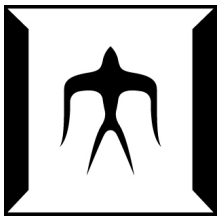
論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Density ratio approaches to multi-label classification and outlier detection
著者(和文)	Hyunha Nam
Author(English)	Hyunha Nam
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第10011号, 授与年月日:2015年9月25日, 学位の種別:課程博士, 審査員:杉山　将,秋山　泰,篠田　浩一,村田　剛志,藤井　敦
Citation(English)	Degree:, Conferring organization: Tokyo Institute of Technology, Report number:甲第10011号, Conferred date:2015/9/25, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Density ratio approaches to multi-label classification and outlier detection

Hyunha Nam

June 2015



Department of Computer Science
Graduate School of Information Science and Engineering
Tokyo Institute of Technology

Thesis Committee:
Masashi Sugiyama, Chair
Yutaka Akiyama
Koichi Shinoda
Tsuyoshi Murata
Atsushi Fujii

Submitted in partial fulfillment of
the requirements for the degree of

Doctor of Engineering

Keywords: Density ratio estimation, multi-label classification, outlier detection, deep learning, convolutional neural network

To my family

Abstract

The ratio of two probability density functions appears in various machine learning tasks such as conditional density estimation, outlier detection, non-stationarity adaptation, dimensionality reduction, independent component analysis, clustering and classification. Therefore the problem of estimating the density ratio is attracting a great deal of attention these days. Recently, direct density ratio estimation methods have been proposed and shown to be effective for many machine learning problems. The key idea of the direct density ratio approach is that the ratio is directly estimated so that difficult density estimation is avoided. There are still two major open topics to improve the density ratio approach in terms of accuracy and computational efficiency. This thesis focuses on them.

Firstly, we investigate a computationally efficient solution for more general problem setting. Multi-label classification allows a sample to belong to multiple classes simultaneously, which is often the case in real-world applications such as text categorization and image annotation. In multi-label scenarios, taking into account correlations among multiple labels can boost the classification accuracy. However, this makes classifier training more challenging because handling multiple labels induces a high-dimensional optimization problem. We propose a scalable multi-label method based on least-square approach to density ratio estimation. Through experiments, we show the usefulness of our proposed method.

Secondly, we investigate a deep model for density ratio estimation. So far,

parametric and non-parametric direct density ratio estimators with various loss functions have been developed, and the kernel least-squares method was demonstrated to be highly useful both in terms of accuracy and computational efficiency. On the other hand, recent study in pattern recognition exhibited that deep architectures such as a convolutional neural network can significantly outperform kernel methods. We propose to use the convolutional neural network in density ratio estimation, and experimentally show that the proposed method tends to outperform the kernel-based method in outlier detection tasks in images.

Given the encouraging experimental results of the proposed methods, we conclude that the proposed density ratio approaches with a deep model and multi-label problem setting are successful and worth a further study in the future.

Contents

List of Figures

List of Tables

Acknowledgments

First of all, I am deeply indebted to my academic supervisor, Professor Masashi Sugiyama, for his patient guidance. It has been an honor to be his Ph.D. student. He has provided me one of the best environments. Moreover, his valuable assistance and heartwarming encouragement was extraordinarily helpful, even though he was always unbelievably busy. I am also very grateful to Professor Yutaka Akiyama, Professor Koichi Shinoda, Associate Professor Tsuyoshi Murata, Associate Professor Atsushi Fujii for reviewing and evaluating my thesis.

Furthermore, I would like to thank all members in my lab.

My research projects were financially supported by the JSPS Global COE CompView program and JST PRESTO program and HASEGAWA scholarship. Without these generous supports, I could not lead the life in Japan. Hence, I would like to express my sincere appreciation to them.

The last but definitely not the least, I would like to express my special thanks to my family, friends and Jun. The love and support of them sustained me.

Chapter 1

Introduction

1.1 Density ratio in machine learning

Data is information for a computer to work on, usually in the form of statistics that can be analysed. The main goal of machine learning is to learn some rules from given data. Then we can extract useful information hidden in data using learned rules.

In statistical machine learning, observed data is assumed to be drawn from unobservable underlying probability density function. The ratio of these two probability densities is the *density ratio*. It appears in many machine learning tasks such as probabilistic classification, outlier detection, multi-task learning, non-stationarity adaptation, two-sample test, change-point detection in time series and conditional density estimation. We introduce the details of probabilistic classification, multi-task learning and outlier detection. These are closely related with the topics in thesis.

1.1.1 Probabilistic classification

The classification is a typical supervised learning problem. The goal of classification is to identify to which of classes a new observed data belongs. Probabilistic classification is based on the class-posterior probability given the new observed data. The class-posterior probability is used to acquire the confidence of class prediction. Some applications require the confidence of the predicted class label to improve the performance (Sugiyama et al., 2012b).

The class-posterior probability is expressed as

$$p(\mathbf{x}|y) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})},$$

where $\mathbf{x}$ is the data and y is a label. Therefore probabilistic classification can be carried out by estimating the density ratio.

1.1.2 Outlier detection

The outlier means inconsistent observation in a given dataset. Outliers occur because mechanical problems, fraudulent behaviour, human or instrument error and changes in system behaviour (Hodge and Austin, 2004).

Since outliers degrade performance significantly, outlier detection is a critical task in many applications. Outlier detection has been used to detect and remove anomalous data from a dataset. It is employed in many fields with different names: anomaly detection, novelty detection, noise detection and exception mining. For example, detecting unauthorized access in wireless sensor network (Branch et al., 2013), topic detection in news documents (Yamanishi et al., 2000), medical condition monitoring such as heart-rate and brain tumor monitoring (Ahmed et al., 2014; Marateb et al., 2012), object detection on streaming data (Assent et al., 2012), unexpected entries detection in a large data set (Buzzi-Ferraris and Manenti, 2011), identifying novel features or misclassified features in the satellite

images (Byers and Raftery, 1998), fraud detection (Fawcett and Provost, 1999) etc. For demands on various applications, outlier detection has been studied in statistics, neural networks, machine learning, data mining communities and also hybrid systems actively.

Usually outlier detection has the unsupervised learning scenario because most applications dose not have prior knowledge of the data. Even though in some applications supervised (Marsland, 2001), or semi-supervised (Fawcett and Provost, 1999; Gao et al., 2006) approaches are available and can perform better than the unsupervised approach, it is very expensive to get the label information in practice. Beside labeling diverse anomalous data is even harder. For this reason, supervised or semi-supervised setting is inappropriate to outlier detection.

The inlier-based outlier detection method is to identify outlier instances in the test set based on the training set consisting only of inlier instances (Hido et al., 2011). Information of inlier samples should be available, nevertheless the inlier-based outlier detection is more reasonable than the semi-supervised learning method due to getting the prior knowledge of inlier samples is simply done from the past observations. The ratio of training and test data densities is estimated as a measure of outlyingness of data. For example new observed data do not contain any outliers then the ratio of training and test data is one and samples with the ratio of training and test data close to zero are plausible to be outliers.

1.1.3 Multi-task learning

Multi-task learning is a method of inductive learning. The main idea is that data shares common information among different tasks and learning all tasks simultaneously and taking into relatedness behind the tasks that account for classifiers will get better performance than solving multiple learning tasks separately (Caruana, 1997).

There are several multi task learning methods: multi-task learning by naive sample sharing, adaptive sample sharing with importance sampling, and implicit sample sharing using regularization (Sugiyama et al., 2012b). Of these, multi-task learning by adaptive sample sharing with importance sampling exploits *importance* for measuring similarity between target task and other tasks. In the context of importance sampling, the density ratio is called the importance.

1.2 Algorithms of density ratio estimation

The goal of density ratio estimation is to estimate the density ratio $r(\mathbf{x})$.

$$r(\mathbf{x}) = \frac{p_{nu}(\mathbf{x})}{p_{de}(\mathbf{x})},$$

where $p_{nu}(\mathbf{x})$ and $p_{de}(\mathbf{x})$ are the probability densities from samples $\{\mathbf{x}_i^{\mathrm{nu}}\}_{i=1}^{n_{\mathrm{nu}}}$ and $\{\mathbf{x}_j^{\mathrm{de}}\}_{j=1}^{n_{\mathrm{de}}}$. "nu" and "de" mean "numerator" and "denominator", respectively.

1.2.1 Density estimation

A naive approach to solve density ratio estimation is to firstly estimate $p_{nu}(\mathbf{x})$ and $p_{de}(\mathbf{x})$ separately. Then the ratio of estimated densities is computed. However, this approach does not perform well. There are two reasons. Firstly, density estimation is a very hard problem. In parametric density estimation such as maximum likelihood estimation, we assume that we know the shape of the distribution. Often this assumption is erroneous in practice then we produce false results. In non-parametric density estimation such as kernel density estimator, we do not need such an assumption. However, it is inaccurate when observed data is insufficient (Kawahara and Sugiyama, 2012) and also very difficult in high dimensional problems (Schölkopf et al., 2001). Furthermore cross validation is computationally infeasible when bandwidths need to be selected for each dimension (Liu et al.,

2007). Secondly, the estimation error incurred in the first density estimation step can be magnified in the second step of computing their ratio (Sugiyama et al., 2012b). Below, a direct density ratio estimator that does not involve density estimation is reviewed.

1.2.2 Least-squares approach

In this section, we review the least-squares approach to direct density ratio estimation. This is called *least-squares importance fitting* (LSIF) (Sugiyama et al., 2012b). Let $\widehat{r}(\mathbf{x})$ be a model of the true density ratio. The following squared error SE is minimized:

$$\begin{aligned} SE(r) &:= \int (\widehat{r}(\mathbf{x}) - r(\mathbf{x}))^2 p_{de}(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &= \int \widehat{r}(\mathbf{x})^2 p_{de}(\mathbf{x}) \mathrm{d}\mathbf{x} - 2 \int \widehat{r}(\mathbf{x}) p_{nu}(\mathbf{x}) \mathrm{d}\mathbf{x} + \int r(\mathbf{x})^2 p_{nu}(\mathbf{x}) \mathrm{d}\mathbf{x}, \end{aligned}$$

where the last term is a constant so can be ignored. The first two terms are denoted by SE:

$$\widetilde{SE}(r) := \int \widehat{r}(\mathbf{x})^2 p_{de}(\mathbf{x}) \mathrm{d}\boldsymbol{x} - 2 \int \widehat{r}(\mathbf{x}) p_{nu}(\mathbf{x}) \mathrm{d}\boldsymbol{x},$$

which is empirically approximated by

$$\frac{1}{n_{\mathrm{de}}} \sum_{j=1}^{n_{\mathrm{de}}} r(\mathbf{x}_j^{\mathrm{de}})^2 - \frac{2}{n_{\mathrm{nu}}} \sum_{i=1}^{n_{\mathrm{nu}}} r(\mathbf{x}_i^{\mathrm{nu}}).$$

A linear density ratio model has been used for LSIF. The model of true density ratio $\widehat{r}(\mathbf{x})$ is modeled by

$$\widehat{r}(\mathbf{x}) = \sum_{b=1}^{B} \theta_b \phi_b(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\theta},$$

where B is the number of parameters and $\boldsymbol{\theta}$ is a parameter vector. $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^B$ is the basis function vector. In practice, we may use a kernel model such as Gaussian kernel.

LSIF is computationally highly efficient and easy to implement. In this thesis, we propose a computationally efficient probabilistic classifier for multi-label problems using LSIF and we introduce a novel deep model for LSIF. Details are described in next section.

1.3 Contributions of this thesis

1.3.1 An overview

We contribute to direct density ratio estimation algorithms for two computer vision topics: multi-label classification and outlier detection. Firstly, computer vision embraces the tasks for image processing, recognition, and annotation. These tasks can be effectively expressed by the multi-label framework that is an image containing more than one label information. Consequently, the issue of learning from multi-label data has attracted great attention from a lot of computer vision tasks. In this vein, we propose a new computationally efficient multi-label classification using direct density ratio estimation. Secondly, outlier detection in large and high dimensional image datasets is critical and a complex matter. Especially, mislabelling by a human error is one of the biggest factors of outliers. As a solution of this issue, we propose a new method of detecting outliers by estimating the density ratio based on a deep model. Then, we experimentally show that proposed method can detect mislabelled data.

In this section, we present our contributions in details.

1.3.2 Computationally efficient multi-label classification

In real-world, not only image annotation but many applications also (e.g. text categorization, audio instrument detection, etc) involve multiple label classes. These

problems can be expressed using the multi-label framework more naturally. Consequently, the issue of learning from multi-label data has attracted great attention from many fields in recent years.

In the previous research, the probabilistic classifier using least-squares approach to density ratio estimation has achieved promising results in the single-label classification (Sugiyama, 2010; Yamada et al., 2011) and the multi-task learning (Simm et al., 2011). However, when the size of dataset and the dimension of label are large, naive implementation of multi-label classification is more computationally expensive than single-label classification and multi-task learning. Because the essential number of training samples for multi-label classification is multiplied according to the dimension of the label.

To overcome this, we suggest computationally efficient solution for multi-label classification in this thesis.

1.3.3 Model exploring

We introduce practical implementation of density ratio estimation for a novel *deep model*. Most of the previous researches of density ratio estimation have used shallow models including the *linear and kernel models*, *log-linear model* and *Gaussian mixture model* (Sugiyama et al., 2012b).

- *Linear and kernel models*: The most widely used model for density ratio estimation (Sugiyama, 2010; Yamada et al., 2011; Sugiyama et al., 2008; Kanamori et al., 2009; Suzuki et al., 2009; Sugiyama et al., 2010).

- *log-linear model*: Another popular model for density ratio estimation.

$$\widehat{r}(\mathbf{x}; \boldsymbol{\theta}, \beta) = \exp(\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\theta} + \beta)),$$

  where β is a normalization parameter.

It has been used for *Kullback-Leibler importance estimation procedure* (KLIEP) (Tsuboi et al., 2009; Kanamori et al., 2010). KLIEP is a method of density-ratio estimation by density fitting under the Kullback-Leibler divergence (Sugiyama et al., 2008; Nguyen et al., 2010).

- *Gaussian mixture model*: A probabilistic model that assumes all data are generated from a mixture of Gaussian distributions. It has been also used for KLIEP (Yamada and Sugiyama, 2009).

$$\widehat{r}(\mathbf{x}; \{\theta_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^{c}) = \sum_{m=1}^{c} \theta_m N(\mathbf{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m),$$

where c is the number of mixing components, $\{\theta_m\}_{n=1}^{c}$ are mixing coefficients, $\{\boldsymbol{\mu}_m\}_{n=1}^{c}$ are means of Gaussian functions and $\{\boldsymbol{\Sigma}_m\}_{n=1}^{c}$ are covariance matrices of Gaussian functions. $N(\mathbf{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ is the multi-dimensional Gaussian density with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

These direct density ratio estimators with a kernel density-ratio model was demonstrated to be highly useful in terms of both accuracy and computational efficiency. On the other hands, recent studies in pattern recognition demonstrated that a deep model tends to perform better than shallow models (Larochelle et al., 2007). Especially, CNN are widely used models for images (Krizhevsky et al., 2012; Jarrett et al., 2009; Lee et al., 2009; Turaga et al., 2010; Ngiam et al., 2010).

In this thesis, we investigate a deep model for density ratio estimation. This allows us to extract features that are robust to spatial variations and have good internal information. We propose to use the CNN model in density ratio estimation and experimentally show that the proposed method tends to outperform the kernel-based method in outlier detection tasks in images.

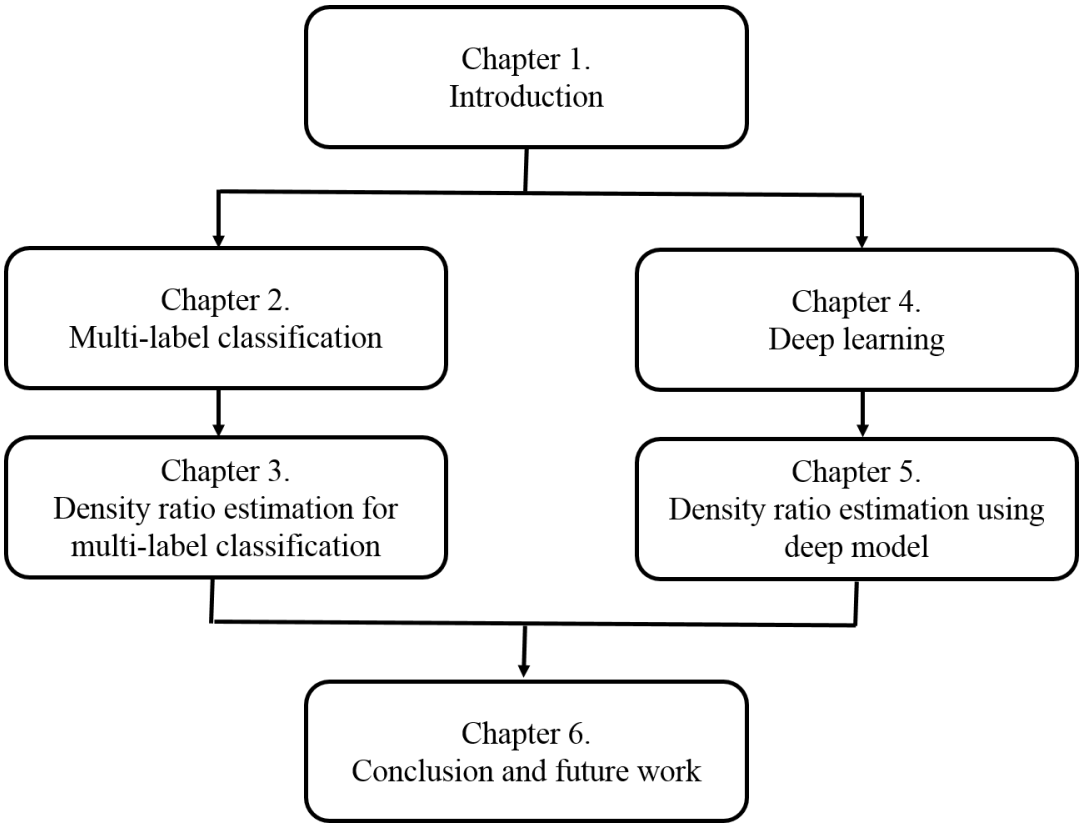


Figure 1.1: Organization of thesis.

