Adaptive Learning Algorithms for Non-stationary Data

by

Yun-Qian Miao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2015

Abstract

With the wide availability of large amounts of data and acute need for extracting useful information from such data, intelligent data analysis has attracted great attention and contributed to solving many practical tasks, ranging from scientific research, industrial process and daily life. In many cases the data evolve over time or change from one domain to another. The non-stationary nature of the data brings a new challenge for many existing learning algorithms, which are based on the stationary assumption.

This dissertation addresses three crucial problems towards the effective handling of non-stationary data by investigating systematic methods for sample reweighting. Sample reweighting is a problem that infers sample-dependent weights for a data collection to match another data collection which exhibits distributional difference. It is known as the density-ratio estimation problem and the estimation results can be used in several machine learning tasks. This research proposes a set of methods for distribution matching by developing novel density-ratio methods that incorporate the characters of different non-stationary data analysis tasks. The contributions are summarized below.

First, for the domain adaptation of classification problems a novel discriminative density-ratio method is proposed. This approach combines three learning objectives: minimizing generalized risk on the reweighted training data, minimizing class-wise distribution discrepancy and maximizing the separation margin on the test data. To solve the discriminative density-ratio problem, two algorithms are presented on the basis of a block coordinate update optimization scheme. Experiments conducted on different domain adaptation scenarios demonstrate the effectiveness of the proposed algorithms.

Second, for detecting novel instances in the test data a locally-adaptive kernel density-ratio method is proposed. While traditional novelty detection algorithms are limited to detect either emerging novel instances which are completely new, or evolving novel instances whose distribution are different from previously-seen ones, the proposed algorithm builds on the success of the idea of using density ratio as a measure of evolving novelty and augments with structural information of each data instance's neighborhood. This makes the estimation of density ratio more reliable, and results in detection of emerging as well as evolving novelties.

In addition, the proposed locally-adaptive kernel novelty detection method is applied in the social media analysis and shows favorable performance over other existing approaches. As the time continuity of social media streams, the novelty is usually characterized by the combination of emerging and evolving. One reason is the existence of large common vocabularies between different topics. Another reason is that there are high possibilities of topics being continuously discussed in sequential batch of collections, but showing different level of intensity. Thus, the presented novelty detection algorithm demonstrates its effectiveness in the social media data analysis.

Lastly, an auto-tuning method for the non-parametric kernel mean matching estimator is presented. It introduces a new quality measure for evaluating the goodness of distribution matching which reflects the normalized mean square error of estimates. The proposed quality measure does not depend on the learner in the following step and accordingly allows the model selection procedures for importance estimation and prediction model learning to be completely separated.

Acknowledgements

There are a great number of people who have contributed significantly to my Ph.D. years and the completion of this dissertation.

First, I would like to express my sincere gratitude to my supervisor Prof. Mohamed Kamel for his continuous encouragement and careful supervision. I thank him not just because his valuable input to the work presented in this dissertation, but also for his attitudes and insights in the mentoring process, which is the key in my development as a researcher.

I would also like to express my appreciation to the committee members, Prof. Robi Polikar, Prof. Fakhri Karray, Prof. Zhou Wang and Prof. Daniel Stashuk, for their valuable comments and suggestions.

I am grateful to my colleagues at the Centre of Pattern Analysis and Machine Intelligence and University of Waterloo for many stimulating discussions and warm friendship. My thanks go to Mehrdad Gangeh, Rodrigo Araujo, Mohamed Kacem Abida, Jamil Abou Saleh, Michael Diu, Mostafa Hassan, Allaa Hilal, Bahador Khaleghi, Yibo Zhang, En-Shiun Annie Lee, Zhenning Li, Arash Tabibiazar, Sepideh Seifzadeh, Hossein Parsaei, Pouria Fewzee, Ahmed Elgohary, Safaa Bedawi and Raymond Zhu. Particularly, I wish to thank Dr. Ahmed Farahat for the considerable advice and assistance.

Many thanks go to the Natural Sciences and Engineering Research Council of Canada (NSERC), the Ontario Graduate Scholarship (OGS) program, and the Graduate Studies Office at the University of Waterloo for providing me the financial support.

At the end, I am heartily thankful to my wife and my two sons for their understanding and accompanying. I wish to thank my parents and parents-in-law for believing in me. Thank you all the people who made this possible.

Dedication

I would like to dedicate my work to my parents.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

cLSIF:	constrained Least-Squares Importance Fitting
iwCV:	importance-weighted Cross-Validation
iwLSPC:	importance-weighted Least-Squares Probabilistic Classifier
iwSVM:	importance-weighted Support Vector Machine
locKMM:	locally-adaptive Kernel Mean Matching
uLSIF:	unconstrained Least-Squares Importance Fitting
AUC:	Area Under the ROC Curve
BCU:	Block Coordinate Update
DA:	Domain Adaptation
DDR:	Discriminative Density Ratio
DR:	Density Ratio
EM:	Expectation-Maximization
GMM:	Gaussian Mixture Model
KDE:	Kernel Density Estimation
KLIEP:	Kullback-Leibler Importance Estimation Procedure
KMM:	Kernel Mean Matching
LCV:	Likelihood Cross-Validation
LOF:	Local Outlier Factor
LSE:	Least Square Error
LSIF:	Least-Squares Importance Fitting
MDP:	Markov Decision Process
MI:	Mutual Information
ML:	Maximum Likelihood
MMD:	Maximum Mean Discrepancy
NE:	Normalized Error
NMSE:	Normalized Mean Squared Error
OSVM:	One-class Support Vector Machine
PDF:	Probability Density Function

