0	A_1
$MTL_{\text{II}}(\sigma_0)$	σ_0	σ_0
$MTL_{\text{II}}(\sigma_1)$	σ_0	σ_1
$MTL_{\text{II}}(\sigma_2)$	σ_0	σ_2
$MTL_{\text{II}}(\sigma_3)$	σ_0	σ_3

Performance results of MTL_{II} are shown in figure 4.3 with different $\sigma(n_2)$. If we use constant σ_0, the false alarm rate is very high when n_2 is small because of the bad estimation while lack of samples from $\boldsymbol{X}_2$. Both the false alarm rate and the miss alarm rate will become more stable as n_2 increases due to better estimation for $\tilde{x}^{(3)}$. However, with the variation of kernel parameters $\sigma_1, \sigma_2, \sigma_3$, when n_2 is small, the larger σ is, the closer of A_1 is to a matrix with 1 elements (that means we are using a kernel matrix which is very close to the one just based on the old features), so the false alarm rate increases less ($FA_{MTL_{\text{II}}}(\sigma_3) < FA_{MTL_{\text{II}}}(\sigma_2) < FA_{MTL_{\text{II}}}(\sigma_1) < FA_{MTL_{\text{II}}}(\sigma_0)$).

As for the miss alarm rate on $\boldsymbol{X}_{\text{negI}}$ (figure 4.3(b)) (outliers come from all features), the method with variable kernel parameters increases a bit at the beginning and decreases to the same value as we use constant kernel. The same trend happens

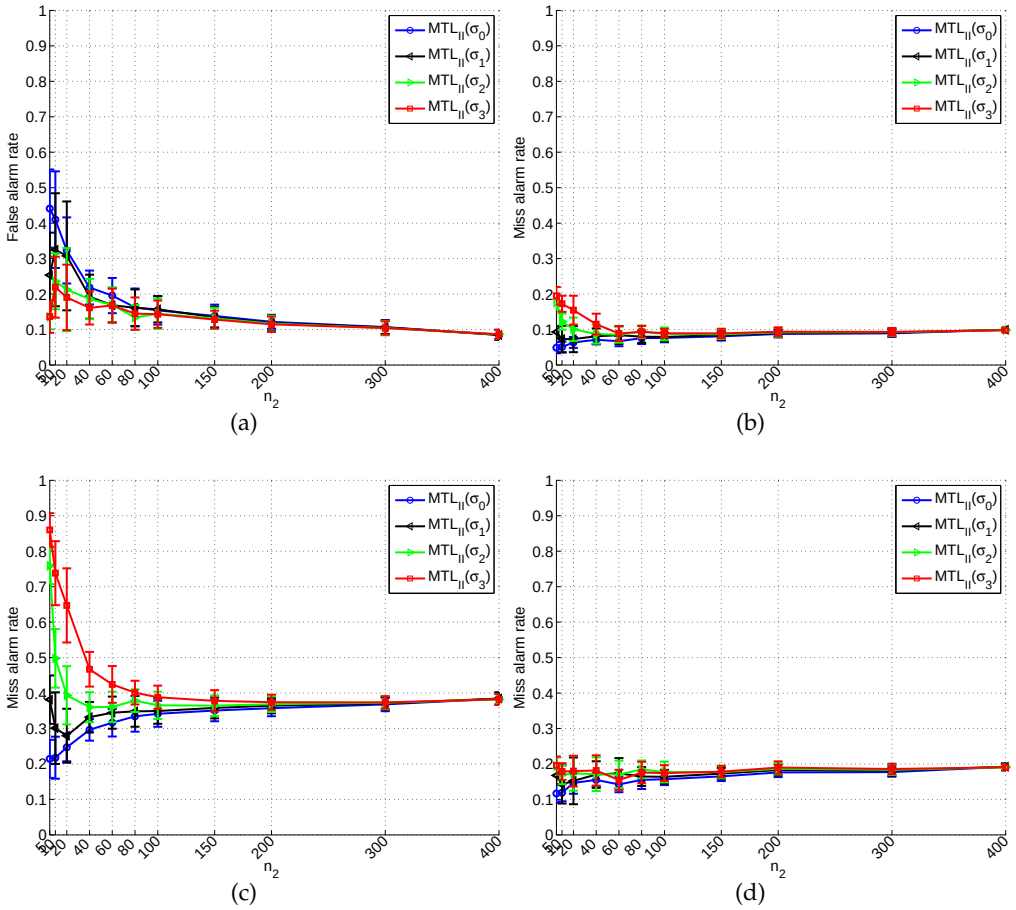


FIGURE 4.3: Results of different kernel parameters for MTL_{II}: (a) false alarm rate, (b) miss alarm rate on $\boldsymbol{X}_{\text{negI}}$ (uniform data for all features), (c) miss alarm rate on $\boldsymbol{X}_{\text{negII}}$ (uniform data only for new feature), (d) miss alarm rate on $\boldsymbol{X}_{\text{negIII}}$ (uniform data only for old features).

on dataset $\boldsymbol{X}_{\text{negII}}$ (figure 4.3(c)) (outliers only come from the new feature) except at the beginning, where the miss alarm rate is relatively high, but as we increase n_2, we decrease σ and the miss alarm rate decreases rapidly to the same value with constant σ_0.

This kind of trend makes meaningful sense because when new feature is added, when n_2 is small, if outliers are only from the new feature, we cannot decide whether they are negative observations or not at the very beginning, instead we would rather keep a relative stable false alarm rate while reducing the miss alarm rate rapidly as n_2 increases which means that we take the new feature's information into consideration gradually.

For the miss alarm rate on $\boldsymbol{X}_{\text{negIII}}$ (figure 4.3(d)) (outliers are constrained to the old

features), all methods keep relatively stable which means that we do not increase nor decrease the miss alarm rate if the outliers come from the old features which is expected.

Among all these groups, $MTL_{\text{II}}(\sigma_3)$ produces a relatively good detection model when new feature is added, where σ_3 is relatively large at the beginning and it converges to σ_0 at the end.

4.3.4 Comparative study

We use $MTL_{\text{II}}(\sigma_3)$ to compare with the other possible methods listed in table 4.1, results are reported in figure 4.4. Besides that, in order to study the problem that might happen is the data mismatch problem for the old feature space (that means the data distribution for the old features may experience a change due to system maintenance or update), we give a rotation of $\frac{\pi}{6}$ to the first two features in $\boldsymbol{X}_2$ to study the model's performance on this situation, and the results are shown in figure 4.5.

For the method T_1, which is trained on the old features of $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, the false alarm rate is almost constant around 0.1, but the miss alarm rate is the highest one among all the other methods because it does not take into consideration of the new feature.

For T_2 which is based only on $\boldsymbol{X}_2$ since the new feature is added, it gives very high false alarm rate when n_2 is small, which means that it does not make full use of the information from the former detection system at the beginning, as n_2 increases large enough (here $n_2 > 150$), it produces more stable false alarm rate and miss alarm rate.

Thus, both T_1 and T_2 will converge to the desired false alarm rate and the miss alarm rate decreases (as n_2 increases) to a lower bound related to the problem itself except for T_1 when outliers are from the new feature. For T_2, it has a rough start and it takes time to gather more data to get stable performance.

If we combine the estimated dataset $\tilde{\boldsymbol{X}}_1$ and $\boldsymbol{X}_2$ to train a detection model, named as T_{Big}, the false alarm rate is lower than that of T_2, and the miss alarm rate will end up with the same as T_2. However, with a rotation of the first two features in $\boldsymbol{X}_2$,

it will increase the chance of miss alarm at the end (which is shown in figure 4.5(b), 4.5(c) and 4.5(d)), because T_{Big} tends to enclose all the training dataset together. That means T_{Big} is not practical when data distribution of the old features experiences a change in the new detection system, even a small change.

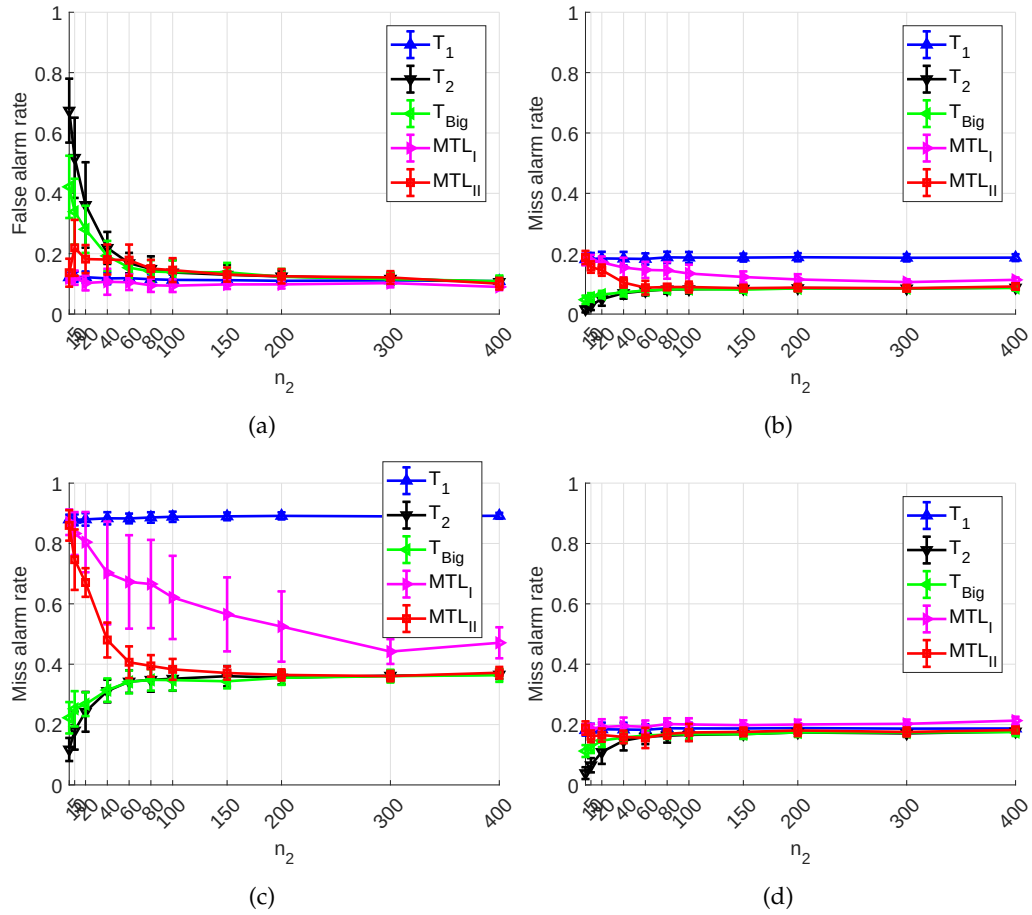


FIGURE 4.4: Compare results of different methods: (a) false alarm rate, (b) miss alarm rate on $\boldsymbol{X}_{\text{negI}}$ (uniform data for all features), (c) miss alarm rate on $\boldsymbol{X}_{\text{negII}}$ (uniform data only for new feature), (d) miss alarm rate on $\boldsymbol{X}_{\text{negIII}}$ (uniform data only for old features).

For multi-task learning method, both MTL_{I} and MTL_{II} gives a transition from the old detection system T_1 (which is just based on the old features) to the new modified system T_2 (which is based on the new dataset $\boldsymbol{X}_2$ since new feature is added) as n_2 increases. The false alarm rate of MTL_{I} is a bit lower than that of MTL_{II}, and both of them are relatively stable compared to T_2 and T_{Big}. But for miss alarm rate, only MTL_{II} converges to that of T_2 while MTL_{I} does not as n_2 increases. In addition, the general miss alarm rate of MTL_{II} is much lower than that of MTL_{I}, this difference is much larger when original features' distribution also changes after system update

(figure 4.5). Therefore, compared to MTL_{I}, MTL_{II} gives a better transition from the old detection system to the new one, it can keep the false alarm rate relatively stable while decrease the miss alarm rate rapidly to a stable value.

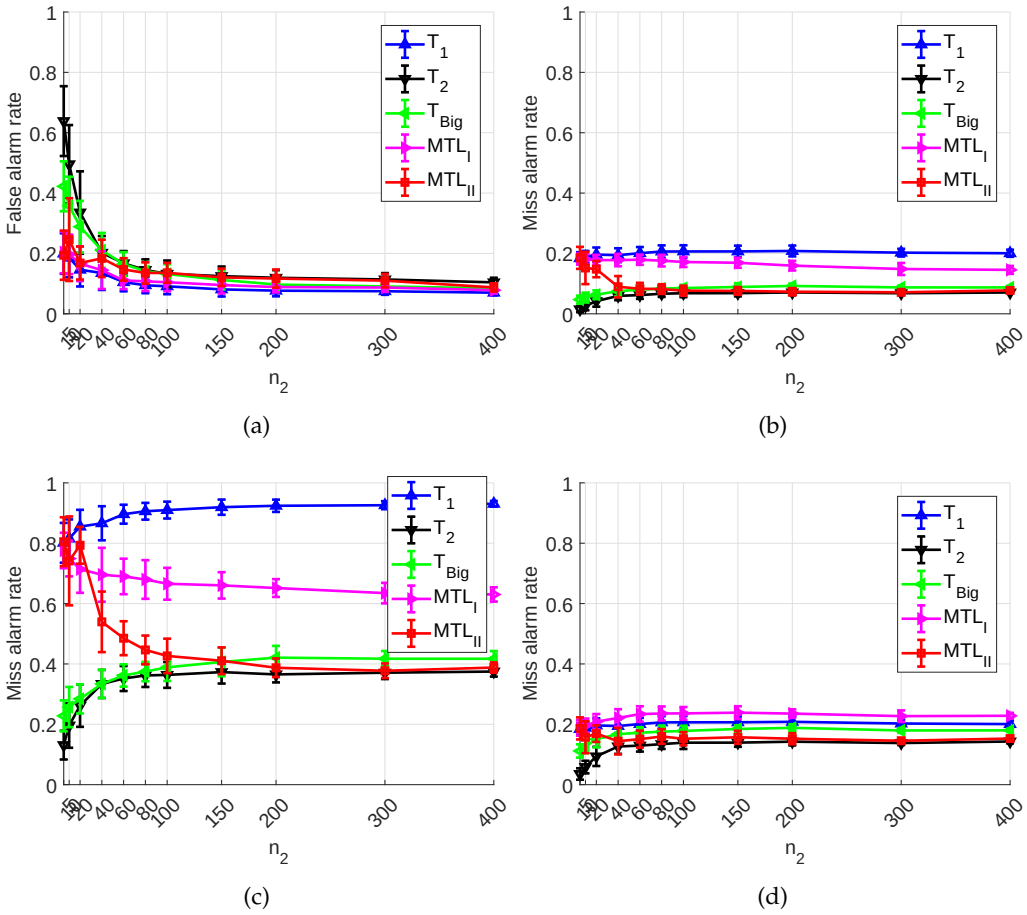


FIGURE 4.5: Compare results with $\frac{\pi}{6}$ rotation in $\boldsymbol{X}_2$ for the first two features: (a) false alarm rate, (b) miss alarm rate on $\boldsymbol{X}_{\text{negI}}$ (uniform data for all features), (c) miss alarm rate on $\boldsymbol{X}_{\text{negII}}$ (uniform data only for new feature), (d) miss alarm rate on $\boldsymbol{X}_{\text{negIII}}$ (uniform data only for old features).

As a result, we can conclude that:

- T_1 will increase the miss alarm rate if outliers are from the new added feature.
- T_2 has a rough start, especial when the number of samples is limited, it takes time to collect more data to get stable performance.
- T_{Big} has much larger miss alarm rate because it tends to enclose all the training samples which fails to detect the evolving of the data distribution.
- To some extend, both MTL_I and MTL_{II} give a transition from the old system

to the new one. But MTL_{II} converges faster than MTL_I especially when there is a data mismatch problem for the old features (the update of the system).

4.4 Chapter summary

In this chapter, we use the multi-task learning idea to solve the transfer learning problem for one-class SVM when new features are added. Two approaches, MTL_{I} and MTL_{II} are proposes to tackle such problem.

MTL_{I} deletes the new feature when computes the cross matrix K_{st} between source task and target task. To some extend, it gives a transition from the old one to the new one because we still keep the new feature in K_{tt}. If we tune the parameter μ properly as n_2 increases, this approach gives a certain balance from the old detection system to the new one. However, one drawback of this method is that the kernel matrix is not always positive semi-definite which means that a global optimisation can not be guaranteed with standard approach.

MTL_{II} is an opposite mean because it tends to fill the potential new feature in the old detection system. As the kernel matrix of multi-task learning model can be divided into two parts, one part is based on the former features and the other part is based on the new feature. Typical methods can be used to estimate the potential new feature in the source dataset in order to compute the kernel matrix based on that new feature. When the number of samples is small, no matter what kind of estimation method is used, it is hard to get a good estimation. As a result, a variable kernel is used to balance the importance of the new features with the number of new samples and at last it converges to the same value as used in the old detection system.

Experimental results show that both MTL_{I} and MTL_{II} gives a transition from the old detection system to the new modified one, and their performances are better than other possible solutions when the detection system experiences a measurement change. Furthermore, MTL_{II} outperforms MTL_{I}, and it gives a good transition from the old system to the new one with additional new features.

As the proposed multi-task learning method in chapter 3 and chapter 4 can be solved by classical C one-class SVM, in the next chapter we will study the online learning problem for one-class SVM.