1.4 Organization

The thesis consists of 6 chapters (see Figure 1.1). In this section we describe the organization of this thesis. Chapter 2 and Chapter 3 talk about the computationally efficient solutions for more general problem setting.

In Chapter 2, we introduce multi-label classification. Section 2.1 gives a brief introduction of the single-label classification and the multi-class classification. Section 2.2 reviews the multi-task classification and its applications. Section 2.3 shows an overview of multi-label classification and its application. Various transformation methods of the multi-label classification problem are introduced.

Chapter 3 covers our work on density ratio estimation for multi-label classi-

fication. Section 3.1 reviews the least-squared probabilistic classification (LSPC) for a single-label dataset. Section 3.2 shows the LSPC for multi-task learning. In Section 3.3, we extend multi-task LSPC to multi-label setup in a computationally efficient manner. We give the experiment results comparing with existing methods.

Subsequently, Chapters 4 and 5 are devoted to deep models for density ratio estimation. In Chapter 4, we present an deep learning method. Section 4.1 introduces deep learning. We review deep learning algorithms according to several criteria in Section 4.2. Especially, Section 4.2.1 and Section 4.2 explain deep autoencoders and convolutional neural networks (CNN).

In Chapater 5, we present density ratio estimation using deep models and demonstrate the performance of our proposed method in inlier-based outlier detection. In Section 5.1, we explain related studies and an overview of the proposed method. In Section 5.2, we introduce least-squares importance fitting (LSIF) that uses the squared-loss to fit a density ratio model to data. It is using a linear model. Section 5.3 derives how to apply the uLSIF criterion to the multilayer perceptron (MLP). We propose the uLSIF based on a deep CNN model in Section 5.4. Section 5.4.1 introduces the CNN model and Section 5.4.2 formulates the problem of training our method. In Section 5.5, we describe the inlier-based outlier detection and experimental setup. Then the proposed CNN-based uLSIF is compared with the kernel-based uLSIF and the kernel-based KLIEP in experiments of inlier-based outlier detection.

Finally, we summarize this thesis and show future works in Chapter 6.

Chapter 2

Multi-label classification

This chapter focuses on a multi-label scenario then introduces other problem settings, single-label classification, multi-task classification and those algorithms.

2.1 Single-label classification

Figure 2.1 shows the overview of single-label learning. In traditional single-label setting, a set of examples are associated with a single-label y from a set of disjoint classes. The classes is learned independently, so this is also called single-task learning.

A single-label dataset is composed of n examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)$, where $y \in \{1, \ldots, Y\}$, and Y is the number of classes. If $Y = 2$, this is binary classification or a binomial problem, otherwise $Y > 2$, this is multi-class or a multinomial classification problem. For example, in a binary classification problem, a medical image is classified into the disease or not and in a music retrieval system, a music file would be classified into the result set of a search or not by the relevance of a searching keyword. In a multi-class classification problem, a medical image can be labeled as one of these categories of organ "heart", "brain",

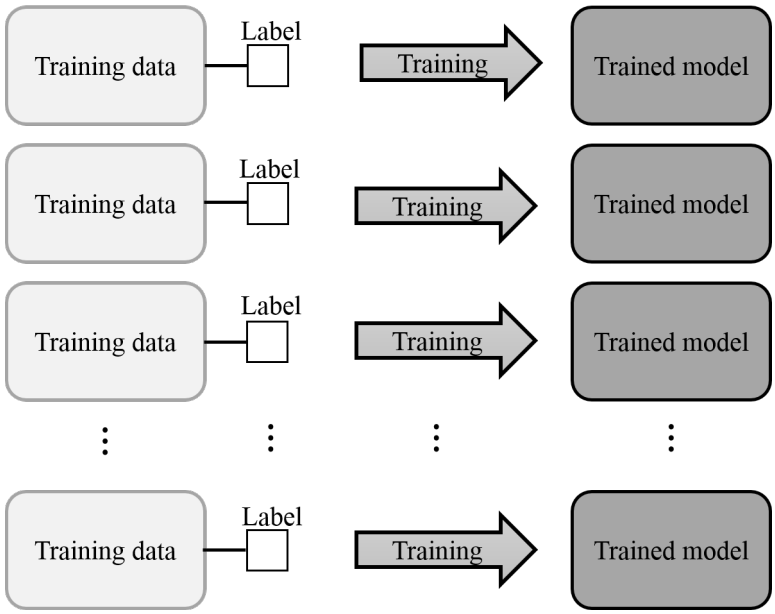


Figure 2.1: Single-label learning.

”pancreas”, etc. A music file also is labeled as one of various classes according to its genre, title, singer, etc.

2.2 Multi-task classification

Multi-task learning is a method of inductive learning. The main idea of multi-task learning is that learning all tasks simultaneously and taking into relatedness behind the tasks that account for classifiers will get better performance than solving multiple learning tasks separately. A multi-task learning dataset is given by $(\mathbf{x}_1, y_1, t_1), (\mathbf{x}_2, y_2, t_2), \ldots, (\mathbf{x}_n, y_n, t_n)$, where $t_n \in 1, \ldots, T$ denotes the task index. Figure 2.2 denotes the overview of multi-task learning. Each task may have different input data. All tasks are learned simultaneously to model the intrinsic relatedness between the tasks in contrast with single-task learning in Figure 2.1.

One application of multi-task learning is the recommendation system (Lenk

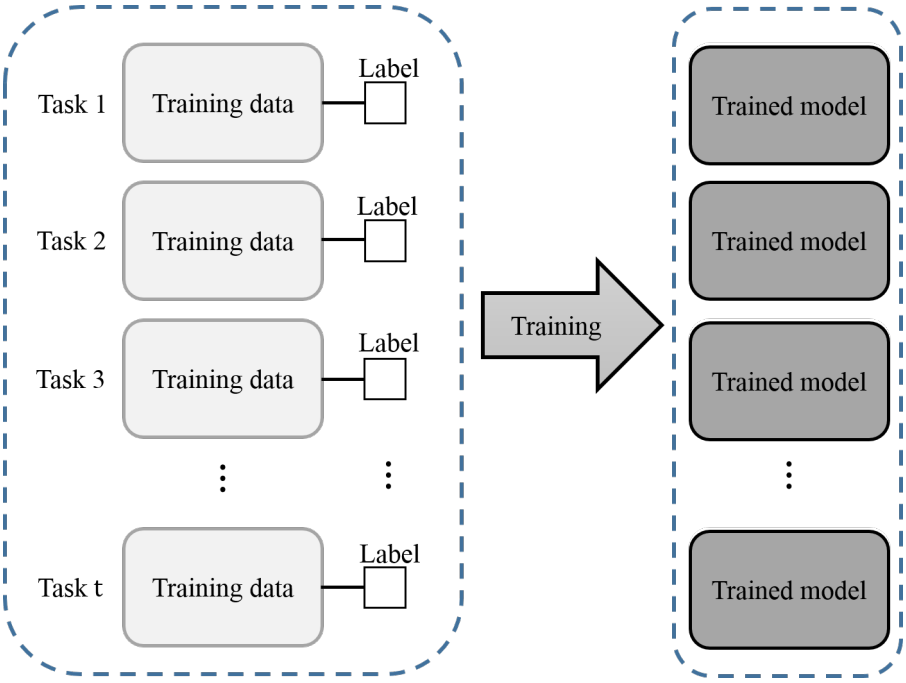


Figure 2.2: Multi-task learning.

et al., 1996). Each consumer is a task. The features of a product (to use car for example, fuel efficiency, drive type, transmission type, etc) are input data and output will be the score of user's preference to products. Similarly, email spam-filter (Attenberg et al., 2009) and web search (Chapelle et al., 2010) are also using multi-task learning.

2.3 Multi-label classification

The classification problem where a single sample can belong to multiple classes at the same time is called *multi-label classification*. In the multi-label problem setting, a sample can belong to a set of labels $\mathbf{y}$ at the same time. A dataset D is composed of n examples $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, where $\mathbf{y}_n = (y_{n,1}, \ldots, y_{n,T})^\top \in \{1, \ldots, Y\}$, and T is the number of labels. Figure 2.3 show the overview of multi-

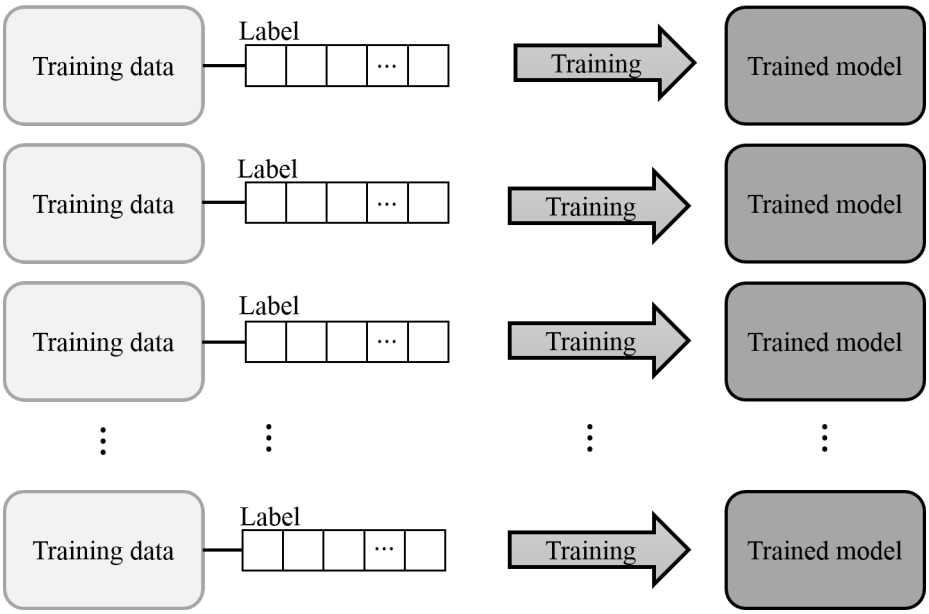


Figure 2.3: Multi-label learning.

label learning. Some labels may share the input data. All labels are learned simultaneously to model the label relatedness.

For example, in web-based audio repositories each audio signal is associated with various tags such as “acoustic”, “drum”, and “vocal”. Also, in image annotation tasks (Everingham et al., 2010), a single photo can be simultaneously annotated as “ocean”,“sun”, and “tree”. Similarly, in text mining (Project, 2009), a news article about *Transformer* can be categorized into the “car”, “robot”, and “movie” categories.

The multi-label classification problem can be transformed into different problem setting such as single-label classification or regression problems (Tsoumakas et al., 2010). We introduce the details of transformation methods and related previous studies below.

To describe transformation methods of the multi-label classification problem, we will use an example of the multi-label dataset describe in Table 2.1. In this

example, a data point is an audio file that belongs to one and more of four labels: violin, cello, piano and vocal.

Table 2.1: Example of a multi-label dataset

Label / Data	Violin	Cello	Piano	Vocal
Audio 1	1	0	1	0
Audio 2	0	0	1	1
Audio 3	1	0	0	0
Audio 4	0	1	1	0

The simplest way to solve the multi-label problem is to force the learning problem into single-label classification. Table 2.2 shows the first method that randomly or subjectively deletes labels, leaving one. The second method as shown in Table 2.3, removes every data which has multi-label information from the dataset. However, since these two methods have too rigid rules and discard a lot of meaningful information from the original dataset, we do not consider further in this thesis.

Table 2.2: Transformed dataset using first method

Label / Data	Violin	Cello	Piano	Vocal
Audio 1	0	0	1	0
Audio 2	0	0	0	1
Audio 3	1	0	0	0
Audio 4	0	1	0	0

The third method constructs the set of labels that exist in the dataset then con-

Table 2.3: Transformed dataset using second method

Label / Data	Violin	Cello	Piano	Vocal
Audio 3	1	0	0	0

sider each set of labels as a single-label (Boutell et al., 2004). Table 2.4 has the transformed result from Table 2.1. However, the third method is at the risk of too sparse data set (a large number of classes and few data per class) because it produces the set of labels in accordance with every data. Therefore the third method has been used in the past. In McCallum (1999), they did document recognition using the third transformation method. Based on probabilistic generative model a document is expressed by a mixture of the words distributions of its labels.

Table 2.4: Transformed dataset third method

Label / Data	Violin	Violin & piano	Piano & cello	Piano & vocal
Audio 1	0	1	0	0
Audio 2	0	0	1	0
Audio 3	1	0	0	0
Audio 4	0	0	0	1

The fourth method has been commonly used. This makes a transformed dataset D_y by duplicating dataset D as the number of labels T. Then each data has single-labels as in Table 2.5 that is the transformed result of the dataset described in Table 2.1 the dataset. This method is the same as treating the multi-class classification problem as a binary classification problem. This condition makes it hard to take label correlation into account. To get the output, this method restores

each label information from a duplicated dataset to multi-label vector (Li and Ogihara, 2003). In Zhang and Zhou (2007a), the k-nearest neighbor classifier for multi-label learning (ML-kNN) follows the paradigm of the fourth transformation method that ML-kNN uses the kNN algorithm independently for each label.

Table 2.5: Transformed dataset fourth method

Label / Data	Violin	The others
Audio 1	1	0
Audio 2	0	1
Audio 3	1	0
Audio 4	0	1

Label / Data	Cello	The others
Audio 1	0	1
Audio 2	0	1
Audio 3	0	1
Audio 4	1	0

Label / Data	Piano	The others
Audio 1	1	0
Audio 2	1	0
Audio 3	0	1
Audio 4	1	0

Label / Data	Vocal	The others
Audio 1	0	1
Audio 2	1	0
Audio 3	0	1
Audio 4	0	1

The fifth method transforms the multi-label dataset as shown in Table 2.6. Each data $(\mathbf{x}, \mathbf{y})$ is decomposed into $|\mathbf{y}|$ data $(\mathbf{x}, y)$ for all $y \in \mathbf{y}$. In this procedure, the loss of information is unavoidable by eliminating some labels. The transformed dataset is used for learning a single-label coverage-based classifier. In Chen et al. (2007), the fifth method is used for multi-label feature selection to improve multi-label classification performance.

The sixth method treats multi-label as respective T single-labels as shown in Table 2.7. Each data $(\mathbf{x}, \mathbf{y})$ is decomposed into T data $(\mathbf{x}, y)$. This simple method

Table 2.6: Transformed dataset fifth method

Data	Label
Audio 1a	Violin
Audio 1b	Piano
Audio 2a	Piano
Audio 2b	Vocal
Audio 3	Violin
Audio 4a	Cello
Audio 4b	Piano

is widely used for many adaptation methods which extend exist specific learning methods to deal with multi-label data.

In Schapire and Singer (2000); De Comité et al. (2003), AdaBoost is extended for multi-label data: AdaBoost.MH and AdaBoost.MR and a combination of AdaBoost.MH with the decision tree. The conditional random field is also used inGhamrawi and McCallum (2005) for multi-label classification. Back-propagation multi-label learning (BP-MLL) (Zhang and Zhou, 2006) extends the back-propagation algorithm for multi-label learning. They proposes a new cost function that takes multi-labels into account. The multi-class proposes multi-label perceptron (MMP) (Crammer and Singer, 2003) is the algorithm for multi-label ranking based on the perceptron algorithm.

The sixth method is straightforward and effective in applying to many adaptation methods. However, it is hard to take label correlation into account. In the next chapter, we introduce a multi-label classification method that use the sixth method and also can consider label correlation.

Table 2.7: Transformed dataset sixth method

Data		Label
Audio 1a	Violin	1
Audio 1b	Cello	0
Audio 1c	Piano	1
Audio 1d	Vocal	0
Audio 2a	Violin	0
Audio 2b	Cello	0
Audio 2c	Piano	1
Audio 2d	Vocal	1
Audio 3a	Violin	1
Audio 3b	Cello	0
Audio 3c	Piano	0
Audio 3d	Vocal	0
Audio 4a	Violin	0
Audio 4b	Cello	1
Audio 4c	Piano	1
Audio 4d	Vocal	0

Chapter 3

Density ratio estimation for multi-label classification

Multi-label classification allows a sample to belong to multiple classes simultaneously, which is often the case in real-world applications such as audio tagging and image annotation. In multi-label scenarios, taking into account correlation between multiple labels can boost the classification accuracy. However, this makes classifier training more challenging because handling multiple labels induces a high-dimensional optimization problem. In this chapter, we propose a scalable multi-label classifier based on the least-squares probabilistic classifier. The proposed method regards the multi-label classification problem as a multi-task learning problem. Through experiments, we show the usefulness of our proposed method.

3.1 Probabilistic Classification by LSPC

In this section, we review the *least-squares probabilistic classifier* (LSPC) for single-label classification (Sugiyama, 2010; Yamada et al., 2011).