PSD:	Positive Semi-Definite
QP:	Quadratic Programming
RKHS:	Reproducing Kernel Hilbert Space
RLS:	Regularized Least Square
RND:	Radon-Nikodym Derivative
ROC:	Receiver Operating Characteristic
S3VM:	Semi-Supervised Support Vector Machine
SCL:	Structural Correspondence Learning
SSL:	Semi-Supervised Learning
SVDD:	Support Vector Data Description
SVM:	Support Vector Machine
TCA:	Transfer Component Analysis
TSVM:	Transductive Support Vector Machine
VSM:	Vector Space Model

Chapter 1

Introduction

The non-stationary nature of data brings a new challenge for many existing learning algorithms. How to update models that adapt to distribution changes is crucial in many data mining tasks. This dissertation mainly addresses the non-stationarity problem in data mining and aims at providing adaptive solutions.

This chapter introduces the research scope, summarizes the contributions being made, and outlines the structure of this dissertation.

1.1 Motivation

Currently, the information and communication systems are characterized by continuous generation, transfer and storage of large amounts of data. In many cases the data keep evolving over time or change from one domain to another. For example, remote sensing images, which are intensively used for land-use classifications, change over time due to seasonal differences. Another example is speech recognition, which may involve a wide variety of different accents or changes due to speaker gender.

The non-stationarity of data negatively affects the sustainability of traditional data mining techniques and prevents their applicability to new domains. Meanwhile, re-collecting and re-training models for new environments are often difficult or even impossible because of either the time constraint or the cost of sample labeling. Therefore, we are striving for effective adaptive solutions, which require no or very little supervision for each change of domains.

There are numerous data analysis scenarios in which adaptive learning algorithms are needed to deal with the dynamics of data. The following explains several well-known examples.

Sample selection bias. The sampling bias is directly linked to the selection process of training data collection. For example, in social science research, the survey is conducted with extra samples that can be easily accessed by the investigators. Thus, the fitted model with the biased samples will be suboptimal to the more general distributed populations when put into a real application [53].

Time evolving data. This scenario is mostly linked to streaming data, in which distributions keep evolving along the time axis. For example, financial data analysis is not a static task where the underlying patterns are continuously affected by many social, political, environmental, and economic factors [63]. The non-stationary adaptation problem is different from incremental learning in that the first tries to fit the learning model to new data using the previous model for old data, while the second tries to fit the model to both old and new data starting from the previous model.

Adversarial behavior data. Some important applications involve the existence of adversarial behaviors that try to work around existing learned models. The data instances of these adversarial behaviors may appear in the test set only and generate the testing and training data distributional difference. Online information systems are often exposed with this type of non-stationarity, such as the task of spam filtering [15] and network intrusion detection [132]. The adversarial relationship between the intruder and the detector defines the nonstop evolving behavior of intrusions. Another vital application is the biomedical related task, where new types of bacterial serovars are mutated rapidly [2]. It is impossible to collect exhaustive training samples that cover all types of bacteria.

Cross dataset adaptation. The cross dataset is the scenario in which the training dataset and the test dataset are collected in different contexts. Thus, their underlying distributions demonstrate some differences due to different data collection processes and/or equipments. This is especially true in image-based object recognition tasks, where the acquired images show systematic difference in two setups because of the change of light condition, sensor, and so on. We have witnessed several important investigations that deal with cross dataset image-based object recognition [87, 104] and sentiment analysis of customer reviews for cross product category [25, 50].

1.2 Research Objective and Contributions

Most machine learning algorithms are based on an implicit assumption of identical distribution between the training and test data. Thus, the fitted model minimizes a loss function defined on the training data that will be expected to perform well in the unseen test data. However, the wide existence of non-stationary data mining problems are characterized by the distribution divergence between the data for model fitting and for the model to be applied to. This violates the stationary assumption and usually leads to performance degradation of a well-trained system.

The objective of this research is to investigate systematic methods for training sample reweighting to provide adaptive solutions for non-stationary data mining tasks. Sample reweighting is a problem that infers sample-dependent weights for a data collection to match another data collection which exhibits distributional difference. In the statistical and machine learning community, this problem is formulated as the density-ratio estimation problem [47, 53, 64, 110]. This research analyzes the non-stationary data mining problem and proposes a set of methods for effective handling of non-stationary data by developing novel density-ratio methods that incorporate characters of different non-stationary data analysis tasks. The contributions of this dissertation are summarized as follows.