5 Constant False Alarm Rate for Online One-class SVM

Contents

As we can use the classical C one-class SVM to solve the proposed multi-task learning model, the online learning problem for one-class SVM will be studied in this chapter.

In section 5.1, we will introduce the motivation for online learning with constant false alarm rate. Following that, in section 5.2 we will study the online learning for C one-class SVM, and its constraint parameter adaptation in the purpose of getting a relatively stable false alarm rate. In section 5.3, we will propose a more direct way for constant false alarm rate: online ν one-class SVM. Experiments are conducted to compare these methods in section 5.4. At last we conclude this chapter in section 5.6.

5.1 Introduction

In real applications, time series data are usually encountered rather than batch mode data. Results may be assessed after a given delay which enables to add the data to the training set in order to improve the detection function. For that type of situation, online learning algorithm is required rather than classical batch learning mode. Furthermore, the multi-task learning model proposed in chapter 3 can be solved by classical one-class SVM, it would be nice to develop an online learning approach for one-class SVM.

The typical online learning methods for SVM are proposed firstly for two class classification, the idea is to follow the changes of Lagrange parameters as the weight of the new sample increases until the Karush-Kuhn-Tucker conditions are satisfied [Cauwenberghs and Poggio, 2000; Diehl and Cauwenberghs, 2003; Laskov et al., 2006; Tax and Laskov, 2003] and it can be applied to the problem of one class classification situation [Tax and Laskov, 2003]. For one-class SVM, the main problem of the above online learning method is that they do not adapt the constraint parameter C according to the number of training samples. What happens is that the false alarm rate decreases and the miss alarm rate increases gradually as the number of training samples n increases.

The motivation of this chapter is to develop an online one-class SVM algorithm with stable performance as new samples are added. According to Neyman-Pearson Lemma, the most powerful hypothesis test is the one with minimum type II error by given a level of type I error (significance level) [Tong, Feng, and Zhao, 2016]. For one class classification (outliers detection) paradigm, the type I error is the false alarm rate which is usually a user-specified level and the type II error is the miss alarm rate which we want to minimize. As the online learning procedure, for one class SVM we want to keep a relatively stable level of false alarm rate without increasing the miss alarm rate too much.

The following possible solutions can be used to tackle the above problem:

(1) Adapt parameter C for C-OCSVM according to the proportion of outliers.

(2) Adapt parameter C for C-OCSVM according to ν, by using the equivalent relationship between C-OCSVM and ν-OCSVM, as ν gives an upper bound on the fraction of outliers.

(3) Develop an online ν-OCSVM learning method with fixed ν as it gives an upper bound on the proportion of outliers.

The former two approaches need two stages, one is the online learning procedure and the other stage is the parameter adaptation which are kind of complex and not straightforward, while the third one is a direct approach.

5.2 Online C-OCSVM and constraint parameter adaptation

Let $\{\boldsymbol{x}_i, i = 1, ..., n\}$ be the training dataset with n samples and $\boldsymbol{x}_{n+1}$ is the new coming data sample. Remember that the formulation of C-one class SVM is:

$$\begin{aligned} &\min_{\boldsymbol{w}^C, \boldsymbol{\xi}_i^C} \frac{1}{2}\|\boldsymbol{w}^C\|^2 + C\sum_{i=1}^{n}\xi_i^C \\ &s.t.\ \langle \boldsymbol{w}^C, \phi(\boldsymbol{x}_i)\rangle \geq 1 - \xi_i^C,\ \xi_i^C \geq 0, i = 1, 2, ..., n. \end{aligned} \tag{5.1}$$

Introducing the Lagrange multipliers $\boldsymbol{\alpha}^C$, it can be shown that:

$$\boldsymbol{w}^C = \sum_i \alpha_i^C \phi(\boldsymbol{x}_i), \tag{5.2}$$

where $\phi(\boldsymbol{x}_i)$ is an implicit mapping from the original space to the Hilbert feature space. Then the dual problem can be cast as:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}^C} \frac{1}{2}\sum_{i,j}\alpha_i^C\alpha_j^C K_{i,j} - \sum_i \alpha_i^C \\ &s.t.\ 0 \leq \alpha_i^C \leq C. \end{aligned} \tag{5.3}$$

where K is kernel matrix with entry $K_{i,j} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle$.

5.2.1 Online C-OCSVM

The Karush-Kuhn-Tucker conditions of problem (5.1) could be given as:

$$f_i = \sum_{j=1}^{n} K_{i,j}\alpha_j^C - 1 \begin{cases} > 0, & \alpha_i^C = 0, \\ = 0, & 0 \leq \alpha_i^C \leq C, \\ < 0, & \alpha_i^C = C, \end{cases} \tag{5.4}$$

where f_i is the separating function used for decision $g(\boldsymbol{x}_i) = sign(f_i)$, corresponding to training data $\boldsymbol{x}_i$. Then three groups of Lagrangian multipliers index $\mathcal{M}$, $\mathcal{E}$ and $\mathcal{C}$ could be defined as:

- $\mathcal{M} = \{i : f(\boldsymbol{x}_i) = 0,\ \alpha_i^C \in [0, C]\}$ for samples on the margin.
- $\mathcal{E} = \{i : f(\boldsymbol{x}_i) < 0,\ \alpha_i^C = C\}$ for samples outside the decision region.
- $\mathcal{C} = \{i : f(\boldsymbol{x}_i) > 0,\ \alpha_i^C = 0\}$ for samples inside the decision region.

Let m, e, c denote the number of elements in $\mathcal{M}, \mathcal{E}, \mathcal{C}$. Similar to the idea of online learning for two class SVM [Diehl and Cauwenberghs, 2003]. When new sample comes, we define the corresponding index as set $\mathcal{U}$. For initialization, let $\alpha_i = 0$, for $i \in \mathcal{U}$, it will be assigned to set $\mathcal{C}$, if $f_i > 0$ or to set $\mathcal{M}$, if $f_i = 0$. When $f_i < 0$, for $i \in \mathcal{U}$, we need to increase α_i until the new data sample index will eventually be assigned to group sets $\mathcal{C}$, or margin vectors $\mathcal{M}$, or error vectors $\mathcal{E}$, while monitoring the group change of other Lagrangian parameters to satisfy the KKT condition.

For one class SVM, if we express the separating function differentially, we have:

$$\triangle f_i = \sum_{l \in \mathcal{U}} K_{i,l} \triangle \alpha_l^C + \sum_{k \in \mathcal{M}} K_{i,k} \triangle \alpha_k^C, \forall i \in \mathcal{M}. \tag{5.5}$$

Let

$$\triangle \alpha_k^C = \beta_k \triangle \gamma, \forall k \in \mathcal{M}, \tag{5.6}$$

$$\triangle \alpha_l^C = C \triangle \gamma, \forall l \in \mathcal{U}, \tag{5.7}$$

where $\triangle\gamma$ is the increasing rate, β_k is the specific slope for α_k^C.

Then divide equation (5.5) by $\triangle\gamma$:

$$\delta_i = \frac{\triangle f_i}{\triangle \gamma} = \sum_{l \in \mathcal{U}} K_{i,l} C + \sum_{k \in \mathcal{M}} K_{i,k} \beta_k, \forall i \in \mathcal{M}. \tag{5.8}$$

For $i \in \mathcal{M}$, we have $\triangle f_i = 0$, then according to (5.8):

$$\boldsymbol{\beta} = -C K_{\mathcal{M}\mathcal{M}}^{-1} K_{\mathcal{M}\mathcal{U}} \mathbf{1}, \tag{5.9}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_m]^T$, and $K_{\mathcal{M}\mathcal{M}}$ denotes the reduced kernel matrix with index elements in $\mathcal{M}$.

The online learning process comes to detect the following movements of points that may happen:

(1) For all $i \in \mathcal{M}$, two kinds of situations may happen: one is the movement of candidate points from $\mathcal{M}$ to $\mathcal{E}$ which indicates that $\triangle\alpha^C = C - \alpha_i^C$, the other one is the movement of candidate points from $\mathcal{M}$ to $\mathcal{C}$ which indicates that $\triangle\alpha^C = -\alpha_i^C$. According to equation (5.6), then:

$$\triangle\gamma = \begin{cases} \frac{C-\alpha_i^C}{\beta_i}, & \text{if } \beta_i > 0, i \in \mathcal{M}, \\ \frac{-\alpha_i^C}{\beta_i}, & \text{if } \beta_i < 0, i \in \mathcal{M}. \end{cases} \tag{5.10}$$

(2) For all $i \in \mathcal{C}$, the only possible movement of candidate points is from $\mathcal{C}$ to $\mathcal{M}$ which indicates that the separating function f_i changes from positive to 0, that means $\triangle f_i = -f_i$, according to equation (5.8)

$$\triangle\gamma = \frac{-f_i}{\delta_i}, \delta_i < 0, i \in \mathcal{C}. \tag{5.11}$$

(3) For all $i \in \mathcal{E}$, the only possible movement of candidate points is from $\mathcal{E}$ to $\mathcal{M}$ which indicates that the separating function f_i changes from negative to 0, according to equation (5.8)

$$\triangle\gamma = \frac{-f_i}{\delta_i}, \delta_i > 0, i \in \mathcal{E}. \tag{5.12}$$

(4) For all $i \in \mathcal{U}$ which satisfies $f_i < 0$, two kinds of situations may happen: one is the movement from $\mathcal{U}$ to $\mathcal{M}$ which indicates that the separating function changes from negative to 0, the other movement is from set $\mathcal{U}$ to $\mathcal{E}$ which indicates that α_i^C reaches C. According to equation (5.8) and (5.7), then:

$$\triangle\gamma = \begin{cases} \frac{-f_i}{\delta_i}, & \text{if } \delta_i > 0, i \in \mathcal{U}, \\ \frac{C-\alpha_i^C}{C}, & i \in \mathcal{U}. \end{cases} \tag{5.13}$$

Based on the above events, $\triangle\gamma_{min}$ is computed and then the minimum perturbation of the KKT condition is computed by:

$$\alpha_k^C := \alpha_k^C + \beta_k \triangle\gamma_{min}, \forall k \in \mathcal{M}, \tag{5.14}$$

and:

$$\alpha_l^C := \alpha_l^C + C\triangle\gamma_{min}, \forall l \in \mathcal{U}. \tag{5.15}$$

Repeat the above procedure until the set of $\mathcal{U}$ is empty, which means the new arrived sample is assigned to either of the three groups $\mathcal{M}$, $\mathcal{E}$ or $\mathcal{C}$.

5.2.2 Constraint parameter adaptation

For the above online C-OCSVM learning algorithm, the problem is that we cannot get a relatively stable performance as the false alarm rate decreases and the miss alarm rate increases gradually when new samples are added. To deal this problem, we can either adapt the parameter C according to the proportion of outliers or according to parameter ν by using the equivalent relationship between C-OCSVM and ν-OCSVM, as ν gives an upper bound on the proportion of outliers.

In order to get a piece-liner solution to facilitate the parameter adaptation, let $C = \frac{1}{\lambda}$ in the formulation of C-OCSVM, then (5.1) can be rewritten as:

$$\begin{gathered} \min_{\boldsymbol{w}^C, \boldsymbol{\xi}} \frac{\lambda}{2}\|\boldsymbol{w}^C\|^2 + \sum_{i=1}^{n} \xi_i^C \\ s.t.\ \langle \boldsymbol{w}^C, \phi(\boldsymbol{x}_i)\rangle \geq 1 - \xi_i^C,\ \xi_i^C \geq 0, i = 1, 2, ..., n. \end{gathered} \tag{5.16}$$

We note this formulation (5.16) as $\lambda-$OCSVM. Introduce the Lagrange multipliers $\boldsymbol{\alpha}^{\lambda}$, then:

$$\boldsymbol{w}^{C} = \frac{1}{\lambda}\sum_{i}\alpha_i^{\lambda}\phi(\boldsymbol{x}_i). \tag{5.17}$$

According to (5.2) and (5.17):

$$\alpha_i^{C} = \frac{\alpha_i^{\lambda}}{\lambda}. \tag{5.18}$$

The dual problem is:

$$\begin{gathered}\min_{\boldsymbol{\alpha}^{\boldsymbol{\lambda}}} \lambda\left(\frac{1}{2}\sum_{i,j}\frac{\alpha_i^{\lambda}}{\lambda}\frac{\alpha_j^{\lambda}}{\lambda}\langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle - \sum_{i}\frac{\alpha_i^{\lambda}}{\lambda}\right) \\ s.t.\ 0 \leq \alpha_i^{\lambda} \leq 1.\end{gathered} \tag{5.19}$$

which leads to the same solution as (5.3). Then the decision function is:

$$g(\boldsymbol{x}) = sign(f(\boldsymbol{x})), \tag{5.20}$$

where

$$f(\boldsymbol{x}) = \frac{1}{\lambda}\sum_{j}\alpha_j^{\lambda}\langle\phi(\boldsymbol{x}_j),\phi(\boldsymbol{x})\rangle - 1. \tag{5.21}$$

According to [Lee and Scott, 2007], α_i^{λ} has a piecewise-linear solution with λ, when $\lambda_l > \lambda > \lambda_{l+1}$, then:

$$\alpha_k^{\lambda} = \alpha_k^{\lambda_l} - (\lambda_l - \lambda)b_k, k \in \mathcal{M}, \tag{5.22}$$

where $\alpha_k^{\lambda_l}$ is the Lagrangian parameter corresponding to the lth breakpoint and $\boldsymbol{b} = \boldsymbol{K}_l^{-1}\mathbf{1}$, $\boldsymbol{K}_l$ is the $m \times m$ matrix such that $[\boldsymbol{K}_l]_{ij} = \langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle$ for $i, j \in \mathcal{M}$.

One way to adapt the regularization parameter C is by adjusting λ which makes the empirical proportion of outliers equal the given value. This is the first approach that we mentioned in section 5.1.

The second way is to adapt C by fixing ν, as ν gives an upper bound on the fraction of outliers. According to equation (2.34), (2.36) and (5.18):

$$\begin{aligned}\nu &= \frac{1}{nC\rho} \\ &= \frac{1}{n\frac{1}{\lambda}\frac{1}{\sum_i \alpha_i^C}} \\ &= \frac{1}{n\frac{1}{\lambda}\frac{1}{\sum_i \alpha_i^\lambda/\lambda}} \\ &= \frac{\sum_i \alpha_i^\lambda}{n}.\end{aligned} \tag{5.23}$$

Then:

$$\begin{aligned}n\nu &= \sum_i \alpha_i^\lambda \\ &= \sum_{j\in\mathcal{E}} \alpha_j^\lambda + \sum_{k\in\mathcal{M}} \alpha_k^\lambda \\ &= e + \sum_{k\in\mathcal{M}} \left(\alpha_k^{\lambda_l} - (\lambda_l - \lambda)b_k\right) \\ &= e + \sum_{k\in\mathcal{M}} \alpha_k^{\lambda_l} - \lambda_l \sum_{k\in\mathcal{M}} b_k + \lambda \sum_{k\in\mathcal{M}} b_k.\end{aligned} \tag{5.24}$$

As a result, when $\lambda_l > \lambda > \lambda_{l+1}$, then:

$$\lambda = \frac{n\nu - e - \sum_{k\in\mathcal{M}} \alpha_k^{\lambda_l} + \lambda_l \sum_{k\in\mathcal{M}} b_k}{\sum_{k\in\mathcal{M}} b_k}. \tag{5.25}$$

Substitute $C = \frac{1}{\lambda}$ and (5.18) into (5.25), then:

$$C = \frac{C_l \sum_{k\in\mathcal{M}} b_k}{C_l n\nu - C_l e - \sum_{k\in\mathcal{M}} \alpha_k^{C_l} + \sum_{k\in\mathcal{M}} b_k}. \tag{5.26}$$

Notice that λ has a piecewise-linear relationship with ν. It means that we can adjust parameter C for C-OSVM with a reference of ν for ν-OSVM, which gives an estimation of the proportion of outliers. The minimum of ν is:

$$\begin{aligned}\nu_{\min} &= \frac{1}{n}\left(e + \sum_{k\in\mathcal{M}} \alpha_k^{\lambda_l} - \Delta\lambda \sum_{k\in\mathcal{M}} b_k\right) \\ &= \frac{1}{n}\left(e + \frac{1}{C_l}\sum_{k\in\mathcal{M}} \alpha_k^{C_l} - \frac{C_{l+1} - C_l}{C_l C_{l+1}} \sum_{k\in\mathcal{M}} b_k\right).\end{aligned} \tag{5.27}$$

where $\Delta\lambda = \lambda_l - \lambda_{l+1}$. The maximum of ν is:

$$\begin{aligned} \nu_{\max} &= \frac{1}{n}\left(e + \sum_{k\in\mathcal{M}} \alpha_k^{\lambda_l}\right) \\ &= \frac{1}{n}\left(e + \frac{1}{C_l}\sum_{k\in\mathcal{M}} \alpha_k^{C_l}\right). \end{aligned} \tag{5.28}$$

Once we get an online solution, we can compute the corresponding ν according to (5.23) to check if it satisfies the user defined one. Otherwise, we can compute $[\nu_{\min}, \nu_{\max}]$, then decide to increase C or decrease C until the condition is satisfied. The general algorithm for parameter adaptation of online learning is shown in algorithm 3.