Suppose that we are given a set of training samples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ drawn independently from a joint probability distribution with density $p(\boldsymbol{x}, y)$, where $\boldsymbol{x}_n \in \mathbb{R}^D$ is a feature vector, D is the dimensionality of feature vector $\boldsymbol{x}$, $y_n \in \{1, \ldots, Y\}$ is a class label, and Y is the number of classes. The objective of probabilistic classification is to learn the class-posterior probability $p(y|\boldsymbol{x})$ from the training samples. Based on the class-posterior probability, classification of a new sample $\boldsymbol{x}$ can be carried out by $\widehat{y} := \arg\max_{y \in \{1,\ldots,Y\}} p(y|\boldsymbol{x})$, with confidence $p(\widehat{y}|\boldsymbol{x})$.

For each $y \in \{1, \ldots, Y\}$, we model $p(y|\boldsymbol{x})$ by

$$q(y|\boldsymbol{x}; \boldsymbol{\theta}_y) := \sum_{b=1}^{B} \theta_{y,b} \phi_b(\boldsymbol{x}) = \boldsymbol{\theta}_y^\top \boldsymbol{\phi}(\boldsymbol{x}),$$

where B denotes the number of parameters, $\boldsymbol{\theta}_y = (\theta_{y,1}, \ldots, \theta_{y,B})^\top \in \mathbb{R}^B$ is the parameter vector, and $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_B(\boldsymbol{x}))^\top \in \mathbb{R}^B$ is the basis function vector. In practice, we may use a kernel model, i.e., we set $B = N$ and $\phi_b(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_b)$, where $K(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel function.

We fit the above model to the true class-posterior probability $p(y|\boldsymbol{x})$ under the following squared loss:

$$\begin{aligned} J_y(\boldsymbol{\theta}_y) &:= \frac{1}{2} \int \left(q(y|\boldsymbol{x}; \boldsymbol{\theta}_y) - p(y|\boldsymbol{x})\right)^2 p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &= \frac{1}{2} \int q(y|\boldsymbol{x}; \boldsymbol{\theta}_y)^2 p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - \int q(y|\boldsymbol{x}; \boldsymbol{\theta}_y) p(\boldsymbol{x}|y) p(y) \mathrm{d}\boldsymbol{x} + C, \end{aligned}$$

where $p(\boldsymbol{x})$ denotes the marginal density of feature vector $\boldsymbol{x}$ and C is a constant independent of $\boldsymbol{\theta}_y$. Approximating the expectations over $\boldsymbol{x}$ by sample averages and the class-prior probability $p(y)$ by sample ratios, ignoring constant C and fac-

tor $1/N$, and including an ℓ_2-regularizer, we have the following training criterion:

$$\begin{aligned}\widehat{J}_y(\boldsymbol{\theta}_y) &:= \frac{1}{2}\sum_{n=1}^{N} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_y)^2 - \sum_{n:y_n=y} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_y) + \frac{\rho}{2}\|\boldsymbol{\theta}_y\|^2 \\ &= \frac{1}{2}\boldsymbol{\theta}_y^\top \boldsymbol{\Phi}^\top \boldsymbol{\Phi}\boldsymbol{\theta}_y - \boldsymbol{\theta}_y^\top \boldsymbol{\Phi}^\top \boldsymbol{\pi}_y + \frac{\rho}{2}\|\boldsymbol{\theta}_y\|^2,\end{aligned}$$

where $\rho > 0$ is the regularization parameter, $\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N))^\top \in \mathbb{R}^{N\times B}$ is the design matrix, and $\boldsymbol{\pi}_y$ is the N-dimensional class-indicator vector defined, i.e., $\pi_{y,n} = 1$ if $y_n = y$ and $\pi_{y,n} = 0$ otherwise. We can obtain the minimizer $\widehat{\boldsymbol{\theta}}_y$ analytically as

$$\widehat{\boldsymbol{\theta}}_y = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \rho \boldsymbol{I}_B\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\pi}_y,$$

where $\boldsymbol{I}_B$ denotes the B-dimensional identity matrix.

As the number of training samples increases, the solution $q(y|\boldsymbol{x};\widehat{\boldsymbol{\theta}}_y)$ was shown to converge to the true class-posterior probability $p(y|\boldsymbol{x})$ with the optimal convergence rate (Sugiyama, 2010). For a finite sample size, we obtain the final solution by rounding up a negative output to zero and normalization as follows (Yamada et al., 2011):

$$\widehat{p}(y|\boldsymbol{x}) = \frac{\max(0, q(y|\boldsymbol{x};\widehat{\boldsymbol{\theta}}_y))}{\sum_{y'=1}^{Y} \max(0, q(y'|\boldsymbol{x};\widehat{\boldsymbol{\theta}}_{y'}))}.$$

3.2 Multi-Task LSPC

When multiple related learning tasks exist, solving them simultaneously by sharing some common information behind the tasks is expected to be more promising than solving them separately. This is the idea of *multi-task learning*. A computationally efficient multi-task learning method can be developed by combining multiple LSPCs. Here, we review multi-task LSPC (MT-LSPC) (Simm et al., 2011) in a slightly generalized way.

Suppose that we are given a set of training samples $\{(\boldsymbol{x}_n, y_n, t_n)\}_{n=1}^N$, where $t_n \in \{1, \ldots, T\}$ denotes the task index. We assume that $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ are drawn independently from a joint probability distribution with density $p_{t_n}(\boldsymbol{x}, y)$. The objective of multi-task probabilistic classification is to learn the class-posterior probabilities $p_t(y|\boldsymbol{x})$ for $t \in \{1, \ldots, T\}$.

Let us model $p_t(y|\boldsymbol{x})$ for each $t \in \{1, \ldots, T\}$ and $y \in \{1, \ldots, Y\}$ as

$$q(y|\boldsymbol{x}; \boldsymbol{\theta}_{y,t}) := \sum_{b=1}^{B} \theta_{y,b,t} \phi_b(\boldsymbol{x}) = \boldsymbol{\theta}_{y,t}^\top \boldsymbol{\phi}(\boldsymbol{x}),$$

where $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_B(\boldsymbol{x}))^\top \in \mathbb{R}^B$ and $\boldsymbol{\theta}_{y,t} := (\theta_{y,1,t}, \ldots, \theta_{y,B,t})^\top \in \mathbb{R}^B$. The basic idea of MT-LSPC is that solutions of all tasks are imposed to be close to each other in terms of the ℓ_2-norm. More specifically, let us decompose $\boldsymbol{\theta}_{y,t}$ as $\boldsymbol{\theta}_{y,t} = \boldsymbol{\beta}_{y,0} + \boldsymbol{\beta}_{y,t}$, where $\boldsymbol{\beta}_{y,0}$ is the common part of solutions for all tasks and $\boldsymbol{\beta}_{y,t}$ is the individual part of solutions for task t. Then, for $\boldsymbol{\beta}_y := \left(\boldsymbol{\beta}_{y,0}^\top, \boldsymbol{\beta}_{y,1}^\top, \ldots, \boldsymbol{\beta}_{y,T}^\top\right)^\top \in \mathbb{R}^{B(T+1)}$, the training criterion of MT-LSPC is given by

$$\begin{aligned}\widehat{J}_y^{\mathrm{MT}}(\boldsymbol{\beta}_y) := \frac{1}{2}\sum_{n=1}^{N} q(y|\boldsymbol{x}_n; \boldsymbol{\beta}_{y,0} + \boldsymbol{\beta}_{y,t_n})^2 - \sum_{n:y_n=y} q(y|\boldsymbol{x}_n; \boldsymbol{\beta}_{y,0} + \boldsymbol{\beta}_{y,t_n}) \\ + \frac{\omega_0}{2}\|\boldsymbol{\beta}_{y,0}\|^2 + \frac{1}{2}\sum_{t=1}^{T} \omega_t \|\boldsymbol{\beta}_{y,t}\|^2,\end{aligned}$$

where $\omega_0 > 0$ is the regularization parameter for the task-independent part and $\omega_t > 0$ $(t = 1, \ldots, T)$ is the regularization parameter for the task-dependent parts.

For $\mathbf{0}_B$ denoting the B-dimensional zero vector, let

$$\begin{aligned}\boldsymbol{\xi}_t(\boldsymbol{x}) &:= \left(\boldsymbol{\phi}(\boldsymbol{x})^\top, \mathbf{0}_{B(t-1)}^\top, \boldsymbol{\phi}(\boldsymbol{x})^\top, \mathbf{0}_{B(T-t)}^\top\right)^\top \in \mathbb{R}^{B(T+1)}, \\ \boldsymbol{\Xi} &:= (\boldsymbol{\xi}_{t_1}(\boldsymbol{x}_1), \ldots, \boldsymbol{\xi}_{t_N}(\boldsymbol{x}_N))^\top \in \mathbb{R}^{N \times B(T+1)}, \\ \boldsymbol{\Omega} &:= \mathrm{diag}\,(\omega_0, \omega_1, \ldots, \omega_T) \in \mathbb{R}^{(T+1)\times(T+1)}.\end{aligned}$$

Then the MT-LSPC training criterion can be compactly expressed as

$$\widehat{J}_y^{\mathrm{MT}}(\boldsymbol{\beta}_y) = \frac{1}{2}\boldsymbol{\beta}_y^\top \boldsymbol{\Xi}^\top \boldsymbol{\Xi} \boldsymbol{\beta}_y - \boldsymbol{\beta}_y^\top \boldsymbol{\Xi}^\top \boldsymbol{\pi}_y + \frac{1}{2}\boldsymbol{\beta}_y^\top (\boldsymbol{\Omega} \otimes \boldsymbol{I}_B)\boldsymbol{\beta}_y,$$

where $\otimes$ denotes the *Kronecker product*. Because the above $\widehat{J}_y^{\mathrm{MT}}(\boldsymbol{\beta}_y)$ is essentially the same form as the original single-task LSPC training criterion, we can obtain the minimizer $\widehat{\boldsymbol{\beta}}_y$ analytically as

$$\widehat{\boldsymbol{\beta}}_y = \left(\boldsymbol{\Xi}^\top \boldsymbol{\Xi} + \boldsymbol{\Omega} \otimes \boldsymbol{I}_B\right)^{-1} \boldsymbol{\Xi}^\top \boldsymbol{\pi}_y.$$

Suppose that we use a kernel model (i.e., $B = N$). Then, the size of the matrix to be inverted in the above equation is $N(T+1) \times N(T+1)$. Thus, the computational complexity for naively computing the solution $\widehat{\boldsymbol{\beta}}_y$ is $\mathcal{O}(N^3T^3)$, which can be expensive. However, because the rank of $\boldsymbol{\Xi}^\top\boldsymbol{\Xi}$ is at most N, the solution can be computed more efficiently. More specifically, $q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t})$ can be expressed as follows:

$$q(y|\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_{y,t}) = \widehat{\boldsymbol{\theta}}_{y,t}^\top \boldsymbol{\phi}(\boldsymbol{x}) = \widehat{\boldsymbol{\beta}}_y^\top \boldsymbol{\xi}_t(\boldsymbol{x}) = \boldsymbol{\pi}_y^\top \boldsymbol{A}^{-1} \boldsymbol{b}_t,$$

where $\boldsymbol{A}$ is the $N \times N$ matrix and $\boldsymbol{b}_t$ is the N-dimensional vector defined as

$$\begin{aligned} A_{n,n'} &:= [\boldsymbol{\Xi}(\boldsymbol{\Omega}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\Xi}^\top + \boldsymbol{I}_N]_{n,n'} \\ &= \left(\frac{1}{\omega_0} + \frac{\delta_{t_n,t_{n'}}}{\omega_{t_n}}\right) \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}_{n'}) + \delta_{n,n'}, \\ b_{t,n} &:= [\boldsymbol{\Xi}(\boldsymbol{\Omega}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\xi}_t(\boldsymbol{x})]_n = \left(\frac{1}{\omega_0} + \frac{\delta_{t,t_n}}{\omega_t}\right) \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}). \end{aligned}$$

Here $\delta_{t,t'}$ denotes the *Kronecker delta*. The computational complexity for computing the solution in this way is reduced to $\mathcal{O}(N^3)$, which is independent of T.

3.2.1 Reformulation of MT-LSPC

In this chapter, we develop a multi-label method based on MT-LSPC. However, the original MT-LSPC imposes all solutions to be close to each other via the common part, which is not necessarily appropriate in the multi-label scenario. Here,

we derive an extension of MT-LSPC that imposes a multi-task penalty via pairwise similarities between tasks. This pairwise version will be used for developing a multi-label method later.

For $\boldsymbol{\theta}_y := (\boldsymbol{\theta}_{y,1}^\top, \ldots, \boldsymbol{\theta}_{y,T}^\top)^\top \in \mathbb{R}^{BT}$, let us consider the following training criterion:

$$\widehat{J}_y^{\mathrm{MT}'}(\boldsymbol{\theta}_y) := \frac{1}{2}\sum_{n=1}^{N} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_{y,t_n})^2 - \sum_{n:y_n=y} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_{y,t_n}) + \frac{1}{2}\sum_{t=1}^{T}\lambda_t\|\boldsymbol{\theta}_{y,t}\|^2 + \frac{1}{4}\sum_{t,t'=1}^{T}\gamma_{t,t'}\|\boldsymbol{\theta}_{y,t}-\boldsymbol{\theta}_{y,t'}\|^2,$$

where $\lambda_t > 0$ is the regularization parameter for task t and $\gamma_{t,t'} > 0$ is the similarity between tasks t and t' (large $\gamma_{t,t'}$ corresponds to similar tasks). Let

$$\boldsymbol{\psi}_t(\boldsymbol{x}) := \left(\mathbf{0}_{B(t-1)}^\top, \boldsymbol{\phi}(\boldsymbol{x})^\top, \mathbf{0}_{B(T-t)}^\top\right)^\top \in \mathbb{R}^{BT},$$
$$\boldsymbol{\Psi} := (\boldsymbol{\psi}_{t_1}(\boldsymbol{x}_1), \ldots, \boldsymbol{\psi}_{t_N}(\boldsymbol{x}_N))^\top \in \mathbb{R}^{N\times BT}.$$

Then $\widehat{J}_y^{\mathrm{MT}'}$ can be compactly expressed as

$$\widehat{J}_y^{\mathrm{MT}'}(\boldsymbol{\theta}_y) = \frac{1}{2}\boldsymbol{\theta}_y^\top\boldsymbol{\Psi}^\top\boldsymbol{\Psi}\boldsymbol{\theta}_y - \boldsymbol{\theta}_y^\top\boldsymbol{\Psi}^\top\boldsymbol{\pi}_y + \frac{1}{2}\boldsymbol{\theta}_y^\top(\boldsymbol{C}\otimes\boldsymbol{I}_B)\boldsymbol{\theta}_y,$$

where $\boldsymbol{C}$ is the $T\times T$ matrix defined as

$$C_{t,t'} := \delta_{t,t'}\left(\lambda_t + \sum_{t''=1}^{T}\gamma_{t,t''}\right) - \gamma_{t,t'}.$$

Taking the derivative of $\widetilde{J}_y^{\mathrm{MT}}$ with respect to $\boldsymbol{\theta}_y$ and setting it to zero, we have the minimizer $\widehat{\boldsymbol{\theta}}_y$ analytically as

$$\widehat{\boldsymbol{\theta}}_y = \left(\boldsymbol{\Psi}^\top\boldsymbol{\Psi} + \boldsymbol{C}\otimes\boldsymbol{I}_B\right)^{-1}\boldsymbol{\Psi}^\top\boldsymbol{\pi}_y.$$

Using the same trick as MT-LSPC, $q(y|\boldsymbol{x};\widehat{\boldsymbol{\theta}}_{y,t})$ can be efficiently computed based on the following expression:

$$q(y|\boldsymbol{x};\widehat{\boldsymbol{\theta}}_{y,t}) = \widehat{\boldsymbol{\theta}}_{y,t}^\top\boldsymbol{\phi}(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_y^\top\boldsymbol{\psi}_t(\boldsymbol{x}) = \boldsymbol{\pi}_y^\top\boldsymbol{A}'^{-1}\boldsymbol{b}'_t,$$

where $\boldsymbol{A}'$ is the $N \times N$ matrix and $\boldsymbol{b}'_t$ is the N-dimensional vector defined as

$$
\begin{aligned}
A'_{n,n'} &:= [\boldsymbol{\Psi}(\boldsymbol{C}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\Psi}^\top + \boldsymbol{I}_N]_{n,n'} \\
&= [\boldsymbol{C}^{-1}]_{t_n,t_{n'}} \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}_{n'}) + \delta_{n,n'}, \\
b'_{t,n} &:= [\boldsymbol{\Psi}(\boldsymbol{C}^{-1} \otimes \boldsymbol{I}_B)\boldsymbol{\psi}_t(\boldsymbol{x})]_n = [\boldsymbol{C}^{-1}]_{t,t_n} \boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{\phi}(\boldsymbol{x}).
\end{aligned}
$$

The computational complexity for computing the solution in this way is reduced to $\mathcal{O}(N^3 + T^3)$. Note that the factor T^3 comes from the computation of $\boldsymbol{C}^{-1}$; if the task similarity matrix $\boldsymbol{\Gamma}$ (with $\Gamma_{t,t'} = \gamma_{t,t'}$) enjoys nice structure such as being low-rank or sparse, it may be computed more efficiently.

3.3 Multi-Label LSPC

3.3.1 Methodology

In this section, we propose a computationally efficient multi-task classifier based on LSPC called *multi-label LSPC* (ML-LSPC).