- A novel Discriminative Density Ratio (DDR) estimation framework is developed for domain adaptation of classification tasks. This approach emphasizes the discrimination ability of classifiers in the reweighted space during the adaptation procedure. The proposed DDR approach is formulated in a rigorous mathematical format and then two algorithms are presented based on the block coordinate update optimization scheme. Experiments conducted on different domain adaptation scenarios demonstrate the effectiveness of the proposed algorithms.

- A locally-adaptive kernel density-ratio method is proposed for emerging and evolving novelty detection. The proposed approach captures the two characteristics of novelties in one formula through the construction of neighborhood-decided kernels. The effectiveness is shown by comparing other well-known novelty detection methods.

- The locally-adaptive kernel density-ratio method is investigated in the application to analyze the dynamics in social media. Its superiority is shown in the detection of both emerging and evolving topics in tweets data.

- The parameter tuning mechanism for the non-parametric kernel mean matching method is studied and a novel auto-tuning method is presented. The proposed method assesses the quality of candidate choices from a perspective that reflects the normalized mean squared error of estimated density ratios.

1.3 Document Structure

This document consists of seven chapters, which are organized as follows.

After the introduction chapter, Chapter 2 and Chapter 3 serve as background. Chapter 2 provides an overview on the non-stationary learning problem. Chapter 3 describes the probability density ratio estimation problem and reviews state-of-the-art algorithms.

The following three chapters are dedicated to new developments for three unique non-stationary learning problems. In Chapter 4, a discriminative density ratio estimation framework is developed, which aims to effectively deal with domain adaptation classification tasks. In Chapter 5, a novel locally-adaptive density ratio method is proposed for novelty detection tasks, which solves the occurrence of both emerging and evolving novelties. In Chapter 6, the parameter tuning problem for the kernel mean matching method is investigated.

Finally, Chapter 7 concludes this dissertation and discusses some future directions.

Chapter 2

Background: Learning From Non-stationary Data

This chapter provides a background on the topic of adaptive learning for non-stationary data. Section 2.1 gives an overview of different learning paradigms and the challenging problems in machine learning and data mining research. Section 2.2 describes the details of the non-stationary learning problem and lists the learning scenarios that cause the dynamics. In Section 2.3, existing work is reviewed, which covers supervised learning, unsupervised learning, and novelty detection. At the end, the positioning of the dissertation is summarized in Section 2.4.

2.1 The Stationary Assumption of Learning Algorithms

Machine learning is the study of the development of systems that can learn from data, rather than follow only explicitly programmed instructions. Machine learning algorithms are employed in a wide range of computing tasks where hard-coded rule-based algorithms are infeasible. Example applications include spam filtering, image recognition, speech recognition, and natural language understanding. Sometimes machine learning and data mining are used interchangeably. Machine learning focuses on the designing of generic learning algorithms, while data mining aims at extracting patterns and knowledge from large amounts of data with the help of machine learning methods and tools for other tasks, such as data preparation, data management, and result visualization [58, 83].

2.1.1 Machine Learning Paradigms

Depending on different learning settings and objectives, the paradigms of machine learning algorithms can be categorized into the following cases:

Supervised learning. The task of supervised learning is to model the functional relations between the input and output based on the given collection of input and output pairs, i.e. training samples. After a model being learned, we expect it can be used to predict outputs for unseen inputs. Then, the objective of supervised learning is to achieve the best performance in term of minimal generalized error [58, 119], which depends on the model's complexity and the joint distribution of input-output pairs.

Unsupervised learning. In unsupervised learning, outputs are not provided in the training data. The general purpose of unsupervised learning is to extract valuable information from the data. The exact objective and performance evaluation are varying and determined by specific tasks, which include clustering analysis, association rule mining, recommendation system, etc. [56, 65]. Inevitably, the inferred model is decided by the data at hand and the underlying distributional properties.

Semi-supervised learning. Semi-Supervised Learning (SSL) is a learning paradigm that exploits the limited number of labeled data and a large number of unlabeled data to learn a strong model. Semi-supervised learning attracts great interest in machine learning and data mining because it can use readily available unlabeled data to improve supervised learning tasks when the labeled data are scarce or difficult to obtain. There are many SSL algorithms being proposed, which include mixture models, Semi-Supervised Support Vector Machines (S3VMs), manifold learning, co-training, and some others [22, 134]. These algorithms improve generalization performance under their assumptions regarding population distribution and model structure.

Reinforcement learning. Reinforcement learning is to learn a policy function for software agents that maps the situations to actions, with a goal to select optimal actions in an environment in order to maximize some notion of cumulative reward [4]. Different from supervised learning, in reinforcement learning the correct outputs can not be obtained directly. However, it is also not like unsupervised learning because some forms of reward information are available to the policy learner during interactions with the environment.

Usually the environment is formulated as a Markov Decision Process (MDP), and then reinforcement learning algorithms aim to find an optimal action-selection policy, such as the Q-learning algorithm [121]. Reinforcement learning is particularly suitable to solve problems which include a long-term versus short-term reward trade-off, and have been applied successfully to various problems, including robot control, game theory, and economics.