Algorithm 3 Parameter adaptation for online learning

Input:
Solutions $\boldsymbol{\alpha}_{n_1}^{C_1}$ when the number of samples is n_1, the new coming data $\boldsymbol{x}_{\text{new}}$, predefined ν.
Output:
Solutions $\boldsymbol{\alpha}_{n_2}^{C_2}$ when $n_2 = n_1 + 1$.
Compute $\boldsymbol{\alpha}_{n_2}^{C_1}$ using the online learning techniques in section 5.2.1.
Compute $\nu_{\min}$ and $\nu_{\max}$ according to (5.27) and (5.28).
while $\nu \notin [\nu_{\min}, \nu_{\max}]$ **do**
  if $\nu > \nu_{\max}$ **then**
    Decrease C (increase λ) to find the next breakpoint.
    Compute $\nu_{\min}$ and $\nu_{\max}$ according to (5.27) and (5.28).
  else $\{\nu < \nu_{\min}\}$
    Increase C (decrease λ) to find the next breakpoint.
    Compute $\nu_{\min}$ and $\nu_{\max}$ according to (5.27) and (5.28).
  end if
end while
Compute C_2 and $\boldsymbol{\alpha}_{n_2}^{C_2}$ according to (5.26), (5.18) and (5.22).

5.3 A more direct way: online ν-OCSVM

The third way is to develop an online version of ν-OCSVM as parameter ν gives an upper bound on the fraction of outliers, which is shown in equation (2.14).

5.3.1 The main problem

Consider the situation that we have the solution of a dataset $\{\boldsymbol{x}_i, i = 1, ..., n\}$ with n samples, and $\boldsymbol{x}_{n+1}$ is the new added sample. We can rewrite the formulation of ν-OCSVM in (2.6) as follows:

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\xi},\rho} \frac{(n+\gamma)\nu}{2}\|\boldsymbol{w}\|^2 - (n+\gamma)\nu\rho + \sum_{i=1}^{n}\xi_i + \gamma\xi_{n+1} \\ &s.t.\ \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle \geq \rho - \xi_i, \xi_i \geq 0, \rho \geq 0, i = 1, ..., n+1, \end{aligned} \tag{5.29}$$

where γ changes from 0 to 1, which indicates the online learning procedure from n to $n+1$ samples. When $\gamma = 1$ the solution of (5.29) is the same as that of (2.6) with $n+1$ samples.

Introducing the Lagrange multipliers $\boldsymbol{\alpha}$, we can get:

$$\boldsymbol{w} = \frac{1}{(n+\gamma)\nu}\sum_{i=1}^{n+1}\alpha_i\phi(\boldsymbol{x}_i). \tag{5.30}$$

Then the dual problem can be written as:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \frac{1}{2(n+\gamma)\nu}\sum_{i,j}\alpha_i\alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &s.t. \begin{cases} 0 \leq \alpha_i \leq 1, i = 1, ..., n, 0 \leq \alpha_{n+1} \leq \gamma, \\ \sum_{i=1}^{n+1}\alpha_i = (n+\gamma)\nu. \end{cases} \end{aligned} \tag{5.31}$$

For one value of γ, we can get a corresponding solution $\boldsymbol{\alpha}^\gamma$. Then decision function is defined as:

$$g^\gamma(\boldsymbol{x}) = sign(f^\gamma(\boldsymbol{x})), \tag{5.32}$$

where

$$f^\gamma(\boldsymbol{x}) = \frac{1}{(n+\gamma)\nu}\sum_{i=1}^{n+1}\alpha_i^\gamma K(\boldsymbol{x}, \boldsymbol{x}_i) - \rho^\gamma, \tag{5.33}$$

and $\rho^\gamma = \frac{1}{(n+\gamma)\nu}\sum_{i=1}^{n+1}\alpha_i^\gamma K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ when $f^\gamma(\boldsymbol{x}_j) = 0$. As a consequence of KKT conditions, n indexes can be partitioned into 3 sets which are piecewise linear preliminary developments.

- $\mathcal{M} = \{i : f^\gamma(\boldsymbol{x}_i) = 0,\ \alpha_i^\gamma \in [0,1]\}$, for $i = 1, ..., n$.
- $\mathcal{E} = \{i : f^\gamma(\boldsymbol{x}_i) < 0,\ \alpha_i^\gamma = 1\}$, for $i = 1, ..., n$.
- $\mathcal{C} = \{i : f^\gamma(\boldsymbol{x}_i) > 0,\ \alpha_i^\gamma = 0\}$, for $i = 1, ..., n$.

For the new coming data $\boldsymbol{x}_{n+1}$, we define:

- $n+1 \in \mathcal{M}$, if $f^\gamma(\boldsymbol{x}_{n+1}) = 0$ and $\alpha_{n+1}^\gamma \in [0, \gamma]$.
- $n+1 \in \mathcal{E}$, if $f^\gamma(\boldsymbol{x}_{n+1}) < 0$ and $\alpha_{n+1}^\gamma = \gamma$.
- $n+1 \in \mathcal{C}$, if $f^\gamma(\boldsymbol{x}_{n+1}) > 0$ and $\alpha_{n+1}^\gamma = 0$.

Assuming the composition of the 3 groups does not change as $\gamma \in [\gamma^-, \gamma^+]$ for given solution α^{γ^-} and ρ^-. Then according to (5.31):

$$\begin{cases} \sum_{k\in\mathcal{C}} \alpha_k^{\gamma^-} + \sum_{k\in\mathcal{M}} \alpha_k^{\gamma^-} + \sum_{k\in\mathcal{E}} \alpha_k^{\gamma^-} = (n+\gamma^-)\nu, & \text{(5.34a)} \\ \sum_{k\in\mathcal{C}} \alpha_k^{\gamma} + \sum_{k\in\mathcal{M}} \alpha_k^{\gamma} + \sum_{k\in\mathcal{E}} \alpha_k^{\gamma} = (n+\gamma)\nu. & \text{(5.34b)} \end{cases}$$

Let (5.34b)-(5.34a), then:

$$\sum_{k\in\mathcal{M}} \alpha_k^{\gamma} - \sum_{k\in\mathcal{M}} \alpha_k^{\gamma^-} = \begin{cases} (\gamma-\gamma^-)(\nu-1), & n+1 \in \mathcal{E}, \\ (\gamma-\gamma^-)\nu, & n+1 \notin \mathcal{E}. \end{cases} \tag{5.35}$$

Define $\alpha_0^\gamma = (n+\gamma)\nu\rho^\gamma$, then (5.33) can be rewritten as:

$$\begin{aligned} f^\gamma(\boldsymbol{x}) &= f^\gamma(\boldsymbol{x}) - \frac{n+\gamma^-}{n+\gamma} f^{\gamma^-}(\boldsymbol{x}) + \frac{n+\gamma^-}{n+\gamma} f^{\gamma^-}(\boldsymbol{x}) \\ &= \frac{1}{(n+\gamma)\nu}\Bigg(\sum_{i=1}^{n+1} (\alpha_i^\gamma - \alpha_i^{\gamma^-}) K(\boldsymbol{x}, \boldsymbol{x}_i) \\ &\quad - (\alpha_0^\gamma - \alpha_0^{\gamma^-}) + (n+\gamma^-)\nu f^{\gamma^-}(\boldsymbol{x}) \Bigg). \end{aligned} \tag{5.36}$$

5.3.2 Determination of $\boldsymbol{\alpha}^\gamma$

Let $\boldsymbol{\alpha}_\mathcal{M}$ be the vector with Lagrange multipliers α_k, $k \in \mathcal{M}$.

When $f^{\gamma}(\boldsymbol{x}_{n+1}) > 0$ $(\mathcal{C})$ or $f^{\gamma}(\boldsymbol{x}_{n+1}) = 0$ $(\mathcal{M})$, for $l \in \mathcal{M}$, we have $f^{\gamma}(\boldsymbol{x}_l) = f^{\gamma^-}(\boldsymbol{x}_l) = 0$. According to (5.35) and (5.36):

$$\begin{cases} K_{\mathcal{M}\mathcal{M}}(\boldsymbol{\alpha}_{\mathcal{M}}^{\gamma} - \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma^-}) - (\boldsymbol{\alpha}_0^{\gamma} - \boldsymbol{\alpha}_0^{\gamma^-})\mathbf{1} = \mathbf{0}, \\ \mathbf{1}^T(\boldsymbol{\alpha}_{\mathcal{M}}^{\gamma} - \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma^-}) = (\gamma - \gamma^-)\nu. \end{cases} \tag{5.37}$$

Let $A = \begin{bmatrix} K_{\mathcal{M}\mathcal{M}} & -\mathbf{1} \\ -\mathbf{1}^T & 0 \end{bmatrix}$, $\boldsymbol{c}^T = [0...0, 1]$, then (5.37) can be written as:

$$\begin{pmatrix} \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma} \\ \alpha_0^{\gamma} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma^-} \\ \alpha_0^{\gamma^-} \end{pmatrix} + (\gamma - \gamma^-)\nu\boldsymbol{v}, \tag{5.38}$$

where $\boldsymbol{v} = A^{-1}\boldsymbol{c}$.

When $f^{\gamma}(\boldsymbol{x}_{n+1}) < 0$ $(\mathcal{E})$, similarly for $l \in \mathcal{M}$, $f^{\gamma}(\boldsymbol{x}_l) = f^{\gamma^-}(\boldsymbol{x}_l) = 0$, according to (5.35) and (5.36), then:

$$\begin{cases} K_{\mathcal{M}\mathcal{M}}(\boldsymbol{\alpha}_{\mathcal{M}}^{\gamma} - \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma^-}) - (\boldsymbol{\alpha}_0^{\gamma} - \boldsymbol{\alpha}_0^{\gamma^-})\mathbf{1} = (\gamma^- - \gamma)K_{\mathcal{M},n+1}, \\ \mathbf{1}^T(\boldsymbol{\alpha}_{\mathcal{M}}^{\gamma} - \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma^-}) = (\gamma - \gamma^-)(\nu - 1), \end{cases} \tag{5.39}$$

where $K_{\mathcal{M},n+1} = [K(\boldsymbol{x}_{l_1}, \boldsymbol{x}_{n+1}), ..., K(\boldsymbol{x}_{l_{|\mathcal{M}|}}, \boldsymbol{x}_{n+1}), 0]^T$. Introducing A and $\boldsymbol{c}$, then:

$$\begin{pmatrix} \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma} \\ \alpha_0^{\gamma} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha}_{\mathcal{M}}^{\gamma^-} \\ \alpha_0^{\gamma^-} \end{pmatrix} + (\gamma - \gamma^-)\boldsymbol{u}, \tag{5.40}$$

where $\boldsymbol{u} = A^{-1}\left((\nu - 1)\boldsymbol{c} - K_{\mathcal{M},n+1}\right)$.

5.3.3 Partition change detection

As shown in (5.38) and (5.40), $\boldsymbol{\alpha}^{\gamma}$ is a piecewise linear function of γ. The breakpoints correspond to partition change in accordance to KKT conditions. Let $\Delta\gamma = \gamma^+ - \gamma^-$ be the step, during the online learning procedure, the following events need to be detected.

(1) For all $k \in \mathcal{M}$: two situations may happen. One is the movement of candidate points from $\mathcal{M}$ to $\mathcal{C}$ which indicates that $\alpha_k^\gamma = 0$. According to (5.38) and (5.40):

$$\Delta\gamma_k = \begin{cases} -\frac{\alpha_k^{\gamma^-}}{\nu v_k}, & n+1 \notin \mathcal{E}, k \in \mathcal{M}, \\ -\frac{\alpha_k^{\gamma^-}}{u_k}, & n+1 \in \mathcal{E}, k \in \mathcal{M}. \end{cases} \tag{5.41}$$

The other one is the movement of candidate points from $\mathcal{M}$ to $\mathcal{E}$ which indicates that $\alpha_k^\gamma = 1$ except when $n+1 \in \mathcal{M}$, where $\alpha_{n+1}^\gamma = \gamma^+$. Accordingly,

$$\Delta\gamma_k = \begin{cases} \frac{1-\alpha_k^{\gamma^-}}{\nu v_k}, & n+1 \in \mathcal{C} \text{ or } \mathcal{M}, k \in \mathcal{M}, k \neq n+1, \\ \frac{\alpha_{n+1}^{\gamma^-}-\gamma^-}{1-\nu v_{n+1}}, & n+1 \in \mathcal{M}, k = n+1, \\ \frac{1-\alpha_k^{\gamma^-}}{u_k}, & n+1 \in \mathcal{E}, k \in \mathcal{M}. \end{cases} \tag{5.42}$$

(2) For all $k \in \mathcal{C}$: the movement of candidate points from $\mathcal{C}$ to $\mathcal{M}$ indicates that the function changes from $f^\gamma(\boldsymbol{x}_k) > 0$ to $f^\gamma(\boldsymbol{x}_k) = 0$. According to (5.36), (5.38) and (5.40):

$$\Delta\gamma_k = \begin{cases} -\frac{(n+\gamma^-)f^{\gamma^-}(\boldsymbol{x}_k)}{[K_{k,\mathcal{M}},-1]\boldsymbol{v}}, & n+1 \notin \mathcal{E}, k \in \mathcal{C}, \\ -\frac{(n+\gamma^-)f^{\gamma^-}(\boldsymbol{x}_k)}{[K_{k,\mathcal{M}},-1]\boldsymbol{u}+K_{k,n+1}}, & n+1 \in \mathcal{E}, k \in \mathcal{C}, \end{cases} \tag{5.43}$$

where $K_{k,\mathcal{M}}=[K(\boldsymbol{x}_k,\boldsymbol{x}_{l_1}), ..., K(\boldsymbol{x}_k,\boldsymbol{x}_{l_{|\mathcal{M}|}})]$, $K_{k,n+1}=K(\boldsymbol{x}_k,\boldsymbol{x}_{n+1})$.

(3) For all $k \in \mathcal{E}$: the movement of candidate points from $\mathcal{E}$ to $\mathcal{M}$ indicates that the function changes from $f^\gamma(\boldsymbol{x}_k) < 0$ to $f^\gamma(\boldsymbol{x}_k) = 0$. The corresponding equations are the same as (5.43) except that $f^{\gamma^-}(\boldsymbol{x}_k) < 0$.

Then move to γ^+ according to the smallest $\Delta\gamma{>}0$ until $\gamma^+ = 1$.

5.4 Experiments

5.4.1 Experimental settings

Experiments are conducted on toy and real datasets respectively.

The toy datasets are banana, square and spiral shaped data [Hoffmann, 2007], which are shown in figure 5.1. For each one, we use 1,000 samples for training and 10,000

for testing. In order to test the miss alarm rate, 10,000 negative uniform distribution samples are generated which cover the maximum and minimum boundaries of the toy data. That means for given level of false alarm rate, we should choose the classifier with the minimum miss alarm rate (one which encloses the training data with the minimum volume is the tightest one).

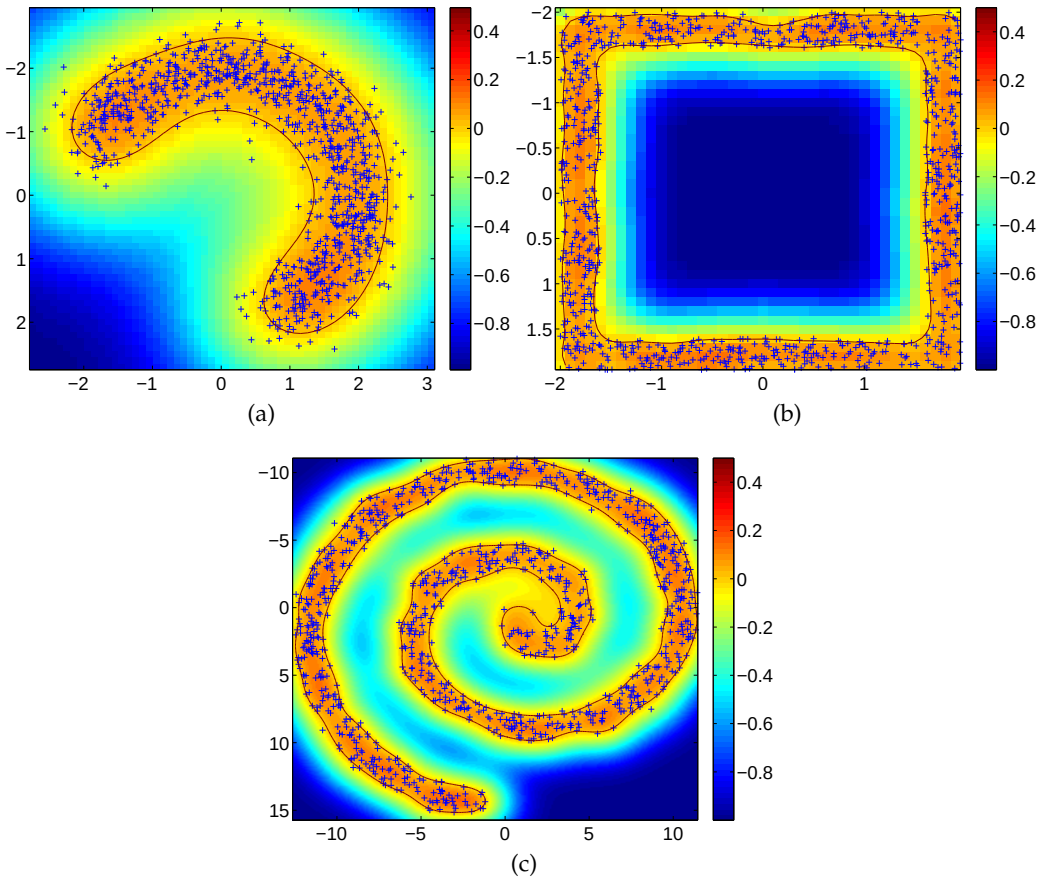


FIGURE 5.1: Contours on toy datasets (n=1,000): (a) banana ($\sigma = 1.06$), (b) square ($\sigma = 0.3$), (c) spiral ($\sigma = 1.5$).