Suppose that we are given a set of training samples $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$, where $\boldsymbol{y}_n = (y_{n,1}, \ldots, y_{n,T})^\top \in \{0,1\}^\top$ is the class-label vector for the n-th sample and T is the number of labels. Input vector $\boldsymbol{x}$ is assumed to be drawn independently from $p(\boldsymbol{x})$, and the t-th element y_t of $\boldsymbol{y} = (y_1, \ldots, y_t)^\top$ is assumed to be drawn from $p_t(y|\boldsymbol{x})$. The objective of multi-label probabilistic classification is to learn the class-posterior probabilities $p_t(y|\boldsymbol{x})$ for $t \in \{1, \ldots, T\}$.

Requiring that similar labels should have similar classification solutions, we can employ a multi-task learning method to solve the multi-label learning problem. Indeed, from the MT-LSPC training criterion, we immediately have the train-

ing criterion for ML-LSPC:

$$\widehat{J}_y^{\mathrm{ML}}(\boldsymbol{\theta}_y) := \sum_{t=1}^{T}\left(\frac{1}{2}\sum_{n=1}^{N} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_{y,t})^2 - \sum_{n:y_{n,t}=y} q(y|\boldsymbol{x}_n;\boldsymbol{\theta}_{y,t}) + \frac{1}{2}\lambda_t\|\boldsymbol{\theta}_{y,t}\|^2\right) + \frac{1}{4}\sum_{t,t'=1}^{T}\gamma_{t,t'}\|\boldsymbol{\theta}_{y,t}-\boldsymbol{\theta}_{y,t'}\|^2.$$

However, a notable difference between multi-task and multi-label formulations is that the number of training samples is N in the multi-task formulation, whereas that in the multi-label formulation is essentially NT. Thus, if we naively apply MT-LSPC to the multi-label problem, the computational complexity is $\mathcal{O}(N^3T^3)$ for a kernel model (i.e., $B = N$), which is expensive. Below, we explain how to mitigate this problem.

Let $\boldsymbol{\Theta}_y := (\boldsymbol{\theta}_{y,1},\ldots,\boldsymbol{\theta}_{y,T}) \in \mathbb{R}^{B\times T}$. Let $\boldsymbol{\pi}_{y,t}$ be the N-dimensional class-indicator vector for the t-th label, i.e., $\pi_{y,t,n} = 1$ if $y_{n,t} = y$, and $\pi_{y,t,n} = 0$ otherwise. Let $\boldsymbol{\Pi}_y := (\boldsymbol{\pi}_{y,1},\ldots,\boldsymbol{\pi}_{y,T}) \in \mathbb{R}^{N\times T}$. Then $\widehat{J}_y^{\mathrm{ML}}$ can be compactly expressed as

$$\widehat{J}_y^{\mathrm{ML}}(\boldsymbol{\theta}_y) = \frac{1}{2}\operatorname{tr}\boldsymbol{\Theta}_y^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Theta}_y - \operatorname{tr}\boldsymbol{\Theta}_y^\top\boldsymbol{\Phi}^\top\boldsymbol{\Pi}_y + \frac{1}{2}\operatorname{tr}\boldsymbol{\Theta}_y\boldsymbol{C}\boldsymbol{\Theta}_y^\top.$$

Taking the derivative of the above equation with respect to $\boldsymbol{\Theta}_y$ and setting it to zero, we obtain

$$\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Theta}_y + \boldsymbol{\Theta}_y\boldsymbol{C} = \boldsymbol{\Phi}^\top\boldsymbol{\Pi}_y. \tag{3.1}$$

This is called the *continuous Sylvester equation* with respect to $\boldsymbol{\Theta}_y$, which often arises in control theory (Sima, 1996).

Various algorithms for solving the Sylvester equation have been developed. One of the simplest methods is based on the eigenvalue decompositions of $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ and $\boldsymbol{C}$ as follows: Let $\boldsymbol{f}_1,\ldots,\boldsymbol{f}_B$ be eigenvectors of $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ associated with eigenvalues $f_1,\ldots,f_B$, and let $\boldsymbol{g}_1,\ldots,\boldsymbol{g}_T$ be eigenvectors of $\boldsymbol{C}$ associated with eigen-

values $g_1, \ldots, g_T$. Then the solution $\widehat{\boldsymbol{\Theta}}_y$ to Eq.(3.1) is given analytically as

$$\widehat{\boldsymbol{\Theta}}_y = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_B)\boldsymbol{Q}(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_T)^\top,$$

where $\boldsymbol{Q}$ is the $B \times T$ matrix defined as

$$Q_{b,t} := \frac{\boldsymbol{f}_b^\top \boldsymbol{\Phi}^\top \boldsymbol{\Pi}_y \boldsymbol{g}_t}{f_b + g_t}.$$

If a kernel model is used (i.e., $B = N$), the computational complexity for solving Eq.(3.1) in this way is $\mathcal{O}(N^3 + N^2T + NT^2 + T^3)$. Note that the terms N^3 and T^3 come from the eigenvalue decompositions of $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ and $\boldsymbol{C}$, which can be performed more efficiently if they enjoy nice structure such as being low-rank or sparse.

For large-scale data, Eq.(3.1) may be solved more efficiently by numerical optimization. Let $\boldsymbol{\theta}_y := (\boldsymbol{\theta}_{y,1}^\top, \ldots, \boldsymbol{\theta}_{y,T}^\top)^\top \in \mathbb{R}^{BT}$. Then Eq.(3.1) can be expressed as

$$\boldsymbol{H}\boldsymbol{\theta}_y = \boldsymbol{h}_y,$$

where

$$\boldsymbol{H} := \boldsymbol{I}_T \otimes (\boldsymbol{\Phi}^\top\boldsymbol{\Phi}) + \boldsymbol{C} \otimes \boldsymbol{I}_B \in \mathbb{R}^{BT \times BT},$$
$$\boldsymbol{h}_y := ((\boldsymbol{\Phi}^\top\boldsymbol{\pi}_{y,1})^\top, \ldots, (\boldsymbol{\Phi}^\top\boldsymbol{\pi}_{y,T})^\top)^\top \in \mathbb{R}^{BT}.$$

If a kernel model is used (i.e., $B = N$), naively solving $\boldsymbol{H}\boldsymbol{\theta}_y = \boldsymbol{h}_y$ takes $\mathcal{O}(N^3T^3)$ time. Here, we take into account the Kronecker structure of $\boldsymbol{H}$, and solve the equation numerically by the *conjugate gradient* method. More specifically, we can compute the matrix-vector product $\boldsymbol{H}\boldsymbol{\theta}_y$ as

$$[\boldsymbol{H}\boldsymbol{\theta}_y]_t = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\theta}_{y,t} + \sum_{t'=1}^{T} C_{t,t'}\boldsymbol{\theta}_{y,t'}.$$

Although the computational complexity for naively computing $\boldsymbol{H}\boldsymbol{\theta}_y$ is $\mathcal{O}(N^3 + N^2T^2)$ including the computation of $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$, that for computing $\boldsymbol{H}\boldsymbol{\theta}_y$ based on the

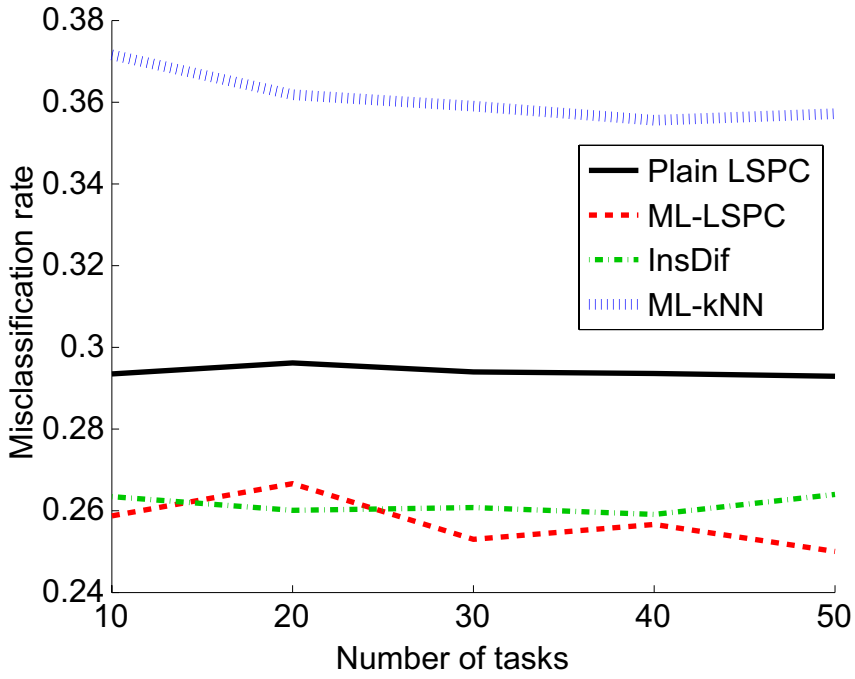


Figure 3.1: Misclassification rate

above expression is reduced to $\mathcal{O}(N^2T + NT^2)$. Note that the term N^2T comes from the computation $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\boldsymbol{\theta}_{y,t}$ and the term NT^2 comes from the computation $\sum_{t'=1}^{T} C_{t,t'}\boldsymbol{\theta}_{y,t'}$. If $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ is approximated by a low-rank matrix and the task similarity matrix $\boldsymbol{\Gamma}$ enjoys nice structure such as being approximately low-rank or sparse, $\boldsymbol{H}\boldsymbol{\theta}_y$ may be approximately computed even more efficiently.

Below, we experimentally evaluate the performance of the proposed ML-LSPC.

3.3.2 Experiments

Toy Dataset

Let the feature dimension be $D = 300$, and we consider T binary classification tasks. Training samples of the t-th task is created as follows: $\boldsymbol{x}_n = (x_{1,n}, \ldots, x_{D,n})^\top$

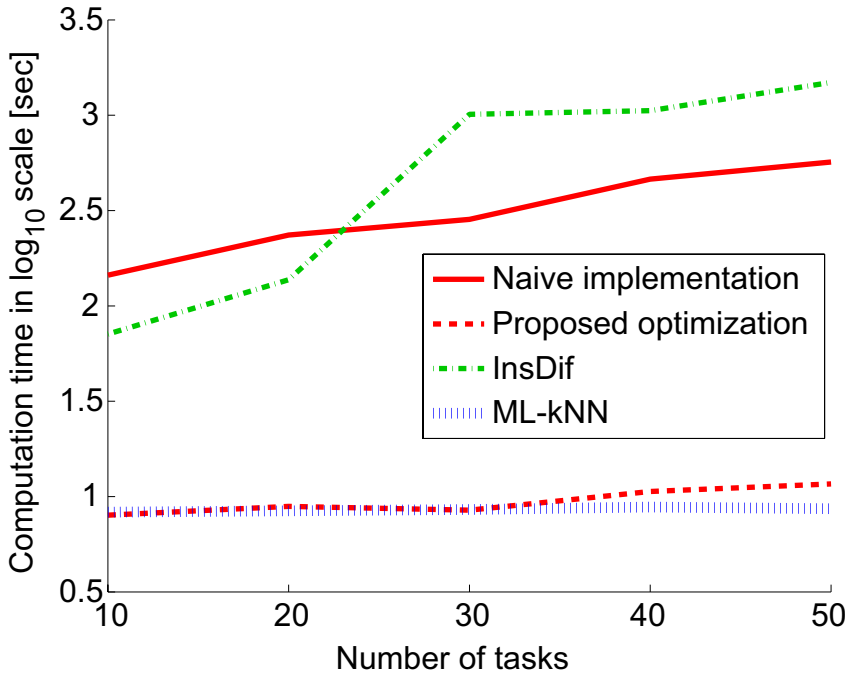


Figure 3.2: Computation time

is independently drawn from the standard normal distribution and $y_{t,n}$ is determined by linear decision boundary

$$\cos(2\pi t/T)x_{1,n} + \sin(2\pi t/T)x_{2,n}$$

(i.e., the decision boundaries are rotated in the subspace spanned by the first two dimensions). We set the number of training samples to $N = 2000$. The label similarity $W_{t,t'}$ is set to $\max(0, \rho_{t,t'})$, where $\rho_{t,t'}$ is the Pearson correlation coefficient between $\{y_{t,n}\}_{n=1}^{N}$ and $\{y_{t',n}\}_{n=1}^{N}$. We use the Gaussian kernel model in LSPCs.

We compare the classification performance of the plain LSPC (i.e., each task is solved separately), the proposed ML-LSPC, the *instance differentiation* method (InsDif) (Zhang and Zhou, 2007a), and the k-*nearest neighbor classifier for multi-*

label learning (ML-*k*NN) (Zhang and Zhou, 2007b) as functions of the number of tasks. All tuning parameters were optimized based on 5-fold cross-validation in terms of the misclassification rate. Figure 3.1 plots the average misclassification rate over 50 runs, showing that ML-LSPC and InsDif perform well. Figure 3.2 plots the computation time of ML-LSPC with naive implementation (we used the left-division function '*mldivide*' in MATLAB®), the proposed optimization method (we used the conjugate gradient function '*pcg*' in MATLAB®), InsDif, and ML-*k*NN. This shows that the proposed optimization method is computationally much more efficient than the naive implementation of ML-LSPC and InsDif.

Enron Email Dataset

Finally, we test the performance of the proposed method on the *Enron Email Dataset*, which consists of 1072 real-world email messages (Project, 2009). Each email message is represented as a 1001-dimensional feature vector, accompanied with 53 labels such as newsletters, jokes, trip reports, worry, etc. We randomly chose $N = 1000$ samples for training, and used the remaining 702 samples for performance evaluation. Because the presence and absence of labels were highly imbalanced in this dataset, we decided to evaluate the test performance (and model selection by cross-validation) in terms of the *F-measure*. The average F-scores for plain LSPC, ML-LSPC, InsDif, and ML-*k*NN over 50 runs were 0.554, 0558, 0.544, and 0.437. This shows that ML-LSPC overall compares favorably with other approaches.

3.4 Discussion

Multi-label classification is useful in various real-world problems such as audio tagging, image annotation, video search, and text mining. However, be-

cause the essential number of training samples for T-dimensional label vectors of size N is NT, naive implementation of multi-label classification is computationally expensive when N and T are large. To overcome this computational bottleneck, we developed a multi-label method based on computationally efficient LSPC (Sugiyama, 2010; Yamada et al., 2011).

Our key idea was to utilize the block structure of the system of linear equations to improve the computational efficiency. Also, we can reduce the computational complexity via the Sylvester equation. Usually, the multi-label classification is transformed into multiple single-label classification or regression problems for computational convenience. However, it means that we miss a chance to use the intrinsic relatedness between labels. In the proposed method, we employ the affinity matrix that measures the similarity between labels of transformed data to improve performance. Through experiments, we showed that the proposed method, ML-LSPC, is promising.

Chapter 4

Deep learning

This chapter provides the overview of the deep learning method and its application.

4.1 What is deep learning?

Deep learning is also called *deep structured learning* (Yu and Deng, 2011), *deep architectures learning* (Bengio, 2009) or *hierarchical learning* (Hinton et al., 2012a). Deep architecture is composed of multi-layers of non-linear operations such as lots of hidden layers in neural networks and many levels of hidden variables in graphical models.

The last decades, the deep learning is becoming a mature field of machine learning with impressive performance obtained in many application areas, especially in computer vision and speech. Deep learning is associated with many fields: machine learning, optimization, neural networks, pattern recognition, artificial intelligence and signal processing. There are three essential advances that can make deep learning attract a great deal of attention these days (Yu and Deng, 2011). Firstly, computational abilities (e.g., graphical processing units) is im-

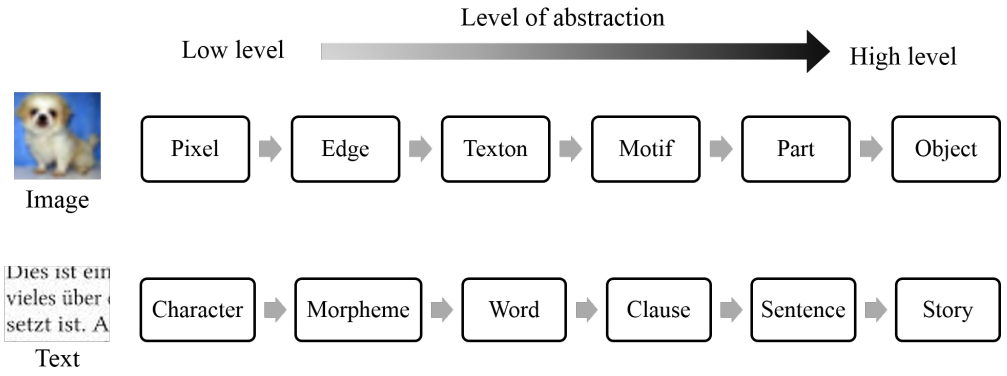


Figure 4.1: Hierarchy of representations

proved drastically in these days. Secondly, we can use large datasets efficiently for training. At last, remarkable advances in machine learning research have been achieved recently. These advances enable the deep learning methods to exploit complex non-linear operations to learn hierarchical feature representations. We introduce more about deep learning and details of important deep learning methods.

The goal of deep learning is to learn the feature hierarchies with increasing levels of abstraction. Each layer of deep architecture transforms input features into a higher-level one. In this thesis, we call the model of deep learning a *deep model* as contrasted with shallow models. Consider for example the task of interpreting an image (LeCun et al., 2010). Good internal structure is established by meaningful and invariant information from raw input (Figure 4.1). Raw pixels are assembled into edges, edges compose textons, textons compose motifs, motifs compose parts and parts are assembled into a complete unit object. Speech and natural language processing data have similar hierarchical compositionality (LeCun and Ranzato, 2013).