Active learning. Similar to semi-supervised learning, active learning is a learning paradigm to deal with situations where unlabeled data is abundant but manually labeling is expensive. In such a scenario, learning algorithms can actively query human experts for labels. This type of iterative labeling query and supervised learning is called active learning [102]. Since the learning algorithm chooses the examples to label, the number of labeled examples to learn a concept can often be much lower than the number required in normal supervised learning.

2.1.2 The Stationary Assumption

Among various challenges raised in machine learning and data mining research [36, 55, 124, 127], the non-stationarity of data, in which the properties of data evolve over time and change from one domain to another, is prominent. However, classical data mining algorithms, including supervised learning and unsupervised learning methods, usually aim to model the statistical characteristics from a given collection of data. The tuning and validation of the built models are then based on some cross validation strategies [56], such as the k-fold cross validation. When deploying these models in real environments, they continue to perform very well on new data as long as these data belong to the same distribution as the original data. This is related to a key assumption behind most machine learning algorithms, which is that the test data is generated from the same distribution as the training data.

For different learning settings, if there is a distributional mismatch between the test and training data, the model fitted using the training data would be sub-optimal for the test scenario. This is a challenging problem that exists in many real application scenarios.

2.2 Non-stationary Data

Today's information and communication systems are characterized by continuous generation, transfer and storage of a large amounts of data. Many instances of these data keep evolving and their properties change from one domain to another. This non-stationary nature of the data negatively affects the sustainability of traditional data mining techniques over time and prevents their applicability to new domains. In the following, the non-stationary phenomenon is examined from a statistical perspective.

2.2.1 Types of Distribution Change

Bayes' rule describes the relationship between different probability functions over the input variable x and the target variable y as

$$p(x, y) = p(y|x)p(x) = p(x|y)p(y) \; , \tag{2.1}$$

where $p(x, y)$ is the joint distribution, $p(x)$ is the marginal distribution, $p(x|y)$ is the class conditional distribution, $p(y)$ is the class prior, and $p(y|x)$ is the posterior.

According to different types of distribution changes, these terminologies are commonly used in the literature: covariate shift, prior change, concept drift, and mixture of changes. The covariate shift implies that the change is in the marginal distribution while the posterior remains the same [105]. The prior change means the class priors are shifted while the class conditional distributions remain the same. This type of change is also referred as the class imbalance problem [66, 112]. In the literature, concept drift usually refers to streaming data and the term is used diversely [90,135]. According to the definition in [91], there are two types of concept drift. One type is the change happening in class conditional distributions, the other type is the change happening in posteriors. For real applications, there are cases in which several types of the distributions demonstrate changes simultaneously. We classify these cases into the category of mixture of change. The different types of distribution changes are summarized in Table 2.1. All the changes are finally reflected in the joint distribution.

The aforementioned types of changes are simplifications the realistic scenarios and they are mainly based on the probability functions of Bayes' rule. The actual learning settings may cause one or several types of distribution changes simultaneously.

2.2.2 Learning Scenarios with Non-stationarity

The actual settings that cause the generation of non-stationary data have the following typical cases.

Table 2.1: Types of distribution change. ('-' means the change in that category is theoretically possible, but is not concerned in the case)

Terminology	$p(x)$	$p(y)$	$p(y\|x)$	$p(x\|y)$	$p(x,y)$
Covariate shift	change	-	same	-	change
Prior change	-	change	-	same	change
Concept drift-1	-	same	-	change	change
Concept drift-2	same	-	change	-	change
Mixture change	possible	possible	possible	possible	change

Sample selection bias. The term sampling bias is directly linked to the selection process of training data collection [53]. For example, in social science research, the survey is conducted with extra samples that are more easy-accessed to the investigators than difficult cases. We can define a selection variable s and $P\left(s=1|\left(x,y\right)\right)$ as the probability of the sample (x,y) being included in the training collection. In the case that $P\left(s=1|\left(x,y\right)\right) = P\left(s=1|x\right)$, the biased selection process is determined by the input variable x only and potentially produces the covariate shift scenario. If $P\left(s=1|\left(x,y\right)\right) = P\left(s=1|y\right)$, the biased selection process is determined by the class information and hence produces prior change. When the biased selection process depends on both x and y, this will produce the scenario of a mixture of change.

Time evolving data. This scenario is mostly linked to streaming data, in which distributions vary with the evolution of time [118, 135]. But the non-stationary adaptation problem is different from incremental learning in that the first tries to fit the learning model to new data using the previous model for old data, while the second tries to fit the model to both old and new data starting from the previous model. One example is the remote sensing applications, where the data for model-building are collected in a season. But later the model will face the mismatch challenges due to the seasonal difference [5, 73].

Adversarial behavior data. Some important applications involve the existence of adversarial behaviors that try to work around the existing learned models. Accordingly, the data instances of these adversarial behaviors appear in the test set only and generate the testing and training data distribution discrepancy. Online information systems are often exposed with this type of non-stationarity, such as the task of spam filtering [15] and network intrusion detection [132]. The adversarial relationship between the intruder and the detector defines the nonstop evolving behavior of intrusions. In biomedical applications, new types of bacterial serovars are mutated rapidly. It is impossible to collect an exhaustive training collection that covers all types of bacteria. On the other hand, classifying pathogenic bacteria as non-pathogenic would have unfortunate consequences.