The real world datasets are ionosphere and handwritten digits data [Lichman, 2013]. Ionosphere contains 225 positives, 126 negatives and with handwritten digits we define 1,134 positives (digits 1,4) and 4,489 negatives (the others). For online learning, the initial number of samples is 10 and we add them one by one. Both use 80% for training and 20% for testing.

Four possible methods are compared with the proposed online ν-OCSVM learning, which is shown in table 5.1. For C-OCSVM with fixed C, two groups of results are given using two different C values $C_{\max}$ and $C_{\min}$. Where $C_{\max}$ corresponds

to solutions with $\nu = 0.1$ when $n = 10$ is the initial number of samples and $C_{\min}$ corresponds to that when $n = N$ is the total number of training samples.

Another approach for C-OCSVM is to adapt C when n increases. As depicted earlier, two methods fall into this framework, one is to tune C so that the proportion of outliers equals a chosen value p, noted as $C(p)$ (we choose $p = 0.1$ as the same to ν). The other way is to adapt C according to ν, which is proposed in section 5.2 (result of this method is the same as ν-OCSVM, we just use ν-OCSVM to denote in the experiments). Alternatively, we can also adapt C by using the approximation relation $C = \frac{1}{n\nu}$ (the true relation is $C = \frac{1}{n\nu\rho}$ for C-OCSVM and ν-OCSVM). The kernel parameter (Gaussian kernel) is chosen by setting $\nu = 0.1$ which makes the proportion of outlier to be close to that value.

TABLE 5.1: Parameter setting for different methods (where p is the proportion of outliers for training data).

	$\nu - OCSVM$	$C - OCSVM$
Parameter setting	$\nu = 0.1$	$C(p)$ $C_{\max}(\nu, n = 10)$ $C_{\min}(\nu, n = N)$ $C = \frac{1}{n\nu}$

5.4.2 Results

Performances are evaluated by considering false alarm rate (FA), miss alarm rate (MA), AUC curve and true alarm rate (TA) when FA is enforced to 0.1 by tuning the threshold.

The results on toy datasets are shown in figure 5.2, 5.3 and 5.4. From figure 5.2, it shows that the FA and MA of the proposed learning ν-OCSVM and the one C-OCSVM by adapting C according to p are relative stable (FA≈0.1, MA≈0.27) when $n > 100$. While the methods using fixed C suffer a gradually decrease of FA and increase of MA as n increases. By adapting C using $C = \frac{1}{n\nu}$, the two type errors are relative stable but are very different from the target value. For the curves of AUC and TA by enforcing FA=0.1, the ν-OCSVM and C-OCSVM($C(p)$) are always at the top along the online learning procedure. The same trend happens on square dataset (figure 5.3) and spiral dataset (figure 5.4) except that they need more data samples

($n > 500$) to get a relatively stable performance for ν-OCSVM and C-OCSVM($C(p)$) as the data shape is more complex than that of banana dataset.

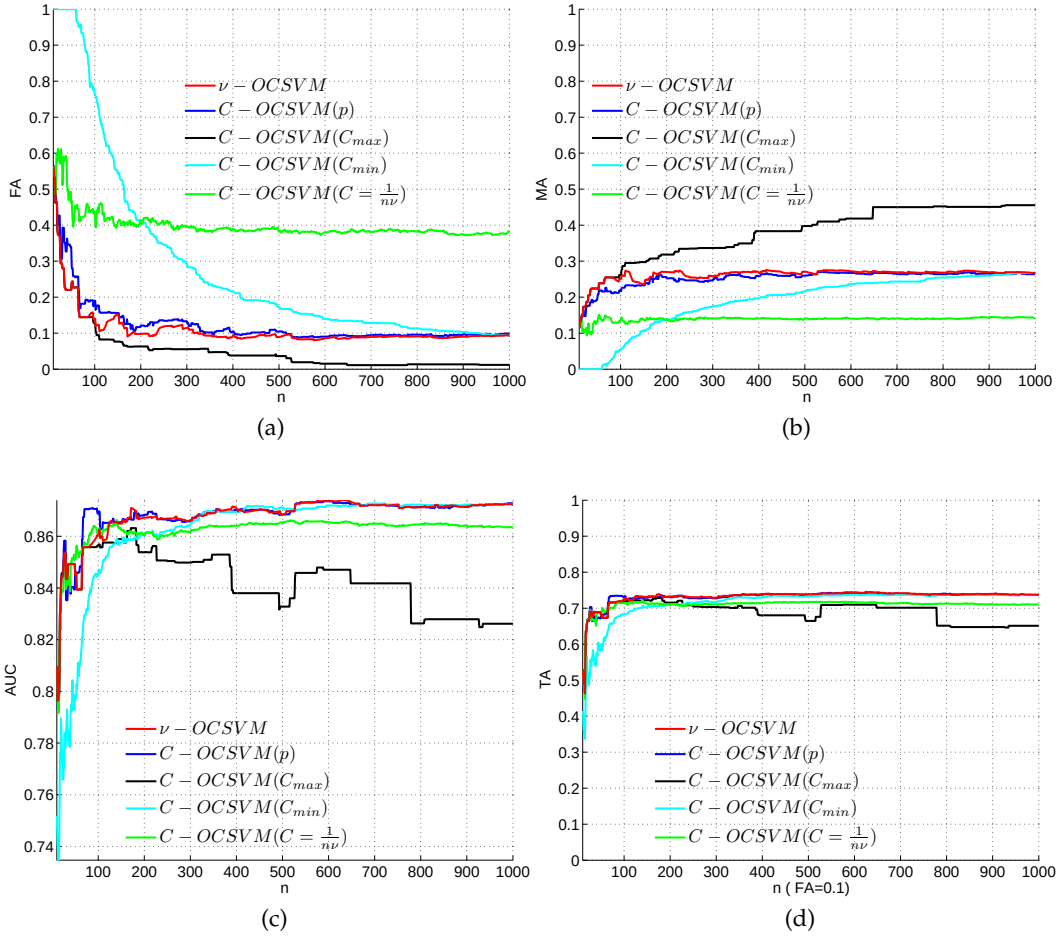
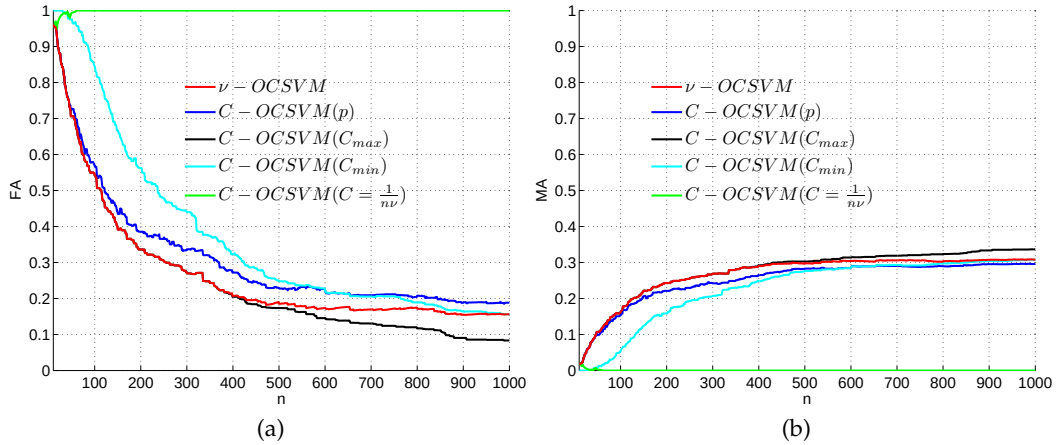
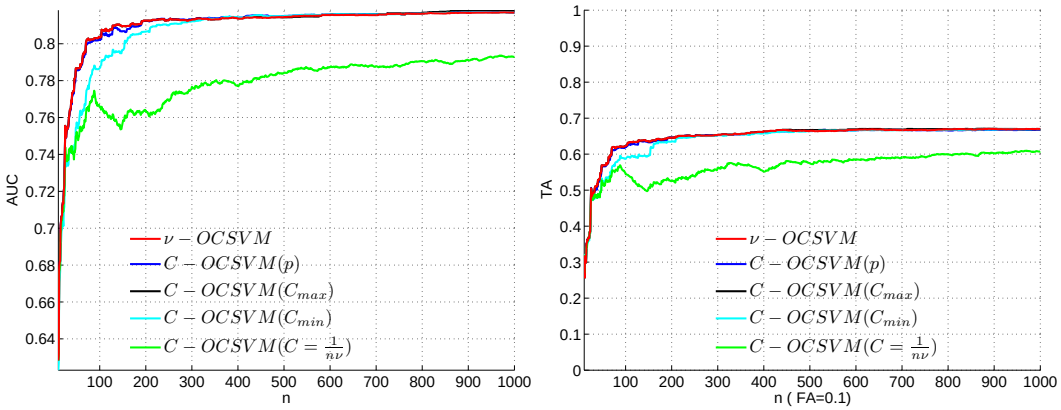


FIGURE 5.2: Banana data (a) false alarm rate, (b) miss alarm rate, (c) AUC, (d) true alarm rate (FA=0.1).

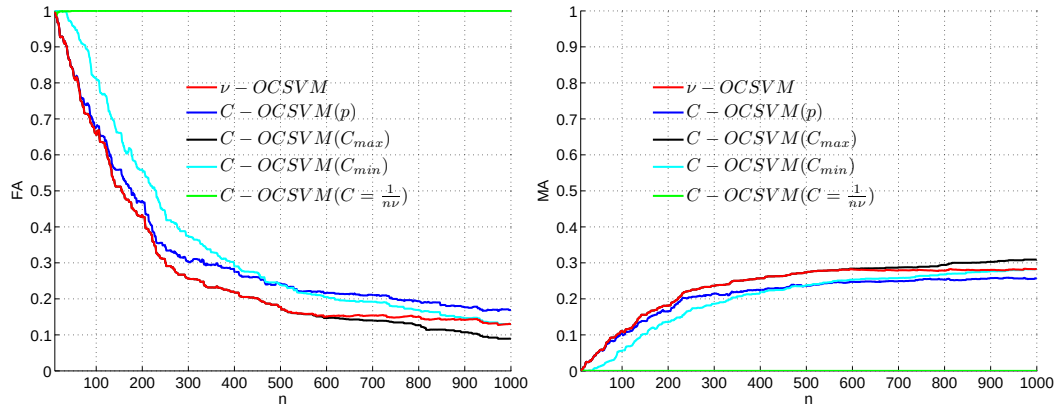


Results on the real datasets are shown in figure 5.5 and figure 5.6. For ionosphere dataset (figure 5.5), the false alarm rate of ν-OCSVM and C-OCSVM($C(p)$) converges to a relatively stable value 0.1 more early (when $n > 80$) than other methods while

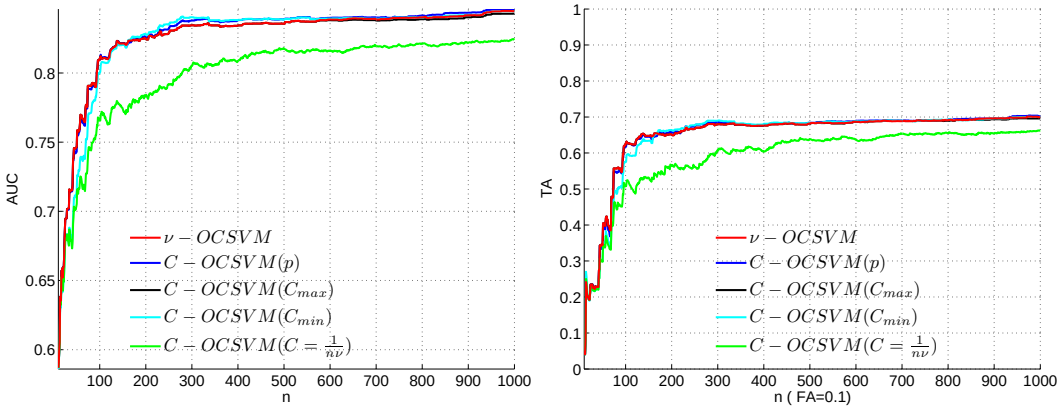


(c) (d)

FIGURE 5.3: Square data (a) false alarm rate, (b) miss alarm rate, (c) AUC, (d) true alarm rate (FA=0.1).



(a) (b)



(c) (d)

FIGURE 5.4: Spiral data (a) false alarm rate, (b) miss alarm rate, (c) AUC, (d) true alarm rate (FA=0.1).

the miss alarm rate is almost the same for all the methods. Similar to the results on toy dataset, the curves of AUC and TA by enforcing FA=0.1, the ν-OCSVM and C-OCSVM($C(p)$) are always at the top along the online learning procedure. For digits data (figure 5.6), although the AUC performance of C-OCSVM($C = \frac{1}{n\nu}$) is the best one, but it is very close to others except for C-OCSVM($C_{\max}$); as for TA, all have almost the same performance except the one with C-OCSVM($C_{\max}$).

Upon above, we can conclude that ν-OCSVM, C-OCSVM($C(p)$) always tend to produce stable performances. Besides that the method C-OCSVM($C(p)$) requires additional computation to select C and change from $C(n)$ to $C(n+1)$ when adding a new sample, which is not as efficient and direct as ν-OCSVM.

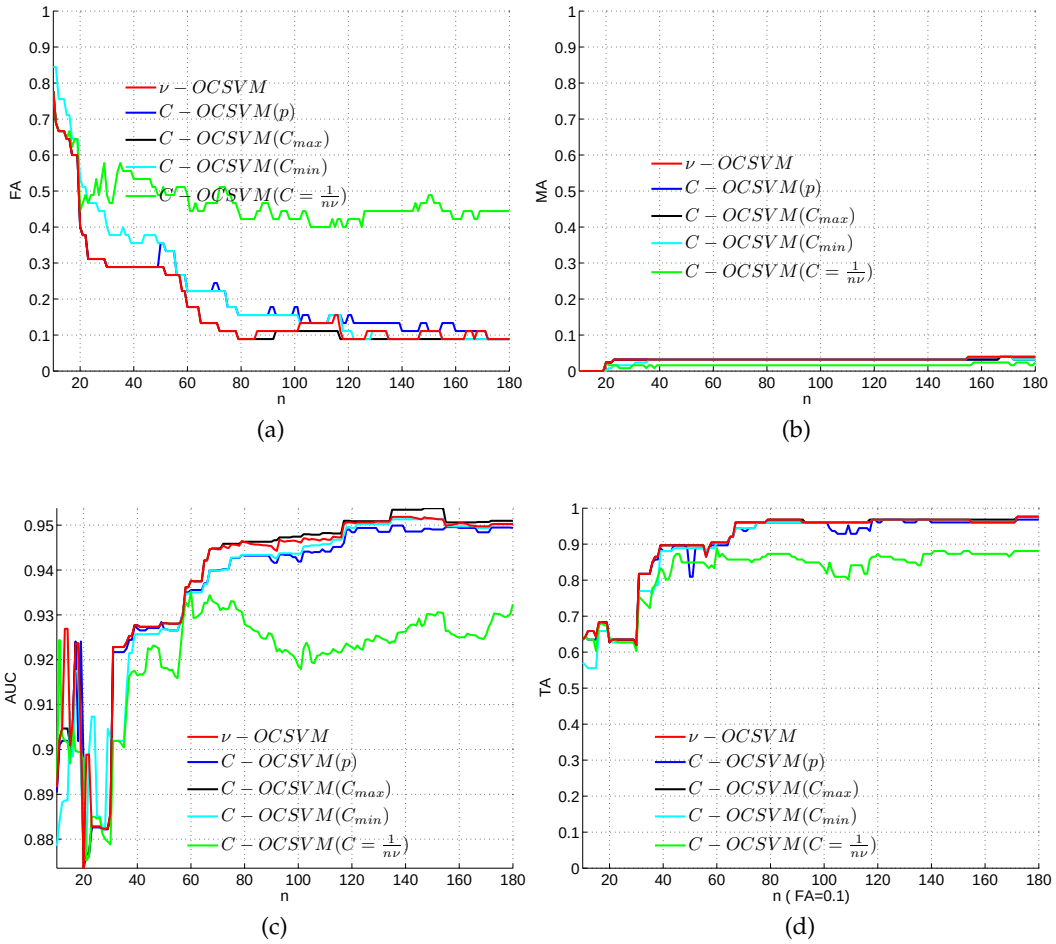


FIGURE 5.5: Ionosphere data (a) false alarm rate, (b) miss alarm rate, (c) AUC, (d) true alarm rate (FA=0.1).