Definition of the *depth* of a deep model is ambiguous here. In LeCun and

Ranzato (2013), they give several examples of non-deep models to better comprehend the depth of a deep model. The kernel methods and support vector machines (SVM) are not a deep model. These have kernels in the first layer and the second layer is a linear function. The first layer is trained by using the samples as templates for the kernel functions. Classification trees and 2-layer models are not deep. There is no hierarchy of features, therefore all decisions are made in the input space. Also, neural networks with 1 hidden layer are not a deep model.

4.2 Deep learning algorithms

The deep learning algorithms are categorized according by several criteria as follows:

Training protocol: supervised , unsupervised, hybrid

In the supervised deep learning algorithms, target labels of data are always available and the parameters are initialized randomly and usually back-propagation with stochastic gradient descent is used for training. The neural networks, recursive neural networks (RNN) and convolutional neural networks (CNN) belong to this. Many applications of image and speech recognition have used this algorithm.

The deep autoencoders, the deep belief networks (DBN) and the deep Boltzmann machines (DBM) are the unsupervised deep learning algorithms. In this case, no information about the target class is available. DBN and DBM train each single-layer separately in an unsupervised way.

Pre-training and supervised fine-tuning are used with the unsupervised method to improve the performance in the hybrid method. In practice, when we only have very few labeled samples, this yields good results.

Learning model: neural networks, probability models, hybrid

Neural networks which have many hidden layers are the deep neural networks (DNN). The deep autoencoders, CNN and RNN are special types of the DNN. Survey propagation (SP) and non-parametric Bayesian (Bayes-NP) method are probability models.

In the hybrid method, DNN can be used to generate samples by sampling from the networks. DBN, DBM and generalized denoising autoencoders belong to this method and are thus generative models.

Processing direction: feed-forward pass, feed-back pass, bi-directional pass

Figure 4.2 shows three types of the deep learning model according to processing directions. Figure 4.2.(a) is a feed-forward pass. Multilayer perceptrons (MLP) and CNN belong to this. This is typical structure of neural networks.

Figure 4.2.(b) is a feed-back pass. Stacked sparse coding and deconvolutional neural networks belong to this. Even though this architecture is quite novel, their solution is based on the connection between traditional ways. For instance, stack sparse coding repeatedly trains separable layers using predictive sparse coding then modifies the trained layers by using autoencoders for each layer.

Figure 4.2.(c) is a bi-directional pass. DBM and stacked autoencoders belong to this. Two neighbouring layers are undirected. This method starts to learn with the lowest level (i.e., input node) and stack upwards.

Of these methods, we describe details of the two simple and effective methods:

- *Deep autoencoder*: A prominent example of the unsupervised deep learning method. we discuss its application in speech processing.

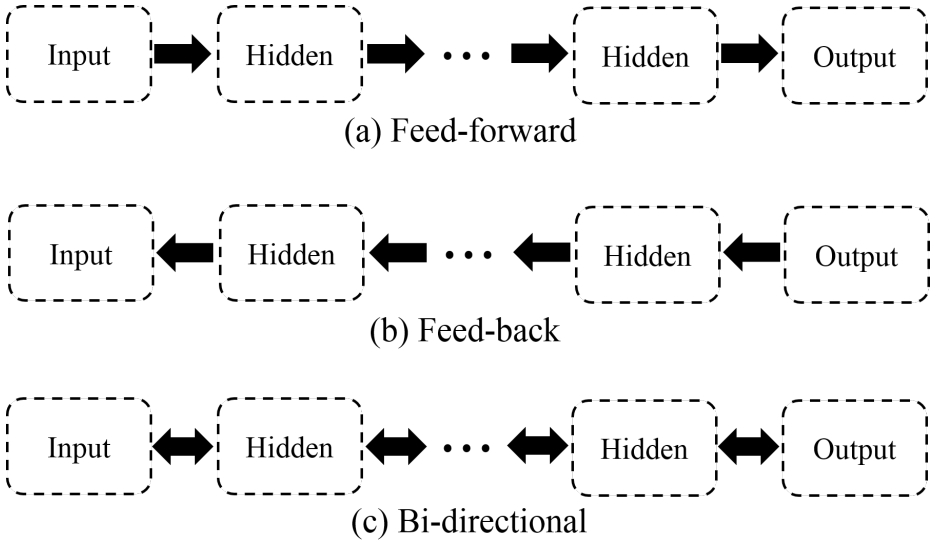


Figure 4.2: Three types of the deep learning method according to processing directions.

- *CNN*: A major example of the supervised deep learning method. We discuss its application in computer vision.

4.2.1 Deep autoencoder

An autoencoder consists of an input layer, one or more hidden layers that learn higher-level abstraction of features, and an output layer which matches the input layer for reconstruction. When the number of hidden layers is greater than one, the autoencoder is considered to be the deep autoencoder. The deep autoencoder (Bengio et al., 2007; Hinton and Salakhutdinov, 2006) is a discriminative DNN whose output targets are the data input itself rather than label information, so this is an unsupervised learning algorithm. The deep autoencoder is often exploited for learning effective representations of the original data.

General shallow autoencoders are trained using back-propagation with stochas-

tic gradient descent. Although this method is simple and reasonably effective for sallow models, there are problems when using back-propagation to train DNN that has many hidden layers, such as the vanishing gradient problem, slow leaning and poor performance, especially when only a limited amount of training data is available. However, these problems can be alleviated by pre-training each layer as a general autoencoder (Hinton et al., 2006).

In speech processing

Deep learning has been getting the spotlight in speech recognition recently. Here we introduce an application for feature learning of speech audio files (Deng et al., 2010). This work exploits a deep autoencoder for extracting binary speech codes from the raw speech spectrogram data. An extracted binary code can be used in speech recognition and information retrieval.

Firstly, an undirected generative model called a Gaussian-Bernoulli restricted Boltzmann machine (RBM) is built. This model has one visible layer of variables with Gaussian noise and one hidden layer of binary latent variables. After learning the first Gaussian-Bernoulli RBM, its hidden units are treated as input for another Bernoulli-Bernoulli RBM. The connection of two RBMs constitute a DBN (Hinton et al., 2006). This is the last step of pre-training. Then this DBN is treated as the deep autoencoder with three layers. This deep autoencoder is fine-tuned using error back-propagation to minimize the reconstruction error. After learning is complete, spectrogram data can be encoded and reconstructed to binary codes by using the trained deep autoencoder.

4.2.2 Convolutional neural networks

The human vision field requires the construction of good internal representations for visual perception (Hubel and Wiesel, 1962). Therefore, constructing and pro-

ducing internal representation is directly connected to extract suitable features. CNN (Lee et al., 2009, 2011) was inspired by such biological visual field processes (Matsugu et al., 2003). This consists of typically six or seven layers of alternate succession of convolution layers and sub-sampling layers with a MLP or DNN on top. Each convolution layer observes particular features at every different location in the input feature map and shares many weights. The sub-sampling layer subsamples the output of the convolution layer and reduces the data rate from the layer below. Hence the output feature map of the convolution layer with appropriately chosen sub-sampling schemes acquires the translational invariance.

CNN is one of the few models that can be trained purely in an supervised way and back-propagation with stochastic gradient descent is usually used for training.

In computer vision

CNN has been found very effective and been commonly used in computer vision, image recognition and video recognition (Ciresan et al., 2012; Dean et al., 2012; Le, 2013). Some say that CNN is the first truly successful deep learning model in a robust manner (Arel et al., 2010) and has won several competitions (Krizhevsky et al., 2012; Goodfellow et al., 2014).

Here we introduce the most remarkable application. In 2012 ImageNet LSVRC competition, 1.2 million high-resolution training images and 150,000 test images are given. Then the task is to get the best classification performance. The CNN model described in Krizhevsky et al. (2012) achieved the best performance. The used CNN has 60 million weights, 650,000 neurons, five convolution layers together with max-pooling sub-sampling layers and two fully-connected DNN on the top. Three important factors contribute to the success. The first is a fast hardware. As mentioned above, improved chip processing abilities (e.g., graphical processing units) enable them to train this very large CNN. The second is the big

dataset. The last factor is powerful regularizer “dropout” (Hinton et al., 2012b).

Chapter 5

Density ratio estimation using a deep model

This section describes an extended density ratio estimation method based on deep convolutional neural networks. We demonstrate the performance of our proposed method in inlier-based outlier detection.

5.1 Introduction

Recently, it was shown (Sugiyama et al., 2012b) that, through the *ratio of probability density functions*, various statistical data analysis paradigms such as outlier detection, non-stationarity adaptation, mutual information estimation, and conditional probability estimation can be efficiently handled in a unified manner. The key idea of this density ratio approach is that, by directly estimating the density ratio, a difficult task of density estimation can be avoided.

So far, various density ratio estimators have been proposed. The simplest approach is to use *logistic regression* to discriminate samples from two distributions (Bickel et al., 2007). *Kernel mean matching* (Gretton et al., 2009) uses the

Hilbert-space embedding of probability distributions to directly approximate the values of the density ratio at data points. The *Kullback-Leibler importance estimation procedure* (KLIEP) fits a density ratio model to data under the log-loss (Sugiyama et al., 2008; Nguyen et al., 2010). *Least-squares importance fitting* (LSIF) (Kanamori et al., 2009) uses the squared-loss to fit a density ratio model to data. Furthermore, all the above methods can be interpreted as fitting a density ratio model to data under the *Bregman divergence* (Sugiyama et al., 2012a).

Among these direct density ratio estimators, an unconstrained version of LSIF (uLSIF) with a kernel density-ratio model was demonstrated to be highly useful in terms of both accuracy (Kanamori et al., 2012) and computational efficiency (Kanamori et al., 2013). For that reason, uLSIF-based machine learning algorithms have been successfully used in solving various machine learning tasks (Sugiyama, 2012; Sugiyama et al., 2013b,c).

On the other hand, recent studies in pattern recognition demonstrated that *deep architecture* tends to perform better than kernel models (Larochelle et al., 2007). In particular, a *convolutional neural network* (CNN) is demonstrated to be an excellent model of images (Krizhevsky et al., 2012; Jarrett et al., 2009; Lee et al., 2009; Turaga et al., 2010; Ngiam et al., 2010), which is motivated by a biological brain (Fukushima, 1980).

The objective of this chapter is to use the CNN model in density ratio estimation. The idea of the proposed method is that trained CNNs with the uLSIF criterion are able to abstract the prior knowledge of internal structure of data such as images, and this extracted feature can improve the performance.

To the best of our knowledge, this is the first attempt to apply deep learning to density ratio estimation, and we develop a gradient-based training algorithm under the squared-loss. We apply the CNN-based density ratio estimator to *inlier-based outlier detection* of images (Hido et al., 2011; Song et al., 2009), and demonstrate

that the proposed method outperforms existing approaches.

The remainder of this chapter is organized as follows. We first review the kernel-based uLSIF method in Section 5.2. Then, we derive a density ratio estimation algorithm for CNNs in Section 5.4, and its experimental performance is investigated in Section 5.5. Finally, we conclude in Section 5.6.

5.2 Direct Density Ratio Estimation by uLSIF

In this section, we review the uLSIF method (Kanamori et al., 2009).

5.2.1 Problem Formulation

Suppose we are given independent and identically distributed training samples $\{\mathbf{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ from training distribution with density $p_{\mathrm{tr}}(\mathbf{x})$ and test samples $\{\mathbf{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ from test distribution with density $p_{\mathrm{te}}(\mathbf{x})$ on some data domain $\mathcal{D} \subset \mathbb{R}^d$. The objective is to estimate the *density ratio*,

$$r(\mathbf{x}) = \frac{p_{\mathrm{tr}}(\mathbf{x})}{p_{\mathrm{te}}(\mathbf{x})},$$

from $\{\mathbf{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{\mathbf{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$.

A naive approach is to first separately estimate $p_{\mathrm{tr}}(\mathbf{x})$ from $\{\mathbf{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $p_{\mathrm{te}}(\mathbf{x})$ from $\{\mathbf{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, and then compute the ratio of estimated densities. However, such a two-step approach does not perform well because the estimation error incurred in the first density estimation step can be magnified in the second step of computing their ratio (Sugiyama et al., 2012b). Below, a direct density ratio estimator that does not involve density estimation is reviewed.

5.2.2 The uLSIF Criterion

Let $r_{\boldsymbol{\alpha}}(\mathbf{x})$ be a model of the density ratio $r(\mathbf{x})$, where $\boldsymbol{\alpha}$ denotes a parameter. The parameter $\boldsymbol{\alpha}$ is determined so that the following squared error is minimized:

$$\begin{aligned} J_0(\boldsymbol{\alpha}) &= \int \Big(r_{\boldsymbol{\alpha}}(\mathbf{x}) - r(\mathbf{x})\Big)^2 p_{\mathrm{te}}(\mathbf{x})\mathrm{d}\mathbf{x} \\ &= \int r_{\boldsymbol{\alpha}}(\mathbf{x})^2 p_{\mathrm{te}}(\mathbf{x})\mathrm{d}\mathbf{x} - 2\int r_{\boldsymbol{\alpha}}(\mathbf{x}) p_{\mathrm{tr}}(\mathbf{x})\mathrm{d}\mathbf{x} \\ &\quad + \int r(\mathbf{x}) p_{\mathrm{tr}}(\mathbf{x})\mathrm{d}\mathbf{x}, \end{aligned}$$

where the last term is a constant so can be ignored. The first two terms are denoted by J:

$$J(\boldsymbol{\alpha}) = \int r_{\boldsymbol{\alpha}}(\mathbf{x})^2 p_{\mathrm{te}}(\mathbf{x})\mathrm{d}\mathbf{x} - 2\int r_{\boldsymbol{\alpha}}(\mathbf{x}) p_{\mathrm{tr}}(\mathbf{x})\mathrm{d}\mathbf{x},$$

which is empirically approximated by

$$\frac{1}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}} r_{\boldsymbol{\alpha}}(\mathbf{x}_j^{\mathrm{te}})^2 - \frac{2}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}} r_{\boldsymbol{\alpha}}(\mathbf{x}_i^{\mathrm{tr}}). \tag{5.1}$$

5.2.3 uLSIF for Kernel Model

Let us consider the following kernel density ratio model:

$$r_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{\ell=1}^{n_{\mathrm{tr}}} \alpha_\ell K(\mathbf{x}, \mathbf{x}_\ell^{\mathrm{tr}}),$$

where $K(\mathbf{x}, \mathbf{x}')$ is a kernel function such as the *Gaussian kernel*:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \quad \text{for } \sigma > 0.$$

Then the uLSIF criterion (5.1), enhanced with the ℓ_2-regularizer, can be expressed as

$$\widehat{J}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \widehat{\mathbf{G}}\boldsymbol{\alpha} - 2\widehat{\mathbf{h}}^\top \boldsymbol{\alpha} + \lambda\|\boldsymbol{\alpha}\|^2,$$

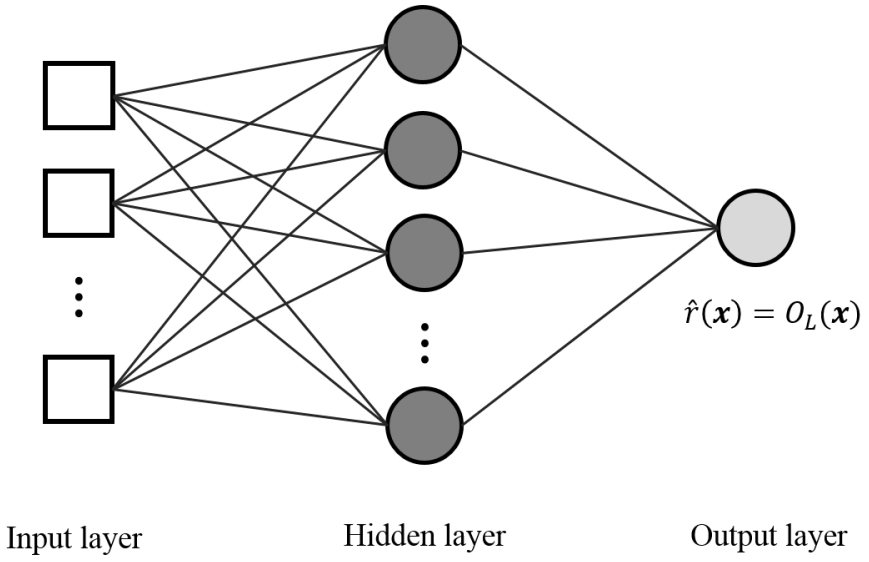


Figure 5.1: Three-layer MLP.

where $\lambda > 0$ is the regularization parameter and

$$\widehat{G}_{\ell,\ell'} = \frac{1}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} K(\mathbf{x}_j^{\mathrm{te}}, \mathbf{x}_\ell^{\mathrm{te}}) K(\mathbf{x}_j^{\mathrm{te}}, \mathbf{x}_{\ell'}^{\mathrm{te}}),$$
$$\widehat{h}_\ell = \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} K(\mathbf{x}_i^{\mathrm{tr}}, \mathbf{x}_\ell^{\mathrm{te}}).$$

Then the minimizer is given analytically as

$$\arg\min_{\boldsymbol{\alpha}} \widehat{J}(\boldsymbol{\alpha}) = (\widehat{\mathbf{G}} + \lambda \mathbf{I})^{-1} \widehat{\mathbf{h}},$$

where $\mathbf{I}$ denotes the identity matrix.