Cross dataset adaptation. The cross-dataset adaptation problem is the scenario in which the training dataset and the test dataset are collected in different contexts. Their underlying distributions demonstrate some differences, and in most cases involve one or more types of distributions. This is especially true in the image-based object recognition task, where the acquired images show systematic difference in two setups because of a change of light condition, devices, and so on. Limited by labeling cost or impossibility of obtaining labels in the target dataset, this learning scenario aims to fit a model by using the labeled training data and the unlabeled target data. We have witnessed several important research works dealing with the cross-dataset image-based object recognition [87,104] and the sentiment analysis of customer reviews on different product categories [25,50].

Overall, for the realistic domain adaption tasks, the distribution changes are usually a mixture of more than one type. The single mode of distribution change, such as the covariate shift, is just a simplification and can fail to capture the change of other aspects.

2.3 Existing Work

Distribution discrepancies between the training and test data violate the identical assumption and lead to the fitted model encountering performance degradation or even becoming totally outdated. Due to the importance of the problem, recent years have witnessed considerable research and development activities to propose various adaptive solutions for non-stationary data mining. In the following, the literature review is conducted, which covers supervised learning, unsupervised learning, and novelty detection.

2.3.1 Domain Adaptation

In supervised learning, including classification and regression, the adaptation mechanism of learning is formally known as the Domain Adaption (DA) problem and it has received considerable attention in the research community during the past few years [8,9,27]. There are three directions of work being proposed to tackle the domain adaptation problem: dynamic ensemble, feature transformation, and sample reweighting.

The ensemble method is a learning paradigm that combines a set of base components to model a complex problem through proper decision fusion rules. It represents a comprehensively studied approach in the streaming data concept drift scenario [37, 74, 120]. The adaptivity is achieved by dynamic weight assignment at the decision layer. The weight of a component in the ensemble is usually decided through evaluating its performance in the most recent batch of data, while assuming the streaming data can continuously obtain the labeling information. For the cross-dataset task, the AdaBoost-style algorithm for model adaptation are proposed [29, 125]. It assumes a number of labeled samples available in the target dataset, and uses different weighting strategies for the training data and labeled target data in the boosting iteration.

Another approach to domain adaptation is based on the assumption of the existence of a domain-invariant feature space. Defining and quantifying a transformation to find such a feature space and then the adaptation can be accomplished by learning a model on the new space. Pan et al. [92] proposed the Transfer Component Analysis (TCA) method that learns the feature transformation to produce a set of common transfer components across domains in a reproducing kernel Hilbert space. A similarly idea is described in [38] with a closed-form solution. Blitzer et al. [9] proposed the Structural Correspondence Learning (SCL) method that learns a common feature space by identifying correspondences among features from different domains. In [25], the deep learning approach is proposed to generate robust cross-domain feature representations using the output of the intermediate layers. The work of [24] is on the direction of combining feature selection with the co-training scheme, which gradually finds a subset of features that are suitable for the target domain only, instead of good for both.

The sample reweighting approach for domain adaptation is to assign sample-dependent weights for the training data with the objective to minimize the distribution discrepancy between the training data and the test data in the reweighted space [53, 110]. Having the weighted training samples, the cost-sensitive learning methods can be applied to produce a model that adapts to the test data distribution. So, the sample reweighting is a general approach that can make use of advances in many cost-sensitive learning algorithms, ranging from the single model such as the cost-sensitive SVM [84], to the ensemble model such as the cost-sensitive boosting [112].

2.3.2 Unsupervised Learning of Non-stationary Data

In unsupervised learning, output values are not provided in training samples. Its general goal is to extract valuable information from data, which includes tasks such as clustering [44], matrix factorization [122] and subset selection [43].

To analyze the properties of non-stationary unlabeled data, different adaptive clustering approaches with different inspirations have been proposed. Such as the evolutionary spectral clustering techniques [16, 76] update learning models constrained with model smoothness. In [99], the self-organizing map method is modified to incorporate short-term and long-term memory that enable discovering the occurrence of new clusters and exploring the properties of structural changes in data at the same time. In [17, 77], incremental clustering methods are designed with the ability to handle the dynamics in the data.

Dynamic changes in data have also gained attention in the non-negative matrix factorization research. In [14], the authors proposed an incremental subspace learning scheme that is capable of adaptively controlling the contribution of each sample to the representation by assigning different weighting coefficients to each sample in the cost function. In the application of automatic speech recognition with highly non-stationary noisy environments, the work of [122] applied the convoluted non-negative matrix factorization to enhance speech signals.

2.3.3 Novelty Detection and Emerging Concept Discovery

In novelty detection, the task is to identify novel instances in the test data that differ in some respect from a collection of training data that contains only *normal* data. It is usually formulated as the one-class classification problem. Because there exists a large number of possible abnormal modes and some of them may be evolving continuously, it is ineffective or even impossible to model all the abnormal patterns. Beside the term of novelty detection [94], some other related topics are also often used as outlier detection [60] and anomaly detection [19].