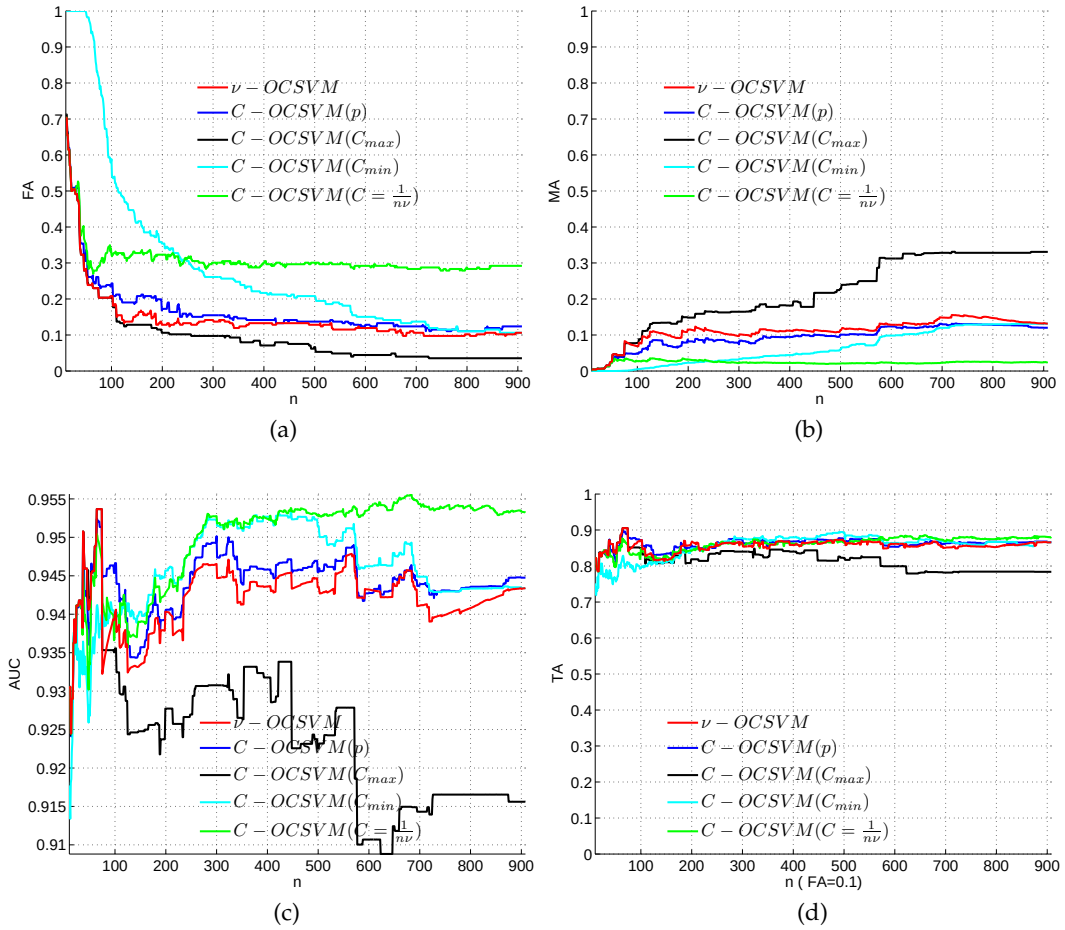


FIGURE 5.6: Hand digits data (a) false alarm rate, (b) miss alarm rate, (c) AUC, (d) true alarm rate (FA=0.1).

5.5 Online multi-task learning

As the multi-task learning approach proposed in chapter 3 can be solved by one class SVM, based on the online learning techniques in this chapter, we can develop an online transition interface from the old detection system to the new one when a new detection system is introduced or the existing system experiences a change. A flow chart of the transition procedure is shown in figure 5.7 by using the necessary techniques proposed in this thesis.

Here we give an example of this online learning procedure (MTL with online ν-OCSVM) to the banana dataset (figure 3.2) in chapter 3. We use 400 source data $\boldsymbol{X}_1$, and the target data $\boldsymbol{X}_2$ increase from 10 to 400 one by one, with $\pi/12$ rotation and $(-0.3, 0.5)$ translation from the source data. Results are averaged by 10 times and are shown in figure 5.8. Similar to the offline results, the false alarm rate decreases

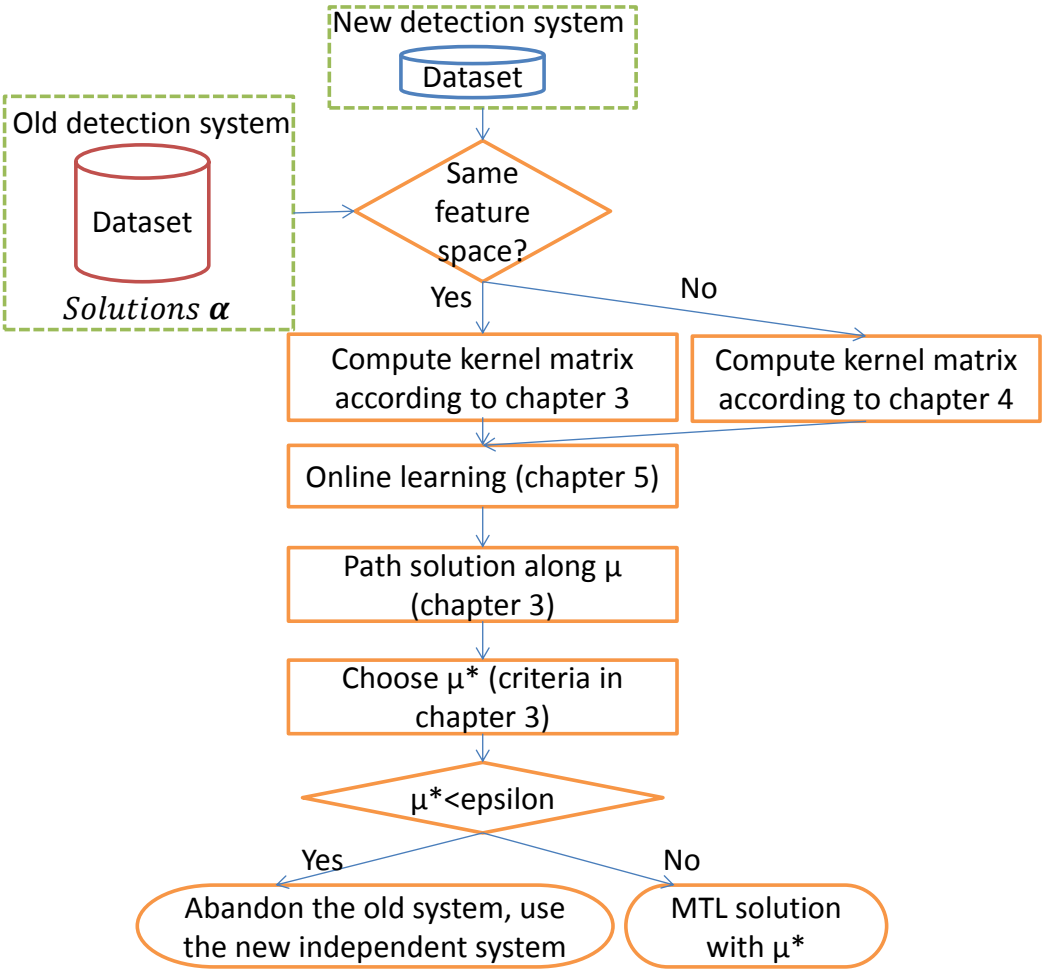


FIGURE 5.7: A flow chart of the transition procedure from an old detection system to the new one (when the chosen μ^* is small enough, we can drop out the old system, and just use the new detection system independently).

to a constant value when n_2 is larger than 150, and at the same time the miss alarm reaches a constant value to the problem itself.

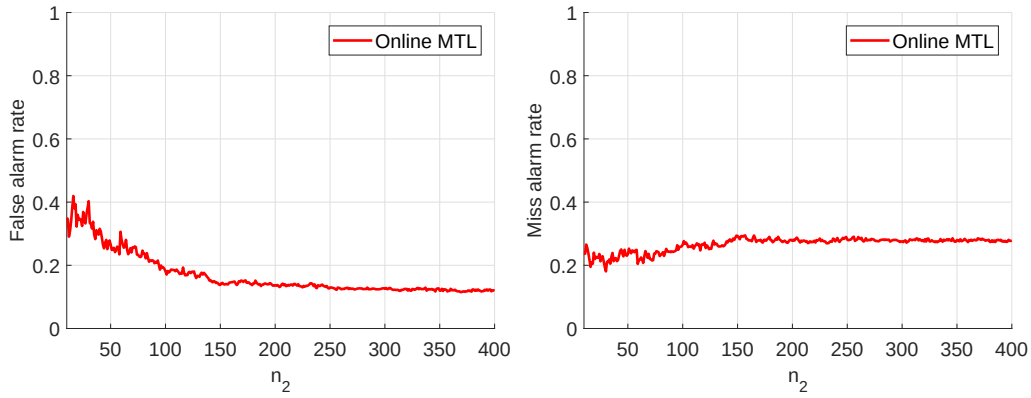


FIGURE 5.8: Online learning on banana dataset: (a) false alarm rate, (b) miss alarm rate.

Discussions: it is possible to develop a ν-OCSVM version multi-task learning parameterized by μ, which can be solved by classique one-class SVM and the solution can move from one union task learning to two independent tasks learning?

- As we know for ν-OCSVM, the parameter ν gives an upper bound on the fraction of outliers, it is not possible to use just one ν to bound the fraction of outliers for source task and target task at the same time under the condition that the solution (controlled by μ) can move from one union task solution to two independent tasks solution.
- Although C version of OCSVM is less interpretive to the fraction of outliers, but the proposed criteria in chapter 3 to choose the optimal μ for MTL has one step which is to choose the solution with desired false alarm rate, to some extend it overcomes this disadvantage.

5.6 Chapter summary

In this chapter, we studied the online learning problem for one-class SVM in the purpose of getting a constant false alarm rate.

For C-OCSVM, two steps are needed to achieve this purpose, one is the online learning and the other step is the parameter adaptation. For ν-OCSVM, it is more straightforward and we proposed the online version of ν-OCSVM learning. These two methods are equal if the same ν is adopted except that the method of online C-OCSVM learning with C adaptation is more complicated and not as direct as ν-OCSVM.

We compare the experiments using four different methods: online learning with fixed C and fixed ν, online learning with C adaptation according to training error p and according to the approximation $C = \frac{1}{n\nu}$. The results show that the ν-OCSVM (equivalent to C-OCSVM with C adaptation according to ν) and C-OCSVM($C(p)$) outperform the other possible methods (with fixed C) and they can keep the test error of false alarm rate and miss alarm rate relatively stable as n increases.

As the multi-task learning model in chapter 3 and chapter 4 can be solved by classical one-class SVM, the proposed method in this chapter to get a constant false alarm rate for online one-class SVM learning can also be applied to the multi-task learning cases directly.

6 Conclusions and Future Works

Contents

6.1 Conclusions

In this thesis, we studied the transfer learning for one-class SVM problem defined in chapter 1. Generally, we divide the research problem into two part, one part is the transfer learning in homogenous space and the other part is the transfer learning in heterogeneous space. We proposed a multi-task learning model to solve this problem. After that, we studied the online learning problem of the proposed model.

(1) From the homogenous side, we consider the situation that the detection system may experience a distribution change for the dataset. Practical motivations are

the lack of data samples for new introduced system or sensor update of an existed detection system. In order to cope with this problem, transfer learning techniques can be used to transfer knowledge from related trained model to the new one in the hope of without significant performance loss.

We proposed a multi-task learning model in chapter 3 to leverage the information needed from the old detection system to the new one. The key feature of the new model is the introduction of a parameter μ to control how much we rely on the former model. This parameter has to be set and changed as the amount of new data coming from the system increases. We define the new detection model as a classical one class SVM with a specific kernel matrix which depends on the introduced parameter. A kernel adaptation method for C-one class SVM is developed in order to get the path solution along that parameter and criteria are established to select a good value.

By using the method, it selects μ which enables to transfer as much information as needed from the old task to the new one to keep approximately stable performances even if the amount of new data is limited. When enough new samples are collected, the model tends to choose small μ which indicates that the former solution and data are not useful anymore. Potential negative transfer learning problem can be avoided and experimental results show that smooth transition from the old detection system to the new one can be obtained.

(2) From the heterogeneous side, we consider the situation of the introduction of new measurements, such as new added sensors to an existed system. We applied the multi-task learning model to this case when new features are added in chapter 4. We proposed two approaches to tackle such problem.

One approach is to ignore the new feature when computing the cross matrix K_{st} between source dataset and target dataset, without changing the internal matrix K_{ss} for source dataset, K_{tt} for target dataset. It gives a certain balance from the old detection system to the new one but produces non positive definite matrix for the optimization problem.

The other approach is to fill the potential new feature in the old detection system.

Typical estimation methods can be used, However, when the number of samples for the new detection system is small, no matter what kind of method is used, it is hard to get a good estimation. As result, a variable kernel is used to balance the importance of the new feature with the number of new samples and at the end it converges to the same value as used in the old detection system. Experiments show that the second approach outperforms the first one, and it gives a good transition from the old detection system to the new one.

(3) As the proposed multi-task learning model can be solved by classical one-class SVM, the online learning techniques for the standard SVM can be applied to multi-task learning directly. In order to get a relatively stable performance during the online learning procedure, we studied the constant false alarm rate problem for one-class SVM in chapter 5.

For parameter C based one-class SVM, one drawback is that it encounters a decrease of false alarm rate and an increase of miss alarm rate during the online learning procedure as the number of samples increases. We first proposed to adapt the parameter C according to the proportion of outliers or according to the parameter ν of ν-OCSVM which gives an upper bound on the fraction of outliers. This method needs two steps, one step is the online learning and the other step is the constraint parameter adaptation, which is kind of complex and not straightforward. For ν-OCSVM, as the parameter ν gives an upper bound on the fraction of outliers, we developed an online version of ν-OCSVM which is more directly. Experiments show that the proposed methods are more stable than the online learning method with fixed C.

6.2 Future works

Possible future works include:

(1) Other transfer learning method, like domain adaptation can also be investigated for one-class SVM. But domain adaptation works well usually when the amount of target data and source data is not very small. In our case, we want to keep

the detection performance since the system is maintained or since new sensor is added and we want to have a smooth transition from the old system to the new one.

(2) As different number of samples may have different kernel density, a variable of kernel parameter may also be used to improve the online learning algorithm for one-class SVM. Accordingly, we can use the proposed kernel path method to move solution from one kernel parameter to another one directly.

A Résumé en Français

Contents

L'objectif de cette thèse est de minimiser la perte de performance du système de détection mono-classe (ou 1-classe) lorsqu'il rencontre un changement de distribution de données. L'idée est d'utiliser l'approche d'apprentissage par transfert pour partager les informations apprises dans l'ancien système vers le nouveau. Pour atteindre notre objectif, nous avons divisé ce problème d'apprentissage du transfert en deux parties : La première partie est consacrée à l'apprentissage du transfert dans l'espace des caractéristiques homogènes et la seconde partie est effectuée dans l'espace des caractéristiques hétérogènes.

A.1 Introduction

Dans le chapitre 1, nous présenterons la problématique de recherche de cette thèse. D'abord nous définirons ce qu'est la classification 1-classe, puis nous définirons également ce qu'est l'apprentissage par transfert et enfin nous présenterons le problème de recherche de cette thèse: le transfert d'apprentissage pour la classification mono-classe.

A.1.1 Formulation du problème

A.1.1.1 classification 1-classe

Dans de nombreuses applications, il est nécessaire ou même obligatoire, dans certaines conditions, de mettre en place des systèmes de surveillance pour superviser le comportement du dispositif opérationnel. Notamment, nous devrions être en mesure de savoir si un système est en bon état et fonctionne normalement comme nous le souhaitons. En cas de présence de comportements anormaux nous avons également besoin d'être alertés. Par exemple, dans le cas de fraude, nous avons besoin d'un système de détection pour assurer la sécurité du paiement en ligne, c'est-à-dire pour chaque transaction, le système de détection doit être en mesure de distinguer une situation considérée comme "légale" pour soutenir la transaction ou "illégale" pour prévenir le comportement frauduleux. Généralement, nous pouvons définir ces comportements comme inliers (événements normaux) ou aberrants (événements anormaux).

Le système basé sur ces observations pour détecter les valeurs aberrantes potentielles est appelé système de détection des valeurs aberrantes, qui est également appelé système de détection de nouveauté ou de détection d'anomalies dans l'état de l'art [Khan and Madden, 2014; Pimentel et al., 2014; Tarassenko et al., 2009].

La **classification 1-classe** tente d'apprendre un classificateur basé uniquement sur les échantillons de classes normales (classe cible) pour identifier les nouveaux échantillons appartenant à cette classe et les distinguées de valeurs aberrantes.

A.1.1.2 Transfert d'apprentissage

Une hypothèse implicite de la plupart des modèles d'apprentissage automatique est que les données d'apprentissage et les données de validation ou d'essai sont issues de la même distribution. Cependant, ceci n'est pas toujours vrai dans les applications réelles. Par exemple, on peut imaginer le cas d'un système de détection d'anomalies au sein d'une centrale nucléaire qui est appris sur la base des données de l'usine A, ne fonctionnera peut-être pas bien sur une nouvelle usine B

parce que des nouvelles mises à jour techniques peuvent être introduites et les données d'essai peuvent être distribuées différemment. Afin d'obtenir un modèle de confiance, lorsque la distribution change, la plupart des modèles d'apprentissage automatique doivent être reconstruits à nouveau en utilisant des nouvelles données collectées [Cao, Zhang, and Yang, 2011]. Procédure ainsi peut s'avérer coûteux et difficile pour re-collecter suffisamment de nouvelles données pour construire un modèle. Inspiré par les activités d'apprentissage humain, l'apprentissage par transfert est apparu pour donner des éléments de réponses, pour tenter de résoudre ce genres de problème.