5.3 uLSIF for Multilayer Perceptron

In this section, to describe uLSIF for CNN, we will begin by describing how to apply the uLSIF criterion (5.1) to the multilayer perceptron (MLP).

5.3.1 MLP

An MLP is a neural network model that consists of multiple layers of nodes: input, hidden and output layer. Each layer fully connected to the next one with weighting, bias and nonlinear activation function. Usually a MLP consists of three or more layers. Figure 5.1 denotes a simple three-layers MLP and we use this model for density ratio estimation.

Hidden layer

In the hidden layer, each node is a neuron and its values are not observed in the inputs. This neuron takes input from the input layer or the previous hidden layer, then compute the activation via an activation function. More specifically, a vector of an output of the lth hidden layer $O_l(\mathbf{x})$ for input $O_l(\mathbf{x})$ is given by

$$\begin{aligned}\mathbf{O}_l(\mathbf{x}) &= f(\mathbf{z}_\mathbf{l}^{\mathrm{MLP}}(\mathbf{x})),\\ \mathbf{z}_\mathbf{l}^{\mathrm{MLP}}(\mathbf{x}) &= \mathbf{W}_{l-1}^{\mathrm{MLP}}\mathbf{O}_{l-1}(\mathbf{x}) + b_l^{\mathrm{MLP}},\end{aligned}$$

where $\mathbf{W}_{l-1}^{\mathrm{MLP}}$ is the weight associate with the connection between the current hidden layer and previous layer and b_l^{MLP} is the bias. Also, $\mathbf{z}_\mathbf{l}^{\mathrm{MLP}}(\mathbf{x})$ denotes the total weighted sum of inputs in layer l, including the bias term. f represents the activation function. In this thesis we will choose the sigmoid function for the hidden layer:

$$f(x) = (1 + e^{-x})^{-1}.$$

Output layer

The output O_L of the MLP for input vector $\mathbf{x}$ is represented as

$$\begin{aligned}O_L(\mathbf{x}) &= h(z_L^{\mathrm{MLP}}(\mathbf{x})),\\ z_L^{\mathrm{MLP}}(\mathbf{x})) &= \mathbf{W}_{L-1}^{\mathrm{MLP}}(\mathbf{x}))\mathbf{O}_{L-1}(\mathbf{x})) + b_L^{\mathrm{MLP}},\end{aligned}$$

where L is the last layer, $\mathbf{W}_{L-1}^{\mathrm{MLP}}$ is a connection parameter, b_L^{MLP} is the bias parameter and $\mathbf{O}_{L-1}$ is the output of the last hidden layer. In the output layer, we will use the *softplus* function as the activation function:

$$h(x) = \log(1 + e^x).$$

This is a smooth approximation to the *rectifier* and biologically more plausible and does not occur the vanishing gradient problem.

5.3.2 Density ratio estimation with MLP

In general, an MLP is trained by the gradient descent algorithm. Thus we train the MLP model with the uLSIF criterion by the gradient descent method for a pair of samples $(\mathbf{x}^{te}, \mathbf{x}^{tr})$ to obtain a local minimizer:

$$\dot{J}(\{\mathbf{W}_L^{\mathrm{MLP}}\}, \{b_L^{\mathrm{MLP}}\}) = O_L(\mathbf{x}^{\mathrm{te}})^2 - 2O_L(\mathbf{x}^{\mathrm{tr}}).$$

The gradients of $\dot{J}$ with respect to $\mathbf{W}_L^{\mathrm{MLP}}$ and b_L^{MLP} are given by

$$\frac{\partial \dot{J}}{\partial \mathbf{W}_L^{\mathrm{MLP}}} = \delta_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}})(\mathbf{O}_{L-1}(\mathbf{x}^{\mathrm{te}}))^\top - \delta_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}})(\mathbf{O}_{L-1}(\mathbf{x}^{\mathrm{tr}}))^\top,$$
$$\frac{\partial \dot{J}}{\partial b_L^{\mathrm{MLP}}} = \delta_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}}) - \delta_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}}),$$

where

$$\delta_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}}) = 2O_L(\mathbf{x}^{\mathrm{te}})h'(z_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}})),$$
$$\delta_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}}) = 2h'(z_L^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}})).$$

The gradients of weight and bias of the lth hidden layer are given by

$$\frac{\partial \dot{J}}{\partial \mathbf{W}_l^{\mathrm{MLP}}} = \delta_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}})(\mathbf{O}_{l-1}(\mathbf{x}^{\mathrm{te}}))^\top - \delta_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}})(\mathbf{O}_{l-1}(\mathbf{x}^{\mathrm{tr}}))^\top,$$
$$\frac{\partial \dot{J}}{\partial b_l^{\mathrm{MLP}}} = \delta_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}}) - \delta_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}}),$$

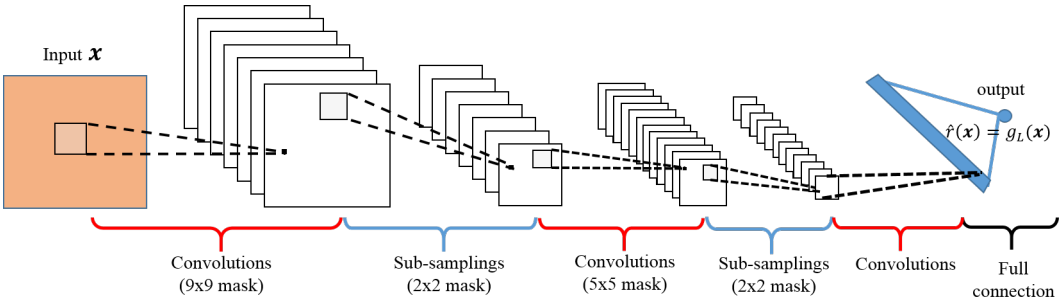


Figure 5.2: CNN.

where

$$\delta_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}}) = ((\mathbf{W}_l^{\mathrm{MLP}})^\top \delta_{l+1}^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}})) \bullet f'(\mathbf{z}_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{te}})),$$
$$\delta_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}}) = ((\mathbf{W}_l^{\mathrm{MLP}})^\top \delta_{l+1}^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}})) \bullet f'(\mathbf{z}_l^{\mathrm{MLP}}(\mathbf{x}^{\mathrm{tr}})).$$

"$\bullet$" denotes the element-wise product.

5.4 uLSIF for CNN

In this section, we apply the uLSIF criterion (5.1) to a *convolutional neural network* (CNN).

5.4.1 CNN

A CNN is a model for 2-dimensional images and consists of multiple layers (Figure 5.2): the input layer, alternate succession of convolution layers and sub-sampling layers, fully connected networks, and the output layer.

Convolution Layer

In the convolution layer, the output of the previous layer is convolved with a mask and put through the activation function to form the output feature map. More

specifically, a matrix of an output feature map of the lth convolution layer $\mathbf{g}_l^u(\mathbf{x})$ for input feature maps $\mathbf{g}_{l-1}^v(\mathbf{x})$ is given by

$$\mathbf{g}_l^u(\mathbf{x}) = f(\mathbf{z}_l^u(\mathbf{x})),$$
$$\mathbf{z}_l^u(\mathbf{x}) = \sum_{v \in \mathcal{M}_v} \mathbf{g}_{l-1}^v(\mathbf{x}) * \mathbf{k}_l^{uv} + b_l^u,$$

where $\mathcal{M}_v$ is a selection of the input feature maps, f represents the sigmoid function,

$$f(x) = (1 + e^{-x})^{-1}.$$

$*$ denotes the convolution operator, b_l^u is a bias parameter, and $\mathbf{k}_l^{uv}$ denotes a mask.

Sub-Sampling Layer

The sub-sampling layer treats each feature map separately and produces sub-sampled versions of the input maps. More formally,

$$\mathbf{g}_l^u(\mathbf{x}) = \mathrm{down}(\mathbf{g}_{l-1}^u(\mathbf{x})) + b_l^u,$$

where "down" denotes a sub-sampling function and b_l^u is the bias parameter. The average value over the neighborhood is computed for each feature map in our method. This reduced-resolution output feature map is robust to variation and noise in the input feature map.

Fully-Connected Layer

The output $g_L(\mathbf{x}_j)$ of the fully-connected layer for input vector $\mathbf{g}_{L-1}(\mathbf{x}_j)$ is represented as

$$g_L(\mathbf{x}) = h(z_L(\mathbf{x})),$$
$$z_L(\mathbf{x}) = \mathbf{W}_L \mathbf{g}_{L-1}(\mathbf{x}) + b_L,$$

where L is the last layer, $\mathbf{W}_L$ is a connection parameter, b_L is the bias parameter and $\mathbf{g}_{L-1}(\mathbf{x})$ is a reshaped output of the last convolution layer. h is the *softplus* function ,

$$h(x) = \log(1 + e^x),$$

which is a smooth approximation to the *rectifier* $\max(0, x)$ (Nair and Hinton, 2010).

5.4.2 Density Ratio Estimation with CNN

The kernel-based uLSIF is a computationally very efficient method. Using a deep CNN, however, would cast huge demand to the computation power during the training process. To economize on the computation cost at every iteration, the stochastic gradient descent algorithm is used in our implementation. Thus we train the CNN model with the uLSIF criterion (5.1) by the *stochastic gradient method* for a pair of samples $(\mathbf{x}^{\mathrm{te}}, \mathbf{x}^{\mathrm{tr}})$ to obtain a local minimizer:

$$\widetilde{J}(\{\mathbf{W}_L\}, \{b_L\}) = g_L(\mathbf{x}^{\mathrm{te}})^2 - 2g_L(\mathbf{x}^{\mathrm{tr}}). \tag{5.2}$$

The gradients of $\widetilde{J}$ with respect to $\mathbf{W}_L$ and b_L are given by

$$\frac{\partial \widetilde{J}}{\partial \mathbf{W}_L} = \delta_L(\mathbf{x}^{\mathrm{te}})(\mathbf{g}_{L-1}(\mathbf{x}^{\mathrm{te}}))^\top - \delta_L(\mathbf{x}^{\mathrm{tr}})(\mathbf{g}_{L-1}(\mathbf{x}^{\mathrm{tr}}))^\top,$$
$$\frac{\partial \widetilde{J}}{\partial b_L} = \delta_L(\mathbf{x}^{\mathrm{te}}) - \delta_L(\mathbf{x}^{\mathrm{tr}}),$$

where

$$\delta_L(\mathbf{x}^{\mathrm{te}}) = 2g_L(\mathbf{x}^{\mathrm{te}})h'(z_L(\mathbf{x}^{\mathrm{te}})),$$
$$\delta_L(\mathbf{x}^{\mathrm{tr}}) = 2h'(z_L(\mathbf{x}^{\mathrm{tr}})).$$

The gradient of the bias for a feature map of the lth convolution layer is given by summing all the elements in each error term:

$$\frac{\partial \widetilde{J}}{\partial b_l^u} = \sum_{n,m} (\boldsymbol{\delta}_l^u(\mathbf{x}^{te}))_{n,m} - (\boldsymbol{\delta}_l^u(\mathbf{x}^{tr}))_{n,m}.$$

The gradient for the mask weight of the lth convolution layer is given by

$$\frac{\partial \widetilde{J}}{\partial \mathbf{k}_l^{uv}} = \sum_{n,m} (\boldsymbol{\delta}_l^u(\mathbf{x}^{te}))_{n,m} (\mathbf{M}_l^v(\mathbf{x}^{te}))_{n,m} - (\boldsymbol{\delta}_l^u(\mathbf{x}^{tr}))_{n,m} (\mathbf{M}_l^v(\mathbf{x}^{tr}))_{n,m},$$

where

$$\mathbf{M}_l^v(\mathbf{x}) = \mathbf{g}_{l-1}^v(\mathbf{x}) \bullet \mathbf{k}_l^{uv},$$

The error terms in the convolution layer are expressed as

$$\boldsymbol{\delta}_l^u(\mathbf{x}^{te}) = f'(\mathbf{z}_l^u(\mathbf{x}^{te})) \bullet \mathrm{up}(\boldsymbol{\delta}_{l+1}^u(\mathbf{x}^{te})),$$
$$\boldsymbol{\delta}_l^u(\mathbf{x}^{tr}) = f'(\mathbf{z}_l^u(\mathbf{x}^{tr})) \bullet \mathrm{up}(\boldsymbol{\delta}_{l+1}^u(\mathbf{x}^{tr})).$$

Here, "up" denotes the up-sampling operation, which can be implemented with the Kronecker product. At the last convolution layer, an error term includes connection parameters of the fully connected layer:

$$\boldsymbol{\delta}_l^u(\mathbf{x}^{te}) = \mathbf{W}_L \left(f'(\mathbf{z}_l^u(\mathbf{x}^{te})) \bullet \mathrm{up}(\boldsymbol{\delta}_{l+1}^u(\mathbf{x}^{te})) \right),$$
$$\boldsymbol{\delta}_l^u(\mathbf{x}^{tr}) = \mathbf{W}_L \left(f'(\mathbf{z}_l^u(\mathbf{x}^{tr})) \bullet \mathrm{up}(\boldsymbol{\delta}_{l+1}^u(\mathbf{x}^{tr})) \right).$$

The gradient of the bias for the lth sub-sampling layer is given by

$$\frac{\partial \widetilde{J}}{\partial b_l^u} = \sum_{n,m} (\boldsymbol{\delta}_l^u(\mathbf{x}^{te}))_{n,m} - (\boldsymbol{\delta}_l^u(\mathbf{x}^{tr}))_{n,m},$$

where

$$\boldsymbol{\delta}_l^u(\mathbf{x}^{te}) = \boldsymbol{\delta}_{l+1}^u(\mathbf{x}^{te}) * \mathrm{rot}(\mathbf{k}_{l+1}^u),$$
$$\boldsymbol{\delta}_l^u(\mathbf{x}^{tr}) = \boldsymbol{\delta}_{l+1}^u(\mathbf{x}^{tr}) * \mathrm{rot}(\mathbf{k}_{l+1}^u).$$

Here "rot" denotes the rotation operator which rotates the mask by 180 degrees.

5.5 Experiments

In this section, the proposed CNN-based uLSIF is compared with the kernel-based uLSIF and the kernel-based KLIEP in inlier-based outlier detection. Note that, in Hido et al. (2011), the kernel-based uLSIF and KLIEP were demonstrated to outperform other outlier detection methods such as the *one-class support vector machine* (Schölkopf et al., 2001) and the *local outlier factor* (Breunig et al., 2000).

5.5.1 Inlier-Based Outlier Detection

The objective of inlier-based outlier detection is to find outliers in a test data set $\{\mathbf{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ given a training data set $\{\mathbf{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ which are known to be inliers. Here, following Hido et al. (2011), the problem of inlier-based outlier detection is formulated as the problem of estimating the density ratio,

$$r(\mathbf{x}) = \frac{p_{\mathrm{tr}}(\mathbf{x})}{p_{\mathrm{te}}(\mathbf{x})},$$

from $\{\mathbf{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{\mathbf{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$. Given that outliers tend to exist in regions with small inlier density $p_{\mathrm{tr}}(\mathbf{x})$, the density ratio $r(\mathbf{x})$ tends to take a small value if $\mathbf{x}$ is an outlier. Outliers are hard to universally define. By contrast, inlier samples are often stable and available abundantly in practice. Therefore the setting of inlier-based outlier detection would be more practical than the (semi) supervised setting.

5.5.2 Experimental Setup

We use a CNN with 5 layers, i.e., the first 4 layers contain two alternate convolution layers and sub-sampling layers, followed by an additional convolution layer for making a vector input which is given to the final fully-connected layer (see Figure 5.2 again). The size of input images depends on the data set. The first layer has 6 convolution masks of size 9×9 with a stride of a pixel for image of

size 32×32 (for smaller images, we use 5×5 convolution masks). In the following sub-sampling layer, each unit calculates the average over the output from neighbors of 2×2 pixels. The sub-sampling layer reduces the resolution of output of the previous layer and the reduced feature is robust to small variations. The second convolution layer has 12 convolution masks of size 5×5 (for smaller images, we use 3×3 convolution masks). Sub-sampling is applied to its output again. The third convolution layer reshapes 12 feature maps generated by the previous sub-sampling layer and provide the vector input to the final fully-connected layer. We use the sigmoid activation function for convolution layers and the ReLU activation function for the last layer. At the learning stage, all parameters are optimized through the stochastic gradient method. We use the constant learning rate 0.1 for all layers. All the model parameters such as the size of the convolution mask, the number of layers and learning rate are tuned in the same way as the previous studies (LeCun et al., 1998, 2010).

We use the MNIST handwritten digit dataset, the USPS handwritten digit dataset, the PIE face image dataset, and the CIFAR-10 image classification dataset. We select one class (e.g., `airplane` in CIFAR-10) and all training samples in this class are regarded as a training set in inlier-based outlier detection. Then all evaluation samples of a selected class and a fraction ρ of evaluation samples from another class (e.g., `bird`) are regarded as a test set in inlier-based outlier detection. Namely, we regard samples in a selected class as inliers and samples in another class as outliers. The *area under the ROC curve* (AUC) value is used as an error metric. Kernel-based uLSIF and KLIEP were designed to use selected 100 test input points as Gaussian centers randomly.

5.5.3 Results

Experimental results are exhibited below.

MNIST and USPS

The MNIST dataset contains hand-written digit images in gray-scale: a training set of 60,000 images and a test set of 10,000 images. Each image consists of 1,024 ($= 32 \times 32$) pixels. The USPS hand-written digit dataset contains 9,298 gray-scale images of size 16×16 which are spilt into a training set of 7,291 images and a test set of 2,007 images. Both datasets have 10 class labels which are integers between 0 and 9. We tried to select similar digits for the inlier class and outlier class as much as possible in MNIST and USPS, because we want to make the problem more difficult.