Various approaches to novelty detection are proposed. The probabilistic-based novelty detection method estimates the Probability Density Function (PDF) of normal data, and assumes that the low density areas have high probability of being novel [46, 107]. The one-class SVM method models the boundary of normal data and assumes that samples located outside of the boundary are novel [100]. The neighborhood-based approach analyzes the distances of k-nearest neighbors, and identifies novel instances if they are relatively far from their neighbors [13].

In [106], Smola et al. proposed a concept of relative novelty and modified the One-class Support Vector Machine (OSVM) to incorporate the reference densities as density ratios. Following, Hido et al. [59] proposed an inlier-based outlier detection method and defined the inline score by using density ratios between the normal and the test data. For the regions that the inlier scores are small, it means the normal data density is low and the test data density is high. Thus, with the relative densities the novel (outlier) instance can be identified if its inlier score is below a threshold.

Emerging concept discovery is another task related to novelty detection, which aims to identify new concepts or classes and update models to incorporate them. In [85], both concept drift and novel class detection are considered in streaming data scenarios with a delayed labeling process. The concept drift problem is addressed by continuously updating an ensemble of classifiers to include the most recent concept changes. The novel class detection is handled by enriching each classifier in the ensemble with a novelty modeler. In [41, 61], active learning is employed to discover new categories and learn their models while pursuing minimal labeling efforts.

2.4 Summary

This chapter introduces the challenging machine learning problem that is arisen from non-stationary data. After analyzing different types of distribution changes and the typical learning scenarios that demonstrate non-stationarity, a comprehensive review of the existing work is presented. The review covers the learning settings of supervised learning, unsupervised learning, and novelty detection.

The rest of this dissertation will focus on the study of probability density-ratio estimation methods and contribute to non-stationary data mining from three aspects: domain adaptation in classification problems, novelty detection and analysis, and the parameter selection problem in the kernel mean matching algorithm.

Chapter 3

A Review on The Estimation of Probability Density Ratios

Estimating the ratio of probability densities from two collections of data is a new topic in the statistical and machine learning community. It has attracted great interest due to its potential for solving many challenging learning problems, such as covariate shift adaptation, outlier detection, semi-supervised learning, mutual information estimation, and some others.

This chapter presents a review on the density-ratio estimation problem. It first describes the formulation of the problem in Section 3.1. Then, the state-of-the-art estimation methods are reviewed in Section 3.2. Following, some applications that can employ the density ratio techniques are discussed in Section 3.3. A summary is given at the end in Section 3.4.

3.1 The Problem Formulation

Let $\mathcal{X} \subset \mathbb{R}^d$ be a d-dimension data space. We are given m independent and identically distributed (*i.i.d.*) data samples $\mathcal{S} = \{x_i \in \mathcal{X} | i = 1, \ldots, m\}$ that are from a distribution with probability density function $p(x)$, and another set of n *i.i.d.* data samples $\mathcal{S}' = \{x'_i \in \mathcal{X} | i = 1, \ldots, n\}$ that are from a different distribution with density function $p'(x)$. Suppose $p'(x)$ is continuous with respect to $p(x)$ (i.e. $p(x) = 0$ implies $p'(x) = 0$). The Density-Ratio (DR) problem (also known as the sample importance estimation problem) is to estimate the ratio

$$\beta(x) = \frac{p'(x)}{p(x)} , \tag{3.1}$$

from the given finite samples $\mathcal{S}$ and $\mathcal{S}'$.

3.1.1 Related Topics

Radon-Nikodym Derivative

The density ratio is a special case of a function termed as *Radon-Nikodym Derivative (RND)* [64], which is defined as a function of the ratio of two derivatives. Taking probability into consideration, the derivative is a probability density. In some work the density-ratio problem is also referred to as a RND problem [128]. But, strictly speaking the RND is a more general term than the density-ratio problem.

Importance Weight

Another term that is used interchangeably with density-ratio is *importance weight*, which takes the concept of importance sampling [47]. The intuition is from the view angle of changing the original problem in Eq. 3.1 into $\beta(x)p(x) = p'(x)$, where the $\beta(x)$ is a sample-dependent weighting term to match one distribution to another distribution. For most cases in this thesis, the density-ratio will be used unless being indicated explicitly.

Necessary Condition

The validity of a density-ratio requires that the support of $p'(x)$ is contained by the support of $p(x)$. This is the necessary condition claimed in the problem, which assumes that $p'(x)$ is absolutely continuous with respect to $p(x)$ [62, 133]. In other words, for regions in the feature space where $p(x) = 0$, we implicitly imply $p'(x) = 0$ for applying the density-ratio to recover the distribution of $p'(x)$ in the learning setting.

To deal with distribution changes, such as covariate shift and sampling bias correction, most existing approaches make use of the density ratios to reweight samples [53, 110]. In fact these proposed methods ignore this necessary condition and the solution is an approximation to some extent.