L'apprentissage par transfert utilise les informations apprises par la tâche source T_S, avec des données source $\boldsymbol{X}_S = \{\boldsymbol{x}_{S1}, ..., \boldsymbol{x}_{Sn}\}$ qui proviennent de l'espace des caractéristiques de la source $\mathcal{X}_S$, pour aider à l'apprentissage de la tâche cible T_T avec les données cibles $\boldsymbol{X}_T = \{\boldsymbol{x}_{T1}, ..., \boldsymbol{x}_{Tn}\}$ de l'espace des caractéristiques cible $\mathcal{X}_T$.

A.1.1.3 Problème de recherche: transfert d'apprentissage pour la classification 1-classe

Dans le cadre de notre travail de recherche, nous considérons les problèmes d'apprentissage de transfert suivants:

(1) Transfert d'apprentissage pour la classification mono-classe dans un espace à caractéristiques homogènes où les données source et cible proviennent du même espace caractéristique $\mathcal{X}_S = \mathcal{X}_T \subseteq \mathbb{R}^d$, mais suivent des distributions différentes $P(\boldsymbol{X}_S) \neq P(\boldsymbol{X}_T)$. Nous avons identifié deux types d'exemples correspondants aux situations suivantes:

   a) Lorsque nous devons introduire un nouveau système de détection, mais nous ne voulons pas attendre longtemps pour avoir suffisamment de données régulières pour apprendre un nouveau modèle, dans ce cas, nous pouvons utiliser l'information disponible dans un autre modèle d'apprentissage connexe pour transférer les connaissances au nouveau modèle.

b) Lorsque nous devons changer ou mettre à jour une partie des capteurs d'un système de détection quelconque en vue d'obtenir un système à jour, les nouveaux échantillons de données dans ce cas devraient probablement suivre une distribution différente. Il faudrait alors adapter la nouvelle distribution des données relative au nouveau système à l'ancien comportement du système de détection tout en préservant les performances.

(2) Transfert d'apprentissage pour la classification mono-classe dans les espaces caractéristiques hétérogènes. Un exemple de cette situation est rencontré lorsque nous voulons ajouter de nouveaux capteurs au système de détection en fonction des suggestions de l'expert du domaine, les données source et cible proviennent alors de différents espaces $\mathcal{X}_S \subseteq \mathbb{X}^d, \mathcal{X}_T \subseteq \mathbb{R}^q$, où $d \neq q$.

A.1.2 Contributions de la thèse

Les principales contributions de cette thèse sont énumérées dans les points suivants:

(1) Un modèle d'apprentissage multi-tâche pour les SVMs mono-classe :

Ce modèle utilise un paramètre $\mu \in [0, 1]$ pour pondérer les informations provenant de l'ancienne et de la nouvelle tache. Ce paramètre permet d'aller d'un modèle unique à plusieurs modèles indépendant. Le problème ainsi formulé peut être résolu par les modèles de SVM mono-classe classiques avec une matrice de noyau spécifique. Lorsqu'il s'applique au cas de transfert d'apprentissage, ce modèle peut tirer parti des informations de la tâche initiale conjointement avec la nouvelle tâche en ajustant correctement le paramètre μ. Il permet une transition en douceur du système de détection original vers le nouveau sans perte de performance significative.

(2) Une méthode d'adaptation au noyau variable:

Comme la matrice de noyau pour le problème dual du modèle proposé est paramétrée par μ, cette méthode permet d'obtenir la solution complète du problème, ce qui signifie que nous pouvons balayer l'espace de solution pour toutes les valeurs de μ.

(3) Un critère de réglage des paramètres pour l'apprentissage multi-tâches :

Comme l'apprentissage du transfert est axé sur la performance de la tâche cible, lorsque nous appliquons ce modèle de transfert d'apprentissage multi-tâches, les critères de réglage des paramètres doivent garantir un bon équilibre entre l'information provenant du système source et celles venant du système cible. Cette équilibre évolue au fil du temp lorsque la quantité d'information des système cible acquise augmente.

(4) Application du modèle d'apprentissage multi-tâches dans le cas où des nouvelles caractéristiques sont ajoutées:

Pour des raisons pratiques, lorsque de nouvelles caractéristiques sont ajoutées à l'ancien système de détection, en utilisant une variation du paramètre de noyau Gaussien, une transition en douceur peut être obtenue de l'ancien système de détection vers le nouveau en supposant que le nombre d'échantillons provenant du nouveau système augmente.

(5) Un algorithme avec des taux de fausses alarmes constants pour les SVM 1-classe :

Comme le modèle proposé peut être résolu à l'aide d'un algorithme SVM 1-class classique, les problèmes d'apprentissage dynamiques de SVM 1-classe sont étudiés et un algorithme avec taux de fausse alarme constant est développé.

A.2 Etat de l'art

Dans le chapitre 2, nous examinerons les travaux connexes portant sur les algorithmes typiques pour SVM 1-classe ainsi que la littérature connexe sur l'apprentissage multi-tâches et l'apprentissage par transfert pour les SVMs 1-classe .

A.2.1 SVM 1-classe

Il existe deux principaux types de SVM 1-classe pour la classification ou la détection des valeurs aberrantes. L'un est proposé par [Schölkopf et al., 1999, 2001], connu sous le nom ν one-classe SVM (ν-OCSVM). Il consiste à trouver un hyperplan optimal dans l'espace caractéristique pour séparer la proportion majoritaire des échantillons de données de l'origine. La proportion de valeurs aberrantes est contrôlée par un paramètre ν qui donne une limite supérieure sur la fraction des valeurs aberrantes et une limite inférieure des fractions de support vecteurs pour l'apprentissage des données.

L'autre est introduit par [Tax and Duin, 1999a,b, 2004], nommé support vector domain description (SVDD), qui vise à trouver une hypersphère avec un volume minimal pour inclure les échantillons de données dans l'espace des caractéristiques. Dans ce cas, la quantité de données dans le l'hypersphère est réglée par un paramètre C.

Selon [Chang and Lin, 2001; Schölkopf et al., 2001; Tax, 2001], il est prouvé que ces deux approches conduisent à la même solution dans une condition de construction que satisfait le noyau Gaussian. Dans ce cas, une relation peut être trouvée entre les paramètres ν et C.

A.2.2 Apprentissage multi-tâche et apprentissage par transfert

L'apprentissage multi-tâches (MTL) [Zhang and Yang, 2017] et l'apprentissage par transfert (TL) [Pan and Yang, 2010; Weiss, Khoshgoftaar, and Wang, 2016] sont de bons moyens pour résoudre le problème de pénurie de données. Les deux sont inspirés par les activités d'apprentissage humain où de nouvelles compétences peuvent être apprises plus efficacement grâce à des expériences ou des compétences connexes.

La différence entre les deux est que l'apprentissage multi-tâche est axé sur l'amélioration simultanée des multiples tâches associées, tandis que l'apprentissage par transfert se préoccupe uniquement de la performance de la tâche cible en utilisant les tâches sources comme tâches auxiliaires.

A.2.2.1 Apprentissage multi-tâche

Différents modèles d'apprentissage multi-tâche sont proposés dans la littérature. Par exemple, [Caruana, 1997] suppose que plusieurs tâches associées partagent des unités cachées liées dans des réseaux neuronaux artificiels; pour les SVM à deux classes, [Evgeniou and Pontil, 2004] suppose que l'hyperplane séparateur de la fonction de décision associé à 1 tâche est construit à l'aide d'une partie commune à toute les tâches et d'une partie spécifique à la tâche considère. Cependant, le domaine de la classification mono-classe est peu couvert. Suivant la même idée de [Evgeniou and Pontil, 2004], [He et al., 2011, 2014] a proposé un algorithme d'apprentissage multi-tâche pour SVM mono-classe. La méthode proposée dans le chapitre 3 est dérivée de celle-ci, nous présentons cette approche en détail.

Considérons le cas de l'apprentissage des tâches T dans le même espace de fonctions $\mathcal{X}$, où $\mathcal{X} \in \mathbb{R}^d$. Nous avons n_t échantillons pour chaque tâche t, $\boldsymbol{X}_t = \{\boldsymbol{x}_{1t}, \boldsymbol{x}_{2t}, ..., \boldsymbol{x}_{n_t t}\} \in \mathcal{X}$. Pour la tâche t, l'un hyperplane définissant le domaine de la classe dans l'espace transformé peut être défini comme suit:

$$f_t(\boldsymbol{x}) = sign(\langle \boldsymbol{w}_t, \phi(\boldsymbol{x}) \rangle - \rho_t), \tag{A.1}$$

où $\boldsymbol{w}_t$ est un vecteur orthogonal à l'hyperplan et ρ_t est un décalage. En utilisant l'hypothèse que le vecteur paramétré $\boldsymbol{w}_t$ peut être divisé en un vecteur moyen $\boldsymbol{w}_0$ et un vecteur spécifique $\boldsymbol{v}_t$ pour chaque tâche; nous pouvons écrire:

$$\boldsymbol{w}_t = \boldsymbol{w}_0 + \boldsymbol{v}_t \tag{A.2}$$

Ensuite, le problème initial pour l'apprentissage multi-tâche peut être formulé comme suit:

$$\begin{gathered} \min_{\boldsymbol{w}_0, \boldsymbol{v}_t, \xi_{it}, \rho_t} \frac{1}{2} \sum_{t=1}^{T} \|\boldsymbol{v}_t\|^2 + \frac{\mu}{2} \|\boldsymbol{w}_0\|^2 + \sum_{t=1}^{T} \sum_{i=1}^{n_t} \frac{\xi_{it}}{n_t \nu_t} - \sum_{t=1}^{T} \rho_t \\ s.t. \ \langle (\boldsymbol{w}_0 + \boldsymbol{v}_t), \phi(\boldsymbol{x}_{it}) \rangle \geq \rho_t - \xi_{it}, \ \xi_{it} \geq 0, \end{gathered} \tag{A.3}$$

pour $t \in \{1, 2, ..., T\}$, $i \in \{1, 2, ..., n_t\}$, ξ_{it} est variable pour échantillonner $\boldsymbol{x}_{it}$ et le paramètre de régularisation μ dont le but est de contrôler la similarité des tâches.

En introduisant les multiplicateurs $\alpha_{it}, \beta_{it} \geq 0$, le problème dual est formulé comme suit:

$$\begin{aligned} \min_{\alpha_{it}} \sum_{t=1}^{T} \sum_{r=1}^{T} \sum_{i=1}^{n_t} \sum_{j=1}^{n_r} \alpha_{it} \alpha_{jr} \left(\frac{1}{\mu} + \delta_{rt} \right) k(\boldsymbol{x}_{it}, \boldsymbol{x}_{jr}) \\ s.t.\ 0 \leq \alpha_{it} \leq \frac{1}{n_t \nu_t}, \sum_{i=1}^{n_t} \alpha_{it} = 1, \end{aligned} \tag{A.4}$$

où $k(\boldsymbol{x}_{it}, \boldsymbol{x}_{jr}) = \langle \phi(\boldsymbol{x}_{it}), \phi(\boldsymbol{x}_{jr}) \rangle$ et δ_{rt} est défini comme:

$$\delta_{rt} = \begin{cases} 1, & \text{si } r = t, \\ 0, & \text{sinon.} \end{cases} \tag{A.5}$$

L'intention de [He et al., 2014] est de résoudre le problème (A.4) en utilisant l'approche standard d'optimisation SVM à une classe. Cependant, le problème (A.4) ne peut pas être résolu au moyen de SVM standard d'une classe même si nous avons les mêmes ν et n pour chaque tâche car la contrainte d'égalité $\sum_{i=1}^{n_t} \alpha_{it} = 1$ pour MTL est la somme des multiplicateurs lagrangiens sur la tâche t. Même si nous avons résolu le problème (A.4), la solution de MTL ne sera jamais identique à celle de 1-classe SVM indépendant, quelle que soit la valeur de μ. La preuve en est que quand μ est assez petit, le terme d'optimisation de (A.4) peut être considéré comme le même que celui de 1-classe SVM, mais la contrainte pour 1-classe SVM indépendant vérifie: $\sum_{t=1}^{T} \sum_{i=1}^{n_t} \alpha_{it} = 1$ alors que la contrainte ici pour MTL satisfait: $\sum_{i=1}^{n_t} \alpha_{it} = 1$ pour chaque t.

A.2.2.2 Transfert d'apprentissage

Un nombre croissant de recherches dans la littérature de transfert d'apprentissage est apparu au cours de la dernière décennie. Ces approches d'apprentissage de transfert peuvent être classées selon différentes caractéristique.

(1) Basé sur la disponibilité des étiquettes, l'apprentissage par transfert peut être catégorisé comme apprentissage par transfert inductif, apprentissage par transfert transductif et apprentissage par transfert non supervisé [Pan and Yang, 2010].

(2) Basé sur la différence au niveau de la distribution de probabilité, la méthode principalement proposée est connue comme l'adaptation entre les domaines [Daume III and Marcu, 2006; Kulis, Saenko, and Darrell, 2011; Long et al., 2014] pour corriger la distorsion entre les distributions de données..

(3) Basé sur la différence entre les espaces d'attributs d'entrée, l'apprentissage de transfert peut être classé comme apprentissage homogène et apprentissage hétérogène.

(4) Sur la base des connaissances à transférer, quelques sous-catégories peuvent être définies, telles que l'apprentissage par transfert d'instance, l'apprentissage par transfert de paramètres, l'apprentissage par transfert de fonctionnalités pour n'en citer que quelques uns [Pan and Yang, 2010; Weiss, Khoshgoftaar, and Wang, 2016].

Notamment, la thématique d'apprentissage non supervisée pour l'apprentissage par transfert est peu étudiée dans la littérature, notamment dans le cas où le domaine source et le domaine cible ont un nombre différent d'espaces de caractéristiques.

A.2.3 Conclusions

A partir de l'analyse ci-dessus, la classification mono-classe (détection des valeurs aberrantes) apparaît comme un apprentissage non supervisé, car les valeurs aberrantes ou la détection d'événements anormaux ne sont effectués qu'à partir des données normales pendant le processus d'apprentissage. Nous pouvons intégrer l'apprentissage par transfert des SVM 1-classe dans l'apprentissage non supervisé, qui est un domaine peu étudié dans les littératures d'apprentissage par transfert.

La différence majeure entre l'apprentissage par transfert et l'apprentissage multi-tâche est lié au fait qu'ils ont différent objectif pour l'apprentissage. L'apprentissage par transfert est axé sur les résultats de l'apprentissage des tâches cibles alors que l'apprentissage multi-tâches prend toutes les tâches d'apprentissage comme ayant la même importance. Si nous utilisons l'apprentissage multi-tâche dans le cadre de l'apprentissage par transfert, un problème de transfert négatif potentiel doit être correctement résolu (c'est-à-dire que l'exécution d'un modèle basé uniquement sur

l'ensemble de données cible est meilleur que celui du modèle d'apprentissage par transfert).

A.3 Transfert d'apprentissage dans un espace homogène

Dans le chapitre 3, nous proposerons un nouveau modèle d'apprentissage multi-tâche, afin de résoudre le problème d'apprentissage par transfert pour SVM 1-classe avec un changement de distribution de données dans le même espace caractéristique.

A.3.1 Motivation

Pour des raisons pratiques, nous pouvons rencontrer un changement de distribution de données. Par exemple, la mise à jour ou le changement des capteurs dans un système de détection, l'introduction d'un nouveau système de détection, etc. Dans une telle situation, nous aimerions réduire la perte de performance et éviter d'attendre longtemps pour obtenir suffisamment de nouveaux échantillons pour former un modèle fiable.

L'apprentissage multi-tâches est une approche souvent utilisée pour résoudre le problème ci-dessus. Des recherches antérieures montrent que l'apprentissage simultané de plusieurs tâches connexes entraîne de meilleures performances que l'apprentissage indépendant [Evgeniou and Pontil, 2004; He et al., 2014; Yang, King, and Lyu, 2010]. Par exemple, [Yang, King, and Lyu, 2010] a proposé une structure d'apprentissage multi-tâche pour ν -SVM en limitant la différence de la norme L_2 entre chaque paire de paramètres afin d'avoir des solutions similaires pour les tâches associées. Plus tard, la motivation de [Evgeniou and Pontil, 2004] et [He et al., 2014] a conduit à une autre méthode d'apprentissage multi-tâche pour le SVM à une classe en supposant que les paramètres des modèles ou des modèles de tâches connexes sont proches d'une certaine fonction moyenne.

A.3.2 Description du modèle proposé

Dans la méthode proposée, nous nous sommes inspirés par [He et al., 2014], le vecteur normal $\boldsymbol{w}_t$ pour chaque tâche peut être divisé en deux parties : une partie correspond au vecteur moyen commun $\boldsymbol{w}_0$ partagé entre toutes les tâches d'apprentissage et l'autre partie est le vecteur spécifique $\boldsymbol{v}_t$ pour une tâche t spécifique.

$$\boldsymbol{w}_t = \mu \boldsymbol{w}_0 + (1-\mu)\boldsymbol{v}_t, \tag{A.6}$$

où $\mu \in [0,1]$, quand $\mu = 0$, alors $\boldsymbol{w}_t = \boldsymbol{v}_t$, ce qui correspond à l'apprentissage de tâches séparées par T. Quand $\mu = 1$, alors $\boldsymbol{w}_t = \boldsymbol{w}_0$, ce qui correspond à une seule tâche globale.