The mean AUC values and the standard deviations over 20 trials for MNIST and USPS are summarized in Table 5.2, and the mean MSE (mean squared error) values and the standard deviations over 20 trials for MNIST and USPS are shown in Table 5.3. Figure 5.3 depicts the ROC curves for the experiment with inlier class 9 and outlier class 8. The results show that the proposed CNN-based uLSIF works significantly better than kernel-based KLIEP and uLSIF.

Table 5.1 shows the computation time of each algorithm for the USPS dataset (8 and 3). As we mentioned above, kernel-based uLSIF has the best performance in terms of computational efficiency. Mean AUC values (with standard deviations in parentheses) over 50 trials for the PASCAL VOC dataset. The best method in terms of the mean AUC and comparable methods according to the *t-test* at the significance level 5% are specified by bold face.

Table 5.1: Computation time of USPS dataset (8 and 3)

Method	uLSIF (CNN)	uLSIF (kernel)	KLIEP (kernel)
Computation time (sec)	833.4969	0.0028	0.0164

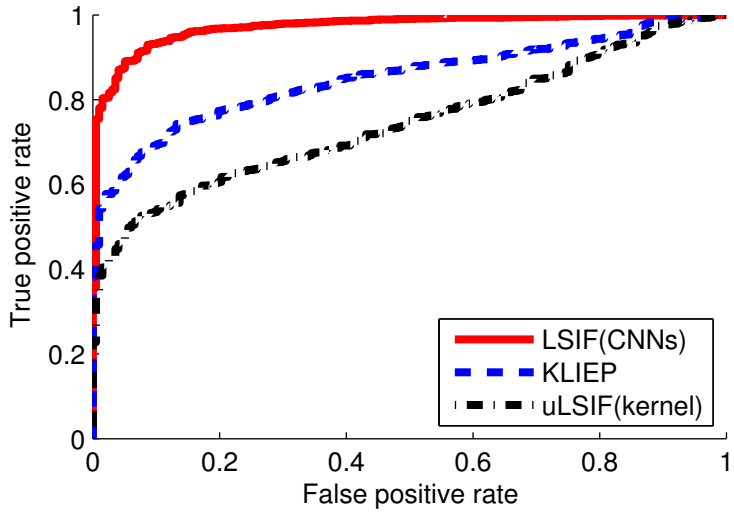


Figure 5.3: ROC curves of MNIST 9 and 8 ($\rho = 0.05$).

PIE

Next, we consider outlier detection on the PIE face image dataset. Face images contain more complex objects than digits and data variability caused by posing variations, illumination conditions, and facial expressions is higher. These facts make outlier detection more challenging for the PIE dataset. The original PIE dataset contains 41,368 color face images with 68 individuals, each person is under 13 different poses, 43 different illumination conditions, and with 4 different expressions. We use the processed PIE data for face recognition (He et al., 2005), which chooses five near-frontal poses and uses all the images under different illuminations and expressions; thus 170 images are given for each individual. Each individual is treated as a class, so the dataset has 68 class labels which are integers between 0 and 67. We select one class (e.g., class `7`) and all training samples in this class are regarded as a training set in inlier-based outlier detection. Then all evaluation samples of the selected class and a fraction ρ of evaluation samples from another class (e.g., class `33`) are regarded as a test set in inlier-based outlier

detection. The inlier class and outlier class are selected randomly. We allocate randomly chosen 5,780 images for training samples and the remaining 5,780 images for test samples in this experiment. All the images are manually aligned and cropped. The cropped images are 32×32 pixels with 256 gray-levels per pixel.

The mean AUC values and the standard deviations over 20 trials are summarized in Table 5.4, and Figure 5.4 depicts the ROC curves for the experiment with inlier class 7 and outlier class 33. The results show that our proposed method achieves better performance for all classes than the kernel-based KLIEP and uLSIF.

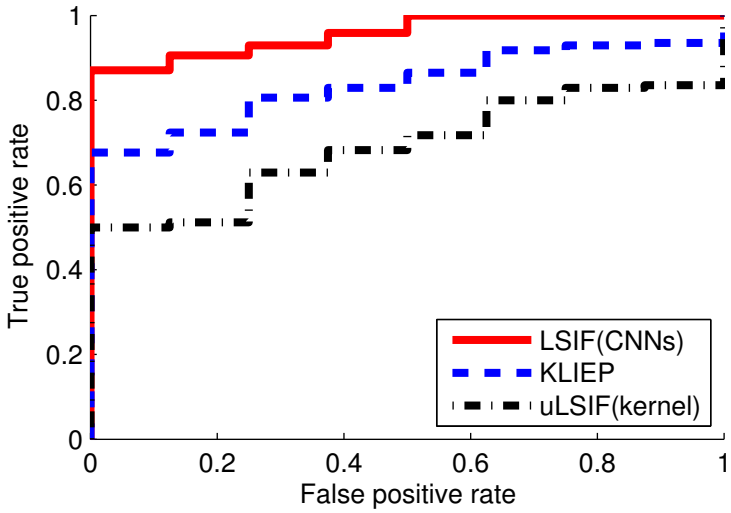


Figure 5.4: ROC curves of PIE 7 and 33 ($\rho = 0.05$).

CIFAR-10

Finally, we use the CIFAR-10 image classification dataset, which consists of 32×32 color images in 10 classes (`airplane`, `car`, `bird`, `cat`, `deer`, `dog`, `frog`, `horse`, `ship`, and `truck`) with 6,000 images per class. The inlier class and outlier class are selected randomly for CIFAR-10. Since images in CIFAR-10

contain more data variability and noise than those in the PIE dataset, outlier detection is expected to be even harder. There are 50,000 training images and 10,000 test images, and we convert color images to gray-scale.

The mean AUC values and the standard deviations over 20 trials are summarized in Table 5.5, and Figure 5.5 depicts the ROC curves for the experiment with inlier class `car` and outlier class `cat`. The results show that the proposed CNN-based uLSIF performs excellently.

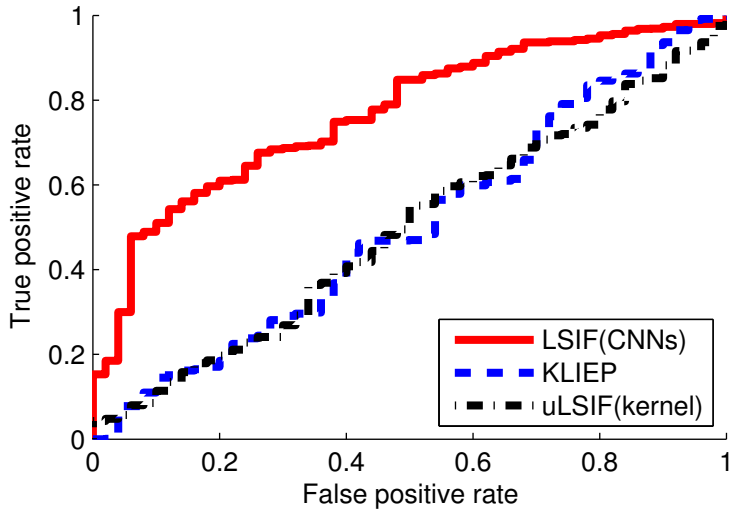


Figure 5.5: ROC curves of CIFAR-10 `car` and `cat` ($\rho = 0.05$).

5.6 Discussion

We proposed to use CNNs in least-squares direct density-ratio estimation, uLSIF, and demonstrated its usefulness in inlier-based outlier detection of images.

In order to investigate the change in AUC values for different number of kernel function, we executed uLSIF and KLIEP with different number of kernel centers for the CIFAR-10 dataset (`car` and `cat`). As shown in Figure 5.8. This shows

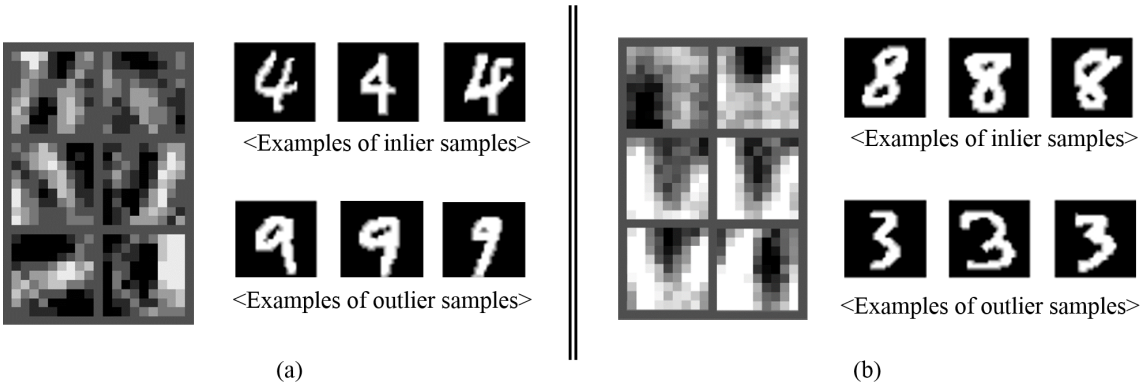


Figure 5.6: The first convolution layer masks. Left (a) depicts first convolution layer masks of MNIST experiments of class 4 and 9. Class 4 is inlier class. Right (b) depicts first convolution layer masks of MNIST experiments of class 8 and 3. Class 8 is inlier class.

that using 100 kernels is reasonable in terms of both accuracy and computation time.

The idea of the proposed method is that trained CNNs with the uLSIF criterion are able to abstract features that are robust to spatial variations and have good internal information of raw images for density ratio estimation. And this extracted feature from deep architecture CNNs can improve the performance. Through remarkable experimental results with MNIST, USPS, PIE and CIFAR-10, we have shown the CNNs based uLSIF outperforms other kernel based methods and is stable over various datasets.

Another interesting thing is that KLIEP has higher AUC values than uLSIF in all experiments despite the same kernel model is used for both. That is because the KLIEP's empirical version of the Kullback-Leibler divergence is more sensitive to outlier (Fujisawa and Eguchi, 2006; Sugiyama et al., 2013a).

Multi-layer structure of CNNs gives us a chance to look into the outlier detec-

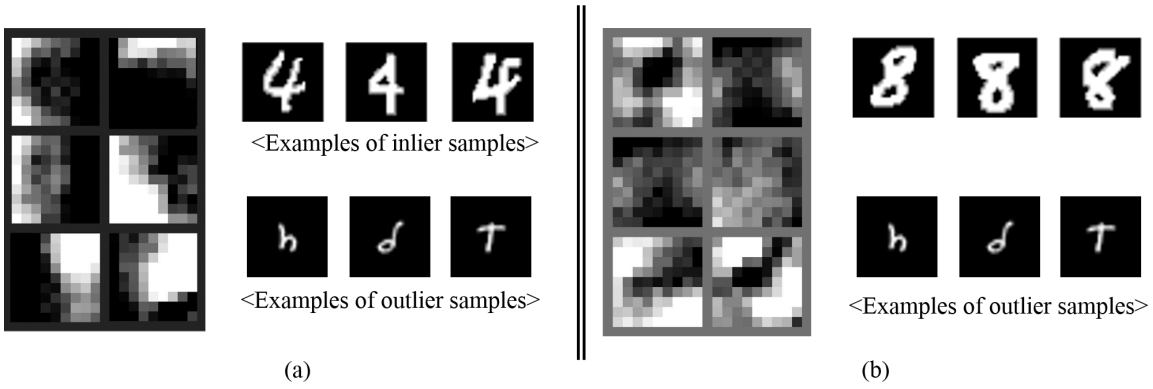


Figure 5.7: The first convolution layer masks. Left (a) depicts first convolution layer masks of MNIST experiments of class 4 and NIST SD19. Class 4 is inlier class. Right (b) depicts first convolution layer masks of MNIST experiments of class 8 and NIST SD19. Class 8 is inlier class.

tion process. Figure 5.6 displays the first convolution masks of size 9×9 learned by the proposed method for the MNIST experiment with (a) inlier class 4 and outlier class 9 and (b) inlier class 8 and outlier class 3. In (a), masks have oblique and rectilinear patterns and they seem like a part of digit 4. In contrast, in (b), masks have U-shaped and turned U-shaped patterns, which are rarely found in outlier class 3. Thus, the masks have learned to detect inliers' edges and lines at different positions and angles in the images, implying that our algorithm is translation-invariant.

We designed another experiment to see the influence of the outlier sample. We used the NIST SD19 handwritten character dataset (Grother, 1995) for outlier samples. It includes handwritten letter sample forms by 3600 writers, 810,000 character images. We selected 1000 images randomly. The resolution and color of these images are modified for the same condition as the MNIST dataset. Fig-

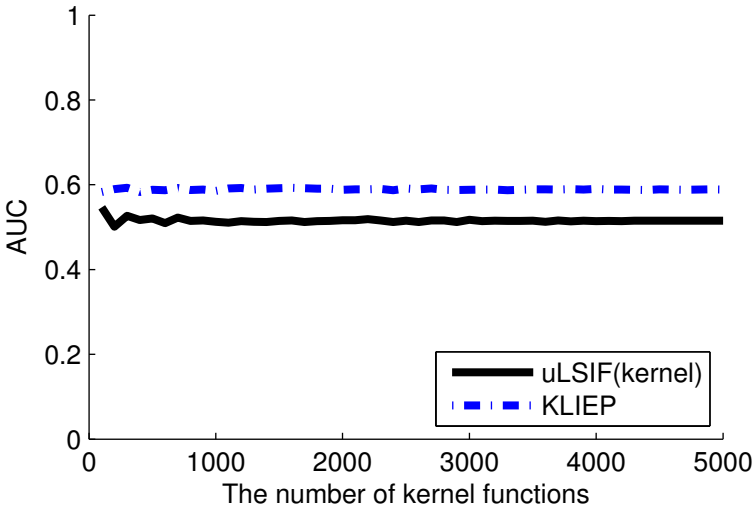


Figure 5.8: Change in AUC values for different number of kernel functions (CIFAR-10 `car` and `cat`)

ure 5.7 displays the first convolution masks of size 9×9 learned by the proposed method for the MNIST experiment with (a) inlier class 4 and outlier samples from the NIST SD19 dataset and (b) inlier class 8 and outlier samples from the NIST SD19 dataset. This figure shows that our trained CNN model with the uLSIF criterion can abstract the parts of the inlier image samples. The part of an object in the image is one of the internal structures that are established by meaningful and invariant information from raw pixels.

The mean AUC values and the standard deviations over 10 trials are summarized in Table 5.6. Since outlier detection problem is easier than other experiments because of totally dissimilar outlier sample to inlier samples, we can get magnificent results. Through these experimental results, we can know that our algorithm is translation-invariant regardless of the type of outlier.

Table 5.2: Mean AUC values and the standard deviations over 20 trials for MNIST and USPS. The best method in terms of the mean AUC and comparable methods according to the *t-test* at the significance level 5% are specified by bold face.

Dataset (Inlier & outlier)		ρ	uLSIF (CNN)	uLSIF (kernel)	KLIEP (kernel)
MNIST	4 & 9	0.01	0.86 ± 0.12	0.64 ± 0.00	0.82 ± 0.00
		0.05	$\mathbf{0.89 \pm 0.01}$	0.65 ± 0.04	0.79 ± 0.06
	2 & 5	0.01	0.95 ± 0.05	0.79 ± 0.00	0.95 ± 0.00
		0.05	$\mathbf{0.98 \pm 0.01}$	0.91 ± 0.01	0.97 ± 0.00
	8 & 3	0.01	$\mathbf{0.97 \pm 0.02}$	0.91 ± 0.00	0.94 ± 0.00
		0.05	$\mathbf{0.97 \pm 0.01}$	0.83 ± 0.01	0.91 ± 0.00
	9 & 8	0.01	$\mathbf{0.97 \pm 0.05}$	0.88 ± 0.00	0.92 ± 0.00
		0.05	$\mathbf{0.97 \pm 0.01}$	0.71 ± 0.02	0.87 ± 0.00
USPS	4 & 9	0.03	$\mathbf{0.87 \pm 0.11}$	0.53 ± 0.00	0.73 ± 0.00
		0.05	$\mathbf{0.95 \pm 0.13}$	0.68 ± 0.03	0.82 ± 0.01
	2 & 5	0.03	0.97 ± 0.11	0.91 ± 0.00	0.93 ± 0.00
		0.05	0.99 ± 0.02	0.94 ± 0.01	0.98 ± 0.00
	8 & 3	0.03	$\mathbf{0.95 \pm 0.04}$	0.73 ± 0.00	0.77 ± 0.00
		0.05	$\mathbf{0.91 \pm 0.07}$	0.67 ± 0.02	0.83 ± 0.02
	9 & 8	0.03	0.91 ± 0.14	0.92 ± 0.00	0.94 ± 0.00
		0.05	$\mathbf{0.95 \pm 0.06}$	0.59 ± 0.02	0.80 ± 0.03

Table 5.3: Mean MSE values over 20 trials for MNIST and USPS. The best method in terms of the mean MSE and comparable methods according to the *t-test* at the significance level 5% are specified by bold face.