3.2 Density-Ratio Estimation Methods

Recently a number of methods have been proposed to estimate the Density Ratio (DR) [111]. According to different optimization formulations, existing methods can be classified into four categories: 1) Density estimation-based two-step approach; 2) Moment matching, e.g. the Kernel Mean Matching (KMM) algorithm; 3) Density model fitting, e.g. the Kullback-Leibler Importance Estimation Procedure (KLIEP) algorithm; and 4) Density-ratio model fitting, e.g. the constrained Least-Squares Importance Fitting (cLSIF) and unconstrained Least-Squares Importance Fitting (uLSIF) algorithm. Here we will review these well-known density-ratio estimation algorithms.

3.2.1 Two-step Approach

The straightforward way to estimate the density-ratio is composed of two steps. First, the Probability Density Function (PDF) $p(x)$ and $p'(x)$ are estimated from the two collections of samples $\mathcal{S}$ and $\mathcal{S}'$ separately. Then, the density ratio $\beta(x)$ is obtained by taking the division of them as $\beta(x) = {}^{p'(x)}/_{p(x)}$. Therefore, the density-ratio estimation does not involve any more burden than estimating two density functions. The following explains two often-used density estimation methods, the Gaussian Mixture Model and Kernel Density Estimation.

Gaussian Mixture Model

Estimating the density function itself has a rich history. There are mainly two directions of work on density estimation: the parametric techniques and the non-parametric techniques. Parametric methods assume a model governing the data generation and the parameters of the model are learned from the given data. The Gaussian Mixture Model (GMM) [97] is a popular parametric approach, which assumes the data are generated from a weighted mixture of Gaussian distributions as

$$\hat{p}(x|\lambda) = \sum_{i=1}^{M} w_i g(x|u_i, \Sigma_i) \ , \tag{3.2}$$

where $w_i, i = 1, \dots M$ are the mixture weights, $g(x|u_i, \Sigma_i), i = 1, \dots M$ are the M component Gaussian distributions. Each component is a d-dimension Gaussian as

$$g(x|u_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x - u_i)^T \Sigma_i^{-1} (x - u_i) \right) . \tag{3.3}$$

Then the GMM is modeled by the set of M component mean vectors, component covariance matrices, and the weights for the components, expressed as

$$\lambda = \{w_i, u_i, \Sigma_i | i = 1, \dots, M\} . \tag{3.4}$$

To learn the parameters λ from the training data X, Maximum Likelihood (ML) estimation using the Expectation-Maximization (EM) algorithm is a well-established method [32]. The idea of the EM algorithm is to iteratively update the model λ such that the likelihood $p(X|\lambda) = \Pi_{x \in X} (p(x|\lambda))$ is maximized, i.e.

$$p(X|\lambda^{(t+1)}) > p(X|\lambda^{(t)}) . \tag{3.5}$$

Based on the estimation of $\hat{p}(x|\lambda^{(t)})$ at t-iteration (Eq. 3.2), the posterior probability for component i is given as

$$\Pr(i|x, \lambda^{(t)}) = \frac{w_i g(x|u_i^{(t)}, \Sigma_i^{(t)})}{\sum_{k=1}^{M} w_k g(x|u_k^{(t)}, \Sigma_k^{(t)})} . \tag{3.6}$$

Then, the next iteration $(t + 1)$ of EM will update the GMM model parameters using the following formulas:

$$w_i^{(t+1)} = \frac{1}{|X|} \sum_{x \in X} \Pr(i|x, \lambda^{(t)}) ; \tag{3.7}$$

$$u_i^{(t+1)} = \frac{\sum_{x \in X} \Pr(i|x, \lambda^{(t)}) x}{\sum_{x \in X} \Pr(i|x, \lambda^{(t)})} ; \tag{3.8}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_{x \in X} \Pr(i|x, \lambda^{(t)}) \left(x - u_i^{(t)}\right)^T \left(x - u_i^{(t)}\right)}{\sum_{x \in X} \Pr(i|x, \lambda^{(t)})} . \tag{3.9}$$

The initial model $\lambda^{(0)}$ of the GMM is typically derived by using some form of vector quantization estimation. The convergence condition of the algorithm can be set by thresholding the changes of the model. The GMM method models the data distribution by imposing a prior restriction on the structure of the model. On one hand, it can produce stable and accurate estimation if the model is properly set. On the other hand, it is not surprising that the GMM method would output a biased model if the predefined candidates do not include any good approximation of the truth.

Kernel Density Estimation

Kernel Density Estimation (KDE) [101] is a popular non-parametric approach for estimating the PDFs. It is closely related to histograms, but is enhanced with the properties of smoothness and continuity because of kernel functions used. For example, by using the Gaussian kernel,

$$k_\sigma(x_i,x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) , \tag{3.10}$$

the kernel density estimator for a given training collection X is

$$\hat{p}(x) = \frac{1}{|X|}\sum_{x_i \in X}\frac{1}{(2\pi\sigma^2)^{d/2}}k_\sigma(x, x_i) . \tag{3.11}$$

The parameter of KDE, such as the kernel bandwidth σ, can be optimized by k-fold cross-validation to maximize likelihood or log-likelihood [57]

$$\max\left\{\prod_{x\in X}(\hat{p}(x))\right\} = \max\left\{\sum_{x\in X}(\log\hat{p}(x))\right\} . \tag{3.12}$$

Because KDE does not make an assumption about the model of the density function, it is flexible to accommodate any complex data distribution. But the flexibility may produce unreliable estimation if there is not enough samples and the dimensionality is too high.