Le problème primal est:

$$\begin{aligned} \min_{\boldsymbol{w}_0, \boldsymbol{v}_t, \xi_{it}} & \frac{1}{2}\mu \parallel \boldsymbol{w}_0 \parallel^2 + \frac{1}{2}(1-\mu)\sum_{t=1}^{T} \parallel \boldsymbol{v}_t \parallel^2 + C\sum_{t=1}^{T}\sum_{i=1}^{n_t}\xi_{it} \\ s.t. \ & \langle \mu \boldsymbol{w}_0 + (1-\mu)\boldsymbol{v}_t, \phi(\boldsymbol{x}_{it})\rangle \geq 1 - \xi_{it}, \ \xi_{it} \geq 0 \end{aligned} \tag{A.7}$$

En introduisant les multiplicateurs de Lagrange $\alpha_{it}, \beta_{it} \geq 0$, alors le Lagrangian dual problème est:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} & -\frac{1}{2}\boldsymbol{\alpha}^T K^{\mu} \boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{1} \\ s.t. \ & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \end{aligned} \tag{A.8}$$

où $\boldsymbol{\alpha}^T = [\alpha_{11}, ..., \alpha_{n_1 1}, ..., \alpha_{1T}, ..., \alpha_{n_T T}]$, et K^{μ} est une matrice de blocs avec des blocs $T \times T$ correspondant à toutes les paires de tâches. Soit K^{μ}_{rt} le bloc correspondant à la tâche r et t, définie comme:

$$K^{\mu}_{rt} = (\mu + (1-\mu)\delta_{rt})\langle \phi(\boldsymbol{X}_t), \phi(\boldsymbol{X}_r)\rangle, \tag{A.9}$$

où δ_{rt} est:

$$\delta_{rt} = \begin{cases} 1, & \text{si } r = t, \\ 0, & \text{sinon.} \end{cases} \tag{A.10}$$

Notez que l'équation (A.8) pourrait être résolue par la classe classique SVM (C-OCSVM) avec la matrice de Gram modifiée spécifique K^{μ}. Dans le cas qui nous

intéresse, la fonction de décision pour la tâche cible est:

$$f_2(\boldsymbol{x}) = sign(g_2(\boldsymbol{x}) - 1), \tag{A.11}$$

où la fonction SVM $g_2(\boldsymbol{x})$ est définie comme:

$$g_2(\boldsymbol{x}) = \boldsymbol{\alpha}^T \begin{bmatrix} \mu k(\boldsymbol{X}_1, \boldsymbol{x}) \\ k(\boldsymbol{X}_2, \boldsymbol{x}) \end{bmatrix}. \tag{A.12}$$

A.3.3 Parcours de l'ensemble de solution

Pour résoudre le problème, le paramètre μ doit être choisi afin d'obtenir une matrice de noyau, où

$$K^\mu = (1 - \mu)K^0 + \mu K^1. \tag{A.13}$$

Cependant, nous ne savons pas exactement quel μ est le mieux adapté à la situation rencontrée. De plus, il est fastidieux de rechercher tout l'espace de la solution en résolvant le modèle de SVM 1-classe pour un ensemble significatif de valeurs possibles de μ.

En surveillant le changement de groupe des paramètres Lagrangians, une solution iterative permet de suivre les évolutions de $\boldsymbol{\alpha}^\mu$ pour l'ensemble du domaine de μ de 0 à 1. Cela facilite le choix du paramètre μ dans une situation donnée avec un certain nombre d'échantillons de la tâche cible.

A.3.4 Critères pour le réglage du μ

Intuitivement, une bonne solution pour le modèle proposé devrait avoir les propriétés suivantes:

- Il pourrait s'adapter au changement de la fonction de décision en raison du changement de distribution des données.
- En même temps, il maintient le taux de fausses alarmes aussi proche que possible de celui désiré (généralement, c'est un niveau donné par l'utilisateur qui indique la proportion d'instances rejetées).

Pour sélectionner une bonne valeur de μ, l'application successive de 3 critères sont proposée:

(1) Tout d'abord, déterminer le nombre de Nième maximal de support vecteurs associé à la nouvelle tâche au sein de α^{μ} noté $\max(\#\text{SV}_2)^{Nth}$. Ce critère supprime les solutions qui repose peu sur les nouvelles données et réduit le domaine de valeurs possibles de μ.

(2) Ensuite, réduire ce domaine initial afin que le taux de fausses alarmes soit aussi proche que possible de p (une proportion de valeurs aberrantes désirées).

(3) Enfin, sélectionner parmi ces solutions celle qui préserve au mieux la détection pour la tâche initiale, dont celle qui impacte le moins de SVM 1-classe de la tâche initiale.

Afin de tester l'efficacité de cette approche, nous construisons lors de simulation une fonction de référence à comparer avec le paramètre sélectionné μ par les critères. Les résultats montrent que le détecteur trouvé à l'aide de ce critère est proche de la référence, ce qui montre que le critère proposé pour μ est fiable.

A.3.5 Expériences et résultats

Les expériences sont menées sur des données synthétiques et des données réelles de qualité du vin. Les résultats montrent que la méthode proposée peut adapter le changement de fonction de décision de l'ensemble de données, elle donne une bonne transition de la tâche précédente à la nouvelle tâche qui est basée sur le nouvel ensemble de données alors que n_2 (nombre d'échantillons dans T_2) augmente progressivement.

La méthode pourrait être utilisée pour gérer une transition en douceur entre les deux tâches et pour définir la nouvelle détection en gardant la continuité de service du système de détection.

A.4 Transfert d'apprentissage dans un espace hétérogène

Dans le chapitre 4, nous adaptons le modèle d'apprentissage multi-tâche proposé dans le chapitre 3 à la situation de l'apprentissage par transfert de SVM 1-classe avec de nouveaux attributs supplémentaires. Cela correspond au second problème de recherche que nous avions défini au début.

A.4.1 Motivation

Nous considérons ici la situation correspondant au transfère d'apprentissage pour SVM 1-classe avec de nouvelles caractéristique supplémentaires associés à la tâche cible. Par exemple, dans l'application de la détection de panne pour un système de moteur, il y a quelques capteurs qui ont déjà été utilisés par un système de diagnostic de moteur pendant un temps et chaque capteur obtient quelques données. Maintenant, en raison de besoins techniques ou d'autres besoins pratiques, tels que l'amélioration des performances de détection, de nouveaux capteurs sont ajoutés à ce système. A notre connaissance, ce genre de problème n'a jamais été abordé dans le contexte de la détection en utilisant le SVM mono-classe.

Au lieu de former un nouveau système de détection depuis le début, l'apprentissage multi-tâche semble être un moyen idéal d'adapter l'ancienne détection à un système mis à jour. Il utilise l'hypothèse qui est satisfaite dans notre contexte que les tâches liées partagent une structure commune ou des paramètres de modèle similaires [Evgeniou and Pontil, 2004]. Ici, nous supposons qu'une tâche est l'ancien système et la seconde est le système mis à jour.

A.4.2 Description du modèle proposé

Définissons l'ensemble de données $\boldsymbol{X}_1 = \{\boldsymbol{x}_{11}, \boldsymbol{x}_{21}, ..., \boldsymbol{x}_{n_1 1}\}$ de l'ancien système de détection T_1 en tant que jeu de données source, où n_1 est le nombre d'échantillons, et $\boldsymbol{x}_{j1} \in \mathbb{R}^d$, $\mathbb{R}^d$ est l'espace caractéristique des sources. Définissons également le jeu de données $\boldsymbol{X}_2 = \{\boldsymbol{x}_{12}, \boldsymbol{x}_{22}, ..., \boldsymbol{x}_{n_2 2}\}$ du système mis à jour T_2 comme jeu de données

cible, où n_2 est le nombre d'échantillons de données cibles, et $\boldsymbol{x}_{j2} \in \mathbb{R}^{d+1}$, $\mathbb{R}^{d+1}$ est l'espace objet cible.

Notez que la matrice de noyau pour le modèle proposé est:

$$K^{\mu} = \begin{bmatrix} K_{ss} & \mu K_{st} \\ \mu K_{st}^T & K_{tt} \end{bmatrix} \tag{A.14}$$

où $K_{ss} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_1) \rangle$, $K_{st} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_2) \rangle$, $K_{tt} = \langle \phi(\boldsymbol{X}_2), \phi(\boldsymbol{X}_2) \rangle$, dans le cas où l'on applique ce modèle à la situation où $\boldsymbol{X}_1$ et $\boldsymbol{X}_2$ proviennent d'un espace de caractéristiques différent. Le problème se produit lorsque nous calculons la matrice K_{st} en raison des différentes caractéristiques entre la tâche source et la tâche cible. Nous avons étudié deux façons différentes de traiter ce problème.

(1) La premières est d'ignorer la nouvelle fonctionnalité uniquement pour le calcul de K_{st}. Alors

$$K_{st} = \langle \phi(\boldsymbol{X}_1), \phi(\boldsymbol{X}_2') \rangle, \tag{A.15}$$

où $\boldsymbol{X}_2' = \{\boldsymbol{x} \backslash x^{(d+1)}\}$ provenant de $\boldsymbol{X}_2$ restreint aux d premières fonctionnalités. Dans une certaine mesure, il donne un équilibre entre l'ancien système de détection et le nouveau système en ajustant le paramètre μ lorsque que n_2 augmente. Cependant, en utilisant cette méthode (appelée MTL_I), la matrice de noyau modifiée n'est pas toujours semi-définie positive, ce qui signifie qu'une solution d'optimisation globale ne peut pas être garantie avec une approche standard.

(2) Une autre façon est de completer la nouvelle mesure $\tilde{x}^{(d+1)}$ manquante pour X_1 en utilisant certaines méthodes d'estimation, par exemple, le plus proche voisin, l'imputation, etc. L'inconvénient de cette méthode est que lorsque le nombre d'échantillons $\boldsymbol{X}_2$ pour la tâche cible est petit, il est difficile de donner une bonne estimation de du attribut $\tilde{x}^{(d+1)}$ dans $\boldsymbol{X}_1$.

Pour tenir compte de cette difficulté, dans le cas d'un noyau Gaussian,

$$\begin{aligned} k(\boldsymbol{x}_i, \boldsymbol{x}_j) &= exp(\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{-2\sigma^2}) \\ &= \prod_{l=1}^{d+1} exp(\frac{\|\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}\|^2}{-2\sigma^2}), \end{aligned} \tag{A.16}$$

donc la matrice du noyau peut être décomposée en deux parties:

$$\begin{aligned} K^{\mu} &= \begin{bmatrix} K_{ss} & \mu K_{st} \\ \mu K_{st}^T & K_{tt} \end{bmatrix}_{\mathbb{R}^{d+1}} \\ &= \underbrace{\begin{bmatrix} K_{ss} & \mu K_{st} \\ \mu K_{st}^T & K_{tt} \end{bmatrix}_{\mathbb{R}^{d}}}_{A_0} \circ \underbrace{\begin{bmatrix} \tilde{K}_{ss} & \tilde{K}_{st} \\ \tilde{K}_{st}^T & K_{tt} \end{bmatrix}_{\mathbb{R}^{1}}}_{A_1}, \end{aligned} \tag{A.17}$$

où $\circ$ est un produit par élément. A_0 est une matrice de noyau basée sur $\mathbb{R}^d$ avec les premières dth attributs. A_1 est la matrice de noyau basé sur $\mathbb{R}^1$ espace avec le $d+1$th attribut estimée $\tilde{x}^{(d+1)}$ de $\boldsymbol{X}_1$ et $x^{(d+1)}$ de $\boldsymbol{X}_2$.

Afin d'équilibrer l'influence du nouvel attribut, on peut choisir un σ_0 fixe pour le sous-espace $\mathbb{R}^d$ et un $\sigma(n)$ variable pour le nouvel attribut.

Pour choisir $\sigma(n)$, nous nous sommes inspirés de l'estimateur de la largeur de fenêtre optimale pour une distribution standard pour l'estimation non paramétrique d'une densité de probabilité [Silverman, 1986]. Cette largeur optimale est données par:

$$h_{opt} = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} n^{-\frac{1}{d+4}}, \tag{A.18}$$

où d est le nombre de dimensions et n est le nombre d'échantillons.

Afin de faire converger $\sigma(n)$ vers la même valeur que σ_0, nous pouvons ajouter des scalaires à h_{opt}. Par conséquent, la fonction de paramètre noyau pour A_1 peut être définie comme:

$$\sigma(n) = c_2 \exp(\frac{c_1}{\sqrt[3]{n}}) h_{opt}, \tag{A.19}$$

où la fonction exponentielle $\exp(\frac{c_1}{\sqrt[3]{n}})$ est décroissante, allant d'une grande valeur

quand n est petite à une petit valeur proche de 1 quand n est grand, ce qui signifie que nous multiplions h_{opt} par un grand nombre au début et nous gardons presque h_{opt} quand n est assez grand. Les constantes c_1, c_2 sont des facteurs d'échelle qui font que $\sigma(n)$ converge vers σ_0 quand n est grand. Nous nommons cette méthode MTL_{II}.

A.4.3 Expériences et résultats

Les expériences sont conçues pour tester la performance du nouveau détecteur selon que les valeurs aberrantes proviennent du nouvel attribut ou des anciens attributs séparément dans un premier temps puis de façon globale.

Les résultats expérimentaux montrent que MTL_{I} et MTL_{II} donnent tous les deux une transition de l'ancien système de détection vers le nouveau système modifié. Leurs performances sont meilleures que d'autres solutions possibles lorsque le système de détection subit un changement d'espace de representation. De plus, MTL_{II} surpasse MTL_{I}, et donne une bonne transition de l'ancien système vers le nouveau avec des attributs supplémentaires.

A.5 Taux constant de fausses alarmes pour l'apprentissage en ligne

Comme nous pouvons résoudre le modèle d'apprentissage multi-tâche proposé SVM par 1-classe classiques (C-OCSVM), nous étudierons au chapitre 5 le problème d'apprentissage en ligne pour le SVM 1-classe, qui peut être utilisé directement avec le modèle développé précédemment.

A.5.1 Motivation

Dans les applications réelles, les données de séries temporelles sont généralement rencontrées plutôt que les données en mode de traitement par lots.

La motivation de ce chapitre est de développer un algorithme de SVM 1-classe en ligne avec une performance stable quand de nouveaux échantillons sont ajoutés. Selon Neyman-Pearson Lemma, le test d'hypothèse le plus puissant est celui avec une erreur de type II minimum en fonction d'un niveau d'erreur de type I (niveau de signification) [Tong, Feng, and Zhao, 2016]. Pour le SVM 1-classe, nous voulons maintenir un taux de fausses alarmes relativement stable sans augmenter trop le taux d'alarmes manquées.

A.5.2 C-OCSVM en ligne et adaptation du paramètre de regularization

Des algorithmes types d'apprentissage en ligne sont proposés pour les SVM à deux classes [Cauwenberghs and Poggio, 2000; Diehl and Cauwenberghs, 2003; Laskov et al., 2006; Tax and Laskov, 2003] et peuvent être appliqués aux SVM à une classe [Tax and Laskov, 2003]. Pour les SVM à une classe, le principal problème de la méthode d'apprentissage en ligne ci-dessus est qu'ils n'adaptent pas le paramètre de contrainte C en fonction du nombre d'échantillons d'apprentissage. Ce qui se passe, c'est que le taux de fausses alarmes diminuera et que le taux d'alarmes manquées augmentera progressivement au fur et à mesure que le nombre d'échantillons n augmente.

Pour résoudre ce problème, nous pouvons soit adapter le paramètre C en fonction de la proportion de valeurs aberrantes ou en fonction du paramètre ν en utilisant la relation équivalente entre C-OCSVM et ν-OCSVM, comme ν donne une limite supérieure à la proportion de valeurs aberrantes. Dans cette section, nous avons proposé des méthodes correspondant à ces 2 options pour résoudre ce problème en deux étapes, une étape d'apprentissage en ligne suivre d'une étape d'adaptation des paramètres.

A.5.3 ν-OCSVM en ligne

Une autre façon est de développer une version en ligne de ν-OCSVM comme le paramètre ν donne une limite supérieure sur la fraction des valeurs aberrantes, ce qui est montré dans l'équation (2.14). En surveillant le changement de groupe des

multiplicateurs de Lagrange , un ν-OCSVM est développé dans cette section. Cette méthode est plus directe par rapport à la C-OCSVM car nous n'avons pas besoin d'adapter le paramètre, une seule étape de traitement suffit.

A.5.4 Expériences et résultats

Nous comparons les expériences en utilisant quatre méthodes différentes: apprentissage en ligne avec C fixe, avec ν fixe, apprentissage en ligne avec adaptation de C selon l'erreur d'apprentissage p et adaptation de C selon l'approximation $C = \frac{1}{n\nu}$. Les résultats montrent que le ν-OCSVM (équivalent à C-OCSVM avec C d'adaptation selon ν) et C-OCSVM ($C(p)$) surpasse les autres méthodes testés (avec C fixe) et ils peuvent maintenir l'estimation du taux de fausse alarme et le taux de non detection relativement stable lorsque n augmente.

Pour C-OCSVM, deux étapes sont nécessaires pour atteindre cet objectif, l'une est l'apprentissage en ligne et l'autre étape est l'adaptation des paramètres. Pour ν-OCSVM, c'est plus simple et nous avons proposé la version en ligne de l'apprentissage ν-OCSVM. Ces deux méthodes sont égales si le même ν est adopté, sauf que la méthode d'apprentissage C-OCSVM en ligne avec adaptation C est plus compliquée et pas aussi directe que ν-OCSVM.