Dataset (Inlier & outlier)		ρ	uLSIF (CNN)	uLSIF (kernel)	KLIEP (kernel)
MNIST	4 & 9	0.01	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	**-0.51 $\pm$ 0.00**	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
	2 & 5	0.01	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
	8 & 3	0.01	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	**-0.52 $\pm$ 0.00**	-0.51 $\pm$ 0.01	-0.51 $\pm$ 0.00
	9 & 8	0.01	**-0.50 $\pm$ 0.00**	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	**-0.51 $\pm$ 0.00**	-0.50 $\pm$ 0.01	-0.50 $\pm$ 0.00
USPS	4 & 9	0.03	-0.49 $\pm$ 0.02	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	-0.50 $\pm$ 0.02	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
	2 & 5	0.03	-0.49 $\pm$ 0.04	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	-0.51 $\pm$ 0.01	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
	8 & 3	0.03	-0.49 $\pm$ 0.04	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	-0.50 $\pm$ 0.03	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
	9 & 8	0.03	-0.49 $\pm$ 0.00	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00
		0.05	-0.50 $\pm$ 0.02	-0.50 $\pm$ 0.00	-0.50 $\pm$ 0.00

Table 5.4: Mean AUC values and the standard deviations over 20 trials for PIE. The best method in terms of the mean AUC and comparable methods according to the *t-test* at the significance level 5% are specified by bold face.

PIE (Inlier & outlier)	ρ	uLSIF (CNNs)	uLSIF (kernel)	KLIEP (kernel)
1 & 12	0.03	**0.97 ± 0.02**	0.92 ± 0.00	0.96 ± 0.00
	0.05	**0.96 ± 0.02**	0.86 ± 0.00	0.93 ± 0.00
2 & 58	0.03	**0.85 ± 0.02**	0.53 ± 0.00	0.75 ± 0.00
	0.05	**0.81 ± 0.08**	0.37 ± 0.00	0.66 ± 0.00
4 & 35	0.03	0.86 ± 0.04	0.81 ± 0.00	0.85 ± 0.00
	0.05	0.69 ± 0.07	0.42 ± 0.00	0.53 ± 0.00
7 & 33	0.03	**0.90 ± 0.04**	0.64 ± 0.00	0.82 ± 0.00
	0.05	**0.94 ± 0.05**	0.65 ± 0.00	0.84 ± 0.00
12 & 62	0.03	0.56 ± 0.21	0.57 ± 0.00	0.65 ± 0.00
	0.05	0.67 ± 0.18	0.60 ± 0.00	0.63 ± 0.00
18 & 28	0.03	0.80 ± 0.07	0.62 ± 0.00	0.77 ± 0.00
	0.05	**0.81 ± 0.05**	0.51 ± 0.00	0.67 ± 0.00
19 & 32	0.03	**0.85 ± 0.02**	0.69 ± 0.00	0.83 ± 0.00
	0.05	0.84 ± 0.05	0.59 ± 0.00	0.76 ± 0.00
21 & 30	0.03	0.71 ± 0.12	0.51 ± 0.00	0.69 ± 0.00
	0.05	**0.95 ± 0.10**	0.66 ± 0.00	0.84 ± 0.00
24 & 37	0.03	0.91 ± 0.04	0.86 ± 0.00	0.90 ± 0.00
	0.05	**0.92 ± 0.03**	0.84 ± 0.00	0.89 ± 0.00
21 & 47	0.03	0.65 ± 0.13	0.51 ± 0.00	0.63 ± 0.00
	0.05	**0.67 ± 0.13**	0.33 ± 0.00	0.36 ± 0.00
40 & 11	0.03	0.81 ± 0.05	0.81 ± 0.00	0.83 ± 0.00
	0.05	0.86 ± 0.07	0.83 ± 0.00	0.85 ± 0.00

Table 5.5: Mean AUC values and the standard deviations over 20 trials for CIFAR-10. The best method in terms of the mean AUC and comparable methods according to the *t-test* at the significance level 5% are specified by bold face.

CIFAR (Inlier & outlier)	ρ	uLSIF (CNN)	uLSIF (kernel)	KLIEP (kernel)
Car and cat	0.03	**0.71 ± 0.06**	0.57 ± 0.01	0.65 ± 0.00
	0.05	**0.79 ± 0.03**	0.50 ± 0.01	0.52 ± 0.00
Airplane and bird	0.03	**0.63 ± 0.03**	0.51 ± 0.00	0.61 ± 0.00
	0.05	**0.65 ± 0.05**	0.50 ± 0.01	0.61 ± 0.00
Truck and car	0.03	0.58 ± 0.06	0.45 ± 0.02	**0.66 ± 0.00**
	0.05	0.66 ± 0.02	0.51 ± 0.00	0.65 ± 0.00
Ship and airplane	0.03	**0.64 ± 0.03**	0.61 ± 0.00	0.61 ± 0.00
	0.05	0.66 ± 0.02	0.49 ± 0.00	0.64 ± 0.00
Dog and frog	0.03	0.78 ± 0.08	0.59 ± 0.03	0.74 ± 0.00
	0.05	0.77 ± 0.10	0.64 ± 0.02	0.73 ± 0.01

Table 5.6: Mean AUC values over 10 trials for MNIST and NIST SD19. The best method in terms of the mean AUC and comparable methods according to the *t-test* at the significance level 5% are specified by bold face

Data (Inlier & outlier)	ρ	uLSIF (CNN)	uLSIF (kernel)	KLIEP (kernel)
MNIST 4 & NIST SD19	0.05	**1 ± 0.00**	0.45 ± 0.07	0.93 ± 0.02
MNIST 8 & NIST SD19	0.05	**1 ± 0.00**	0.65 ± 0.04	0.97 ± 0.00

Chapter 6

Conclusions and future work

In this chapter, we summarize the major contributions of this thesis and present several future works.

6.1 Conclusions

The density ratio framework is applicable to various in machine learning tasks. The direct density ratio approach estimates the density ratio directly so that we can avoid a difficult and fallible method that estimates each density respectively. This thesis was devoted to the direct density ratio approach to multi-label classification and outlier detection. The contributions in this thesis are summarized as follows:

- In Chapter 3, we proposed a computationally efficient multi-label method called *ML-LSPC* (multi-label least squares probabilistic classifier). We employed a multi-task learning method *MT-LSPC* (multi-task least squares probabilistic classifier) to solve the multi-label learning problem because similar labels should have similar classification solutions. However, because the essential number of training samples for multi-label classification

is multiplied according to the dimension of the label, naive implementation of ML-LSPC is computationally more expensive than MT-LSPC.

To improve computational efficiency, we introduced two ideas (Section 3.3). The first idea was to utilize the block structure of the system of linear equations. The second idea used a solver of the continuous Sylvester equation which often arises in control theory. Through experiments, we showed that the proposed method, ML-LSPC is promising.

- In Chapter 5, we proposed to use deep convolutional neural networks (CNN) in least-squares direct density-ratio estimation (uLSIF), and demonstrated its usefulness in inlier-based outlier detection of images. To the best of our knowledge, this is the first attempt to apply deep learning to density ratio estimation. There are several reasons for applying deep learning to density ratio estimation: First, the recent studies in pattern recognition demonstrated a deep model tends to perform better than shallow models. Especially, CNN has been found very effective and been commonly used in computer vision. Second, multi-layer structure of deep architecture gives us a chance to look into the outlier detection process.

  To describe uLSIF for CNN, We formulated uLSIF for a simple three-layers multilayer perceptron (MLP) in Section 5.3. Subsequently, we presented the uLSIF based on a deep CNN model in Section 5.4. The CNN model is trained with the uLSIF criterion (5.1) by the gradient descent method for a pair of samples to obtain a local minimizer.

  In Section 5.5, we presented the experimental setup and experiments on various real-world datasets. Outliers occur due to several reasons such as mechanical error, network problem and fraudulent behaviour. Among them, mislabelling by a human error is the most common factor of outliers. There-

fore, we had focused on mislabelling data and designed experiments. About hand written digit dataset, we tried to select similar digits to the inlier class for outlier class as much as possible. About PIE and CIFAR-10, the inlier class and outlier class are selected randomly. The experimental results showed that the proposed method gives a more accurate performance of the outlier detection than the shallow model based methods.

6.2 Future works

In this section, we will discuss possible future works.

In this thesis, we used the kernel model in LSPCs. This method achieved good results in terms of both accuracy and computational efficiency. On the other hands, we demonstrated the usefulness of a deep model in density ratio estimation in Chapter 5. A deep model based uLSIF for multi-label classification can also be investigated.

We showed the computation time of each algorithm in Section 5.5.3. The kernel-based uLSIF has the best performance in terms of computational efficiency. On the other hand, the deep CNN-based uLSIF requires huge computation power for training. To economize on the computation cost, we used the stochastic gradient descent algorithms in this thesis. Another possible learning algorithm is Hessian free (Kingsbury et al., 2012). Also several tricks can also be employed for reducing the computation cost of our method:

- *Over-specification*: It trains a deep model which are larger than needed (Livni et al., 2014).
- *Regularization*: Regularizing the weights of a deep model speeds up the convergence (Livni et al., 2014). Dropout (Hinton et al., 2012b) and maxout (Miao et al., 2013) are widely exploited in recent studies.

We used uLSIF for training CNN, but there are many other density ratio estimators (Sugiyama et al., 2012b). Developing algorithms for training CNN with other density ratio estimation approaches is an important future work. In this thesis, we focused on inlier-based outlier detection. However, we believe that the proposed method is also useful in other machine learning tasks that can be tacked via density ratio estimation (Sugiyama et al., 2012b).

Bibliography

Bilal Ahmed, Thomas Thesen, Karen E Blackmon, Yijun Zhao, Orrin Devinsky, Ruben Kuzniecky, and Carla E Brodley. Hierarchical conditional random fields for outlier detection: An application to detecting epileptogenic cortical malformations. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1080–1088, 2014.

Itamar Arel, Derek C Rose, and Thomas P Karnowski. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, 5(4):13–18, 2010.

Ira Assent, Philipp Kranen, Corinna Baldauf, and Thomas Seidl. Anyout: Anytime outlier detection on streaming data. In *Database Systems for Advanced Applications*, pages 228–242. Springer, 2012.

Josh Attenberg, Kilian Weinberger, Anirban Dasgupta, Alex Smola, and Martin Zinkevich. Collaborative email-spam filtering with the hashing trick. In *Conference on Email and Anti-Spam*, 2009.

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

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. 19:153, 2007.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM, 2007.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

Joel W Branch, Chris Giannella, Boleslaw Szymanski, Ran Wolff, and Hillol Kargupta. In-network outlier detection in wireless sensor networks. *Knowledge and information systems*, 34(1):23–54, 2013.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: identifying density-based local outliers. 29(2):93–104, 2000.

Guido Buzzi-Ferraris and Flavio Manenti. Outlier detection in large data sets. *Computers & chemical engineering*, 35(2):388–390, 2011.

Simon Byers and Adrian E Raftery. Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93(442):577–584, 1998.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search ranking. In *Proceedings of international conference on Knowledge discovery and data mining*, pages 1189–1198. ACM, 2010.

Weizhu Chen, Jun Yan, Benyu Zhang, Zheng Chen, and Qiang Yang. Document transformation for multi-label feature selection in text categorization. In *IEEE International Conference on Data Mining*, pages 451–456. IEEE, 2007.

Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

Koby Crammer and Yoram Singer. A family of additive online algorithms for category ranking. *The Journal of Machine Learning Research*, 3:1025–1058, 2003.

Francesco De Comité, Rémi Gilleron, and Marc Tommasi. Learning multi-label alternating decision trees from texts and data. In *Machine Learning and Data Mining in Pattern Recognition*, pages 35–49. Springer, 2003.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

Li Deng, Michael L Seltzer, Dong Yu, Alex Acero, Abdel-Rahman Mohamed, and Geoffrey E Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech*, pages 1692–1695. Citeseer, 2010.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge, 2010.

Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 53–62. ACM, 1999.

Hironori Fujisawa and Shinto Eguchi. Robust estimation in the normal mixture model. *Journal of Statistical Planning and Inference*, 136(11):3989–4011, 2006.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

Jing Gao, Haibin Cheng, and Pang-Ning Tan. Semi-supervised outlier detection. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 635–636. ACM, 2006.

Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM, 2005.

Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 2014.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. In *Dataset shift in machine learning*, volume 3, page 5. MIT press Cambridge, MA, 2009.

Patrick J Grother. Nist special database 19 handprinted forms and characters database. *National Institute of Standards and Technology*, 1995.

Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang. Face recognition using laplacianfaces. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 27(3):328–340, 2005.

Shohei Hido, Yuta Tsuboi, Hisashi Kashima, Masashi Sugiyama, and Takafumi Kanamori. Statistical outlier detection using direct density ratio estimation. *Knowledge and information systems*, 26(2):309–336, 2011.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012a.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

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

Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

Kevin Jarrett, Koray Kavukcuoglu, M Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445, 2009.

Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. Theoretical analysis of density ratio estimation. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 93(4):787–798, 2010.

Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3): 335–367, 2012.

Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. Computational complexity of kernel-based density-ratio estimation: a condition number analysis. *Machine learning*, 90(3):431–460, 2013.

Yoshinobu Kawahara and Masashi Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(2):114–127, 2012.

Brian Kingsbury, Tara N Sainath, and Hagen Soltau. Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

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

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

Quoc V Le. Building high-level features using large scale unsupervised learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8595–8598. IEEE, 2013.

Yann LeCun and M Ranzato. Deep learning tutorial. In *Tutorials in International Conference on Machine Learning*. Citeseer, 2013.

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

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103, 2011.

Peter J Lenk, Wayne S DeSarbo, Paul E Green, and Martin R Young. Hierarchical bayes conjoint analysis: recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15(2):173–191, 1996.

Tao Li and Mitsunori Ogihara. Detecting emotion in music. In *The International Society of Music Information Retrieval*, volume 3, pages 239–240, 2003.

Han Liu, John D Lafferty, and Larry A Wasserman. Sparse nonparametric density estimation in high dimensions using the rodeo. In *International Conference on Artificial Intelligence and Statistics*, pages 283–290, 2007.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pages 855–863, 2014.

Hamid R Marateb, Monica Rojas-Martínez, Marjan Mansourian, Roberto Merletti, and Miguel A Mañanas Villanueva. Outlier detection in high-density surface electromyographic signals. *Medical & biological engineering & computing*, 50(1):79–89, 2012.

Stephen Richard Marsland. *On-line novelty detection through self-organisation, with application to inspection robotics*. University of Manchester, 2001.

Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5):555–559, 2003.

Andrew McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI99 Workshop on Text Learning*, pages 1–7, 1999.

Yajie Miao, Florian Metze, and Shourabh Rawat. Deep maxout networks for low-resource speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 398–403. IEEE, 2013.

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

Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1279–1287, 2010.

XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *Information Theory, IEEE Transactions on*, 56(11):5847–5861, 2010.

The Calo Project. *Enron email dataset*. 2009. URL `http://www.cs.cmu.edu/enron/`.

Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

Vasile Sima. Algorithms for linear-quadratic optimization. 200, 1996.

Jaak Simm, Masashi Sugiyama, and Tsuyoshi Kato. Computationally efficient multi-task learning with least-squares probabilistic classifiers. *IPSJ Transactions on Computer Vision and Applications*, 3:1–8, 2011.

Le Song, Choon H Teo, and Alex J Smola. Relative novelty detection. In *International Conference on Artificial Intelligence and Statistics*, pages 536–543, 2009.

Masashi Sugiyama. Superfast-trainable multi-class probabilistic classifier by least-squares posterior fitting. *IEICE Transactions on Information and Systems*, 93(10):2690–2701, 2010.

Masashi Sugiyama. Machine learning with squared-loss mutual information. *Entropy*, 15(1):80–112, 2012.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. pages 1433–1440, 2008.

Masashi Sugiyama, Ichiro Takeuchi, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Daisuke Okanohara. Least-squares conditional density estimation. *IEICE Transactions on Information and Systems*, 93(3):583–594, 2010.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012a.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012b.

Masashi Sugiyama, Takafumi Kanamori, Taiji Suzuki, M Plessis, Song Liu, and Ichiro Takeuchi. Density-difference estimation. *Neural computation*, 25(10): 2734–2775, 2013a.

Masashi Sugiyama, Song Liu, Marthinus Christoffel Du Plessis, Masao Yamanaka, Makoto Yamada, Taiji Suzuki, and Takafumi Kanamori. Direct divergence approximation between probability distributions and its applications in machine learning. *Journal of Computing Science and Engineering*, 7(2): 99–111, 2013b.

Masashi Sugiyama, Makoto Yamada, and Marthinus Christoffel du Plessis. Learning under nonstationarity: covariate shift and class-balance change. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(6):465–477, 2013c.

Taiji Suzuki, Masashi Sugiyama, Takafumi Kanamori, and Jun Sese. Mutual information estimation reveals global associations between stimuli and biological processes. *BMC bioinformatics*, 10(Suppl 1):S52, 2009.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2010.

Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. *Information and Media Technologies*, 4(2):529–546, 2009.

Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010.

Makoto Yamada and Masashi Sugiyama. Direct importance estimation with gaussian mixture models. *IEICE transactions on information and systems*, 92(10): 2159–2162, 2009.

Makoto Yamada, Masashi Sugiyama, Gordon Wichern, and SIMM Jaak. Improving the accuracy of least-squares probabilistic classifiers. *IEICE transactions on information and systems*, 94(6):1337–1340, 2011.

Kenji Yamanishi, Jun-Ichi Takeuchi, Graham Williams, and Peter Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 320–324. ACM, 2000.

Dong Yu and Li Deng. Deep learning and its applications to signal and information processing [exploratory dsp]. *Signal Processing Magazine, IEEE*, 28(1): 145–154, 2011.

Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 18(10):1338–1351, 2006.

Min-Ling Zhang and Zhi-Hua Zhou. Multi-label learning by instance differentiation. In *AAAI*, volume 7, pages 669–674, 2007a.

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007b.