Comme le modèle d'apprentissage multi-tâche proposé peut être résolu par SVM 1-classe classique, la méthode proposée dans ce chapitre pour obtenir un taux de fausses alarmes constant peut également être appliquée directement aux cas d'apprentissage multi-tâche.

A.6 Conclusions et perspectives

A.6.1 Conclusions

Dans cette thèse, nous avons étudié le problème d'apprentissage de transfert pour SVM 1-classe défini dans le chapitre 1: minimiser la perte de performance d'un

système de détection lorsqu'il rencontre un changement de distribution de données à la suite d'un événement connu (maintenance, ajout de capteur etc.). Principalement, nous divisons la résolution du problème en deux parties, une partie est l'apprentissage de transfert dans l'espace homogène et l'autre partie est l'apprentissage de transfert dans l'espace hétérogène. Nous avons proposé un modèle d'apprentissage multi-tâches pour résoudre ce problème. Après cela, nous avons étudié le problème d'apprentissage en ligne du modèle proposé.

(1) Du côté homogène, nous considérons la situation où le système de détection peut subir un changement de distribution pour l'ensemble de données. Les motivations pratiques sont le manque d'échantillons de données pour la mise à jour d'un système ou d'un capteur nouvellement introduit d'un système de détection existant dans un délai raisonnable. Afin de faire face à ce problème, les techniques d'apprentissage du transfert peuvent être utilisées pour transférer les connaissances d'un modèle entraîné apparenté vers le nouveau dans l'objectif de ne pas subir de perte significative de performances.

Nous avons proposé un modèle d'apprentissage multi-tâches dans le chapitre 3 pour tirer parti des informations nécessaires de l'ancien système de détection vers le nouveau. La contribution clé du nouveau modèle est l'introduction d'un paramètre μ pour contrôler à quel point nous dépendons de l'ancien modèle. Ce paramètre doit être défini et modifié lorsque la quantité de nouvelles données provenant du système augmente. Nous définissons le nouveau modèle de détection comme SVM 1-classe classique avec une matrice de noyau spécifique qui dépend du paramètre introduit. Une méthode d'adaptation de noyau pour C-OCSVM est développée afin d'obtenir la solution de chemin le long de ce paramètre et un critère est établi pour sélectionner une bonne valeur.

L'utilisation de la méthode permet de sélectionner μ, ce qui permet de transférer autant d'informations que nécessaire de l'ancienne tâche vers la nouvelle pour conserver des performances stables même si la quantité de nouvelles données est limitée. Lorsque suffisamment de nouveaux échantillons sont collectés, le modèle a tendance à choisir μ petit, ce qui indique que l'ancienne solution et les

données ne sont plus utiles. Un problème potentiel d'apprentissage par transfert négatif peut être ainsi évité et les résultats expérimentaux montrent qu'une transition en douceur de l'ancien système de détection vers le nouveau peut être obtenue.

(2) Du côté hétérogène, nous considérons la situation de l'introduction de nouvelles mesures, telles que de nouveaux capteurs ajoutés à un système existant. Nous avons appliqué le modèle d'apprentissage multi-tâches à ce cas lorsque de nouvelles caractéristiques sont ajoutées dans le chapitre 4. Nous avons proposé deux approches pour résoudre ce problème.

Une approche consiste à ignorer le nouvel attribut lors du calcul de la matrice croisée K_{st} entre l'ensemble de données source et l'ensemble de données cible, sans modifier la matrice interne K_{ss} pour l'ensemble de données source, K_{tt} pour l'ensemble de données cible. Il donne un certain équilibre entre l'ancien système de détection et le nouveau mais produit une matrice définie non positive pour le problème d'optimisation.

L'autre approche consiste à estimer le nouvel attribut pour les données d'apprentissage de l'ancien système de détection. Des méthodes d'estimation typiques peuvent être utilisées. Cependant, lorsque le nombre d'échantillons pour le nouveau système de détection est faible, peu importe le type de méthode utilisé, il est difficile d'obtenir une bonne estimation. Comme résultat, un noyau variable est utilisé pour équilibrer l'importance de la nouvelle fonctionnalité avec le nombre de nouveaux échantillons et à la fin il converge vers la même valeur que celle utilisée dans l'ancien système de détection. Les expériences montrent que la deuxième approche surpasse la première, et qu'elle permet une bonne transition entre l'ancien système de détection vers le nouveau.

(3) Comme le modèle d'apprentissage multi-tâche proposé peut être résolu par un algorithme de résolution de classique SVM à une classe, les techniques d'apprentissage en ligne pour le SVM standard peuvent être appliquées directement à l'apprentissage multi-tâche. Afin d'obtenir une performance relativement stable au cours de la procédure d'apprentissage en ligne, nous avons

étudié le problème du taux de fausses alarmes constant pour SVM 1-classe dans le chapitre 5.

Pour le SVM 1-classe basé sur le paramètre C de régularisation, un inconvénient est qu'il conduit à une diminution du taux de fausses alarmes et une augmentation du taux de non détection pendant la procédure d'apprentissage en ligne lorsque le nombre d'échantillons augmente. Nous avons proposé d'adapter le paramètre C en fonction de la proportion de valeurs aberrantes ou en fonction du paramètre ν de ν-OCSVM qui donne une limite supérieure à la fraction des valeurs aberrantes. Cette méthode nécessite deux étapes, une étape est l'apprentissage en ligne et l'autre étape est l'adaptation des paramètres de contrainte, qui est plutôt complexe et accroît le volume de calcul nécessaire.

Pour le ν-OCSVM, comme le paramètre ν donne une limite supérieure à la fraction des valeurs aberrantes, nous avons développé une version en ligne de ν-OCSVM qui est plus directe. Les expériences montrent que les méthodes proposées sont plus stables que la méthode d'apprentissage en ligne avec C fixe.

A.6.2 Perspectives

Les travaux futurs possibles comprennent:

(1) Développer une interface de transition en ligne entre l'ancien système de détection et le nouveau lorsqu'un nouveau système de détection est introduit ou que le système existant subit un changement. Un organigramme est montré présenté par figure 5.7 en utilisant les techniques nécessaires proposées dans cette thèse.

(2) Comme pour un nombre différent d'échantillons le meilleur paramètre de largeur de bande σ peut changer, une adaptation du paramètre du noyau peut être aussi mise en oeuvre lors du processus d'apprentissage en ligne pour SVM 1-classe. En conséquence, nous pouvons utiliser la méthode d'adaptation du noyau proposé pour déplacer la solution d'un paramètre du noyau à l'autre directement, sans reconstruire la solution complète.

Bibliography

Blitzer, John, Ryan McDonald, and Fernando Pereira (2006). "Domain adaptation with structural correspondence learning". In: *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pp. 120–128.

Cao, Bin, Yu Zhang, and Qiang Yang (2011). "Transfer Learning, Multi-Task Learning, and Cost-Sensitive Learning". In: *Cost-Sensitive Machine Learning*. CRC Press, pp. 61–86.

Caruana, Rich (1997). "Multitask learning". In: *Machine learning* 28.1, pp. 41–75.

Cauwenberghs, Gert and Tomaso Poggio (2000). "Incremental and Decremental Support Vector Machine Learning". In: *Advances in Neural Information Processing Systems* 13, pp. 409–415.

Chang, Chih-Chung and Chih-Jen Lin (2001). "Training v-support vector classifiers: theory and algorithms". In: *Neural computation* 13.9, pp. 2119–2147.

Cortez, Paulo et al. (2009). "Modeling wine preferences by data mining from physicochemical properties". In: *Decision Support Systems* 47.4, pp. 547–553.

Daume III, Hal and Daniel Marcu (2006). "Domain adaptation for statistical classifiers". In: *Journal of Artificial Intelligence Research* 26, pp. 101–126.

Diehl, Christopher P and Gert Cauwenberghs (2003). "SVM incremental learning, adaptation and optimization". In: *Neural Networks, 2003. Proceedings of the International Joint Conference on*. Vol. 4. IEEE, pp. 2685–2690.

Duan, Lixin, Dong Xu, and Ivor Tsang (2012). "Learning with augmented features for heterogeneous domain adaptation". In: *arXiv preprint arXiv:1206.4660*.

Evgeniou, Theodoros and Massimiliano Pontil (2004). "Regularized multi–task learning". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 109–117.

Gretton, Arthur et al. (2009). "Covariate shift by kernel mean matching". In: *Dataset shift in machine learning* 3.4, p. 5.

He, Xiyan et al. (2011). "One-class SVM in multi-task learning". In: *Annual Conference of the European Safety and Reliability Association, ESREL 2011*.

He, Xiyan et al. (2014). "Multi-task learning with one-class SVM". In: *Neurocomputing* 133, pp. 416–426.

Hoffmann, Heiko (2007). "Kernel PCA for novelty detection". In: *Pattern Recognition* 40.3, pp. 863–874.

Khan, Shehroz S and Michael G Madden (2014). "One-class classification: taxonomy of study and review of techniques". In: *The Knowledge Engineering Review* 29.3, pp. 345–374.

Kulis, Brian, Kate Saenko, and Trevor Darrell (2011). "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, pp. 1785–1792.

Laskov, Pavel et al. (2006). "Incremental support vector learning: Analysis, implementation and applications". In: *The Journal of Machine Learning Research* 7, pp. 1909–1936.

Le, VK and Pierre Beauseroy (2015). "Path for Kernel Adaptive One-Class Support Vector Machine". In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, pp. 503–508.

Lee, Gyemin and Clayton D Scott (2007). "The one class support vector machine solution path". In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 2. IEEE, pp. II–521.

Li, Wen et al. (2014). "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation". In: *IEEE transactions on pattern analysis and machine intelligence* 36.6, pp. 1134–1148.

Lichman, M. (2013). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Long, Mingsheng et al. (2014). "Transfer joint matching for unsupervised domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1410–1417.

Maurer, Andreas, Massimiliano Pontil, and Bernardino Romera-Paredes (2016). "The benefit of multitask representation learning". In: *The Journal of Machine Learning Research* 17.1, pp. 2853–2884.

Moya, Mary M and Don R Hush (1996). "Network constraints and multi-objective optimization for one-class classification". In: *Neural Networks* 9.3, pp. 463–474.

Ng, Andrew (2017). "Artificial Intelligence is the New Electricity". In: *presentation at the Stanford MSx Future Forum*.

Pan, Sinno Jialin, James T Kwok, and Qiang Yang (2008). "Transfer Learning via Dimensionality Reduction." In: *AAAI*. Vol. 8, pp. 677–682.

Pan, Sinno Jialin and Qiang Yang (2010). "A survey on transfer learning". In: *Knowledge and Data Engineering, IEEE Transactions on* 22.10, pp. 1345–1359.

Pimentel, Marco AF et al. (2014). "A review of novelty detection". In: *Signal Processing* 99, pp. 215–249.

Quionero-Candela, Joaquin et al. (2009). *Dataset shift in machine learning*. The MIT Press.

Raina, Rajat et al. (2007). "Self-taught learning: transfer learning from unlabeled data". In: *Proceedings of the 24th international conference on Machine learning*. ACM, pp. 759–766.

Scholkopf, Bernhard and Alexander J Smola (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Schölkopf, Bernhard et al. (1999). "Support Vector Method for Novelty Detection." In: *NIPS*. Vol. 12, pp. 582–588.

Schölkopf, Bernhard et al. (2001). "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7, pp. 1443–1471.

Shi, Xiaoxiao et al. (2010). "Transfer learning on heterogenous feature spaces via spectral transformation". In: *2010 IEEE international conference on data mining*. IEEE, pp. 1049–1054.

Shimodaira, Hidetoshi (2000). "Improving predictive inference under covariate shift by weighting the log-likelihood function". In: *Journal of statistical planning and inference* 90.2, pp. 227–244.

Silverman, Bernard W (1986). *Density estimation for statistics and data analysis*. Vol. 26. CRC press.

Tarassenko, Lionel et al. (2009). "Novelty detection". In: *Encyclopedia of Structural Health Monitoring*.

Tax, David Martinus Johannes (2001). *One-class classification*. TU Delft, Delft University of Technology.

Tax, David MJ and Robert PW Duin (1999a). "Data domain description using support vectors." In: *ESANN*. Vol. 99, pp. 251–256.

Tax, David MJ and Robert PW Duin (1999b). "Support vector domain description". In: *Pattern recognition letters* 20.11, pp. 1191–1199.

Tax, David MJ and Robert PW Duin (2004). "Support vector data description". In: *Machine learning* 54.1, pp. 45–66.

Tax, David MJ and Pavel Laskov (2003). "Online SVM learning: from classification to data description and back". In: *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*. IEEE, pp. 499–508.

Tommasi, Tatiana, Francesco Orabona, and Barbara Caputo (2014). "Learning categories from few examples with multi model knowledge transfer". In: *IEEE transactions on pattern analysis and machine intelligence* 36.5, pp. 928–941.

Tong, Xin, Yang Feng, and Anqi Zhao (2016). "A survey on Neyman-Pearson classification and suggestions for future research". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 8.2, pp. 64–81.

Vapnik, Vladimir N (1995). "The Nature of Statistical Learning Theory". In:

Wang, Tian et al. (2013). "Online least squares one-class support vector machines-based abnormal visual event detection". In: *Sensors* 13.12, pp. 17130–17155.

Weiss, Karl, Taghi M Khoshgoftaar, and DingDing Wang (2016). "A survey of transfer learning". In: *Journal of Big Data* 3.1, p. 9.

Xue, Yongjian and Pierre Beauseroy (2016). "Multi-task learning for one-class SVM with additional new features". In: *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, pp. 1571–1576.

Xue, Yongjian and Pierre Beauseroy (2017). "Transfer learning for one class SVM adaptation to limited data distribution change". In: *Pattern Recognition Letters* 100, pp. 117–123.

Xue, Yongjian and Pierre Beauseroy (2018a). "Constant false alarm rate for online one-class SVM learning". In: *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, accepted.

Xue, Yongjian and Pierre Beauseroy (2018b). "Transfer Learning to Adapt One Class SVM Detection to Additional Features". In: *ICPRAM*, pp. 78–85.

Yang, Haiqin, Irwin King, and Michael R Lyu (2010). "Multi-task learning for one-class classification". In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, pp. 1–8.

Zhang, Yu (2015). "Multi-Task Learning and Algorithmic Stability." In: *AAAI*. Vol. 2. 4, pp. 6–2.

Zhang, Yu and Qiang Yang (2017). "A Survey on Multi-Task Learning". In: *arXiv preprint arXiv:1707.08114*.

Yongjian XUE

Doctorat : Optimisation et Sûreté des Systèmes

Année 2018

Transfert d'apprentissage dynamique pour la détection : une approche par apprentissage multitâche

Le but de cette thèse est de minimiser la perte de performance d'un système de détection lorsqu'il rencontre un changement de distribution de données à la suite d'un événement connu (maintenance, ajout de capteur etc.). L'idée est d'utiliser l'approche d'apprentissage par transfert pour exploiter l'information apprise avant l'événement pour adapter le détecteur au système modifié.

Un modèle d'apprentissage multitâche est proposé pour résoudre ce problème. Il utilise un paramètre pour équilibrer la quantité d'informations apportées par l'ancien système par rapport au nouveau. Ce modèle est formalisé de manière à pouvoir être résolu par un SVM mono-classe classique avec une matrice de noyau spécifique. Pour sélectionner le paramètre de contrôle, une méthode qui calcule les solutions pour toutes les valeurs du paramètre introduit et un critère de sélection de sa valeur optimale sont proposés. Les expériences menées dans le cas de changement de distribution et d'ajout de capteurs montrent que ce modèle permet une transition en douceur de l'ancien système vers le nouveau.

De plus, comme le modèle proposé peut être formulé comme un SVM mono-classe classique, des algorithmes d'apprentissage en ligne pour SVM mono-classe sont étudiés dans le but d'obtenir un taux de fausses alarmes stable au cours de la phase de transition. Ils peuvent être appliqués directement à l'apprentissage en ligne du modèle proposé.

Mots clés : apprentissage automatique - transfert d'apprentissage - détection du signal - détection des anomalies (informatique) - machines à vecteurs de support.

Dynamic Transfer Learning for One-class Classification: a Multi-task Learning Approach

The aim of this thesis is to minimize the performance loss of a one-class detection system when it encounters a data distribution change. The idea is to use transfer learning approach to transfer learned information from related old task to the new one. According to the practical applications, we divide this transfer learning problem into two parts, one part is the transfer learning in homogenous space and the other part is in heterogeneous space.

A multi-task learning model is proposed to solve the above problem; it uses one parameter to balance the amount of information brought by the old task versus the new task. This model is formalized so that it can be solved by classical one-class SVM except with a different kernel matrix. To select the control parameter, a kernel path solution method is proposed. It computes all the solutions along that introduced parameter and criteria are proposed to choose the corresponding optimal solution at given number of new samples. Experiments show that this model can give a smooth transition from the old detection system to the new one whenever it encounters a data distribution change.

Moreover, as the proposed model can be solved by classical one-class SVM, online learning algorithms for one-class SVM are studied later in the purpose of getting a constant false alarm rate. It can be applied to the online learning of the proposed model directly.

Keywords: machine learning - transfer of training - signal detection - anomaly detection (computer security) - support vector machines.

Thèse réalisée en partenariat entre :

Ecole Doctorale "Sciences pour l'Ingénieur"