By using any density estimators to obtain $\hat{p}(x)$ and $\hat{p'}(x)$ separately, then the density ratio of them is easy to obtain by taking a division as $\hat{\beta}(x) = {\hat{p}'(x)}/{\hat{p}(x)}$. However, this naïve two-step density estimation based approach has been revealed to suffer from several problems [62]. First, the information from the given limited number of samples may be sufficient to infer the density-ratio, but insufficient to infer two probability density functions. Second, a small estimation error in the denominator can lead to a large error in the density-ratio. Lastly, the naïve approach would be highly unreliable for high-dimension problems because of the notable 'curse-of-dimensionality' in density estimation. Following the spirit of solving a problem succinctly and avoiding unnecessary steps to solve some more general problems, there are several well-known one-step density-ratio estimation methods being proposed, which are discussed in the following sections.

3.2.2 Kernel Mean Matching

The Kernel Mean Matching (KMM) algorithm [53,62] is a well-known algorithm for density ratio estimation based on infinite-order moment matching. The basic idea behind KMM is that two distributions are equivalent if and only if all moments are matched with each other. By making use of universal reproducing kernels, the infinite order moment matching is implicitly implemented. To be specific, it estimates density ratios by minimizing the Maximum Mean Discrepancy (MMD) [52] between the weighted distribution $p(x)$ and the distribution $p'(x)$ in a Reproducing Kernel Hilbert Space (RKHS) $\phi(x) : x \rightarrow \mathcal{F}$,

$$\mathrm{MMD}^2\left(\mathcal{F}, (\beta, p), p'\right) = \left\| E_{x\sim p(x)}\left[\beta(x)\cdot\phi(x)\right] - E_{x\sim p'(x)}\left[\phi(x)\right]\right\|^2 . \tag{3.13}$$

Theorem-1.2 and Lemma-1.3 in [53] state that if the kernel space is universal and $p'(x)$ is absolutely continuous with respect to $p(x)$, the solution $\beta(x)$ of Eq. 3.13 converges to $p'(x) = \beta(x)p(x)$.

Using empirical means of $\mathcal{S}$ and $\mathcal{S}'$ to replace the expectations, we can obtain a Quadratic Programming (QP) problem as

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \operatorname{argmin}_{\boldsymbol{\beta}}\left[\mathrm{MMD}^2\left(\mathcal{F}, (\beta, p), p'\right)\right] \\
&\approx \operatorname{argmin}_{\boldsymbol{\beta}}\left\|\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{\beta}_i\phi(x_i) - \frac{1}{n}\sum_{j=1}^{n}\phi(x'_j)\right\|^2 \\
&= \operatorname{argmin}_{\boldsymbol{\beta}}\left[\frac{1}{m^2}\sum_{i,j=1}^{m}\beta_i k(x_i, x_j)\beta_j - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\beta_i k(x_i, x'_j) + \frac{1}{n^2}\sum_{i,j=1}^{n}k(x'_i, x'_j)\right] \\
&= \operatorname{argmin}_{\boldsymbol{\beta}}\left[\frac{1}{2}\boldsymbol{\beta}^T K\boldsymbol{\beta} - \boldsymbol{k}^T\beta\right] ,
\end{aligned} \tag{3.14}$$

with respect to two constraints

$$\begin{gathered}
\boldsymbol{\beta}_i \in [0, b]\ i = 1, \ldots, m, \text{ and} \\
\left|\tfrac{1}{m}\textstyle\sum_{i=1}^{m}\boldsymbol{\beta}_i - 1\right| \leq \epsilon ,
\end{gathered}$$

Algorithm 1 KMM Algorithm [62]

```
Input: $\mathcal{S} = \{x_i | i = 1, \ldots, m\}$ , $\mathcal{S}' = \left\{x'_j | j = 1, \ldots, n\right\}$, $b, \epsilon, \sigma$
Output: $\hat{\boldsymbol{\beta}} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_m\right)^T$
Steps:
1: Compute $K$ (Eq. 3.15);
2: Compute $\boldsymbol{k}$ (Eq. 3.16);
3: Boundary constraint: $\mathbf{0} \leq \hat{\boldsymbol{\beta}} \leq b\mathbf{1}$ ;
4: Normalization constraint: $|\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\beta}_i - 1| \leq \epsilon$ ;
5: $\hat{\boldsymbol{\beta}} \leftarrow$ QP_solver $(K, \boldsymbol{k}, \epsilon, b)$ (Eq. 3.14);
```

where K is a kernel matrix defined on $\mathcal{S}$ as

$$K_{ij} = k(x_i, x_j),\ \{x_i, x_j \in \mathcal{S} | i, j = 1, \ldots m\}\ , \tag{3.15}$$

and $\boldsymbol{k}$ is a vector defined on the kernel between $\mathcal{S}$ and $\mathcal{S}'$ as

$$\boldsymbol{k}_i = \frac{m}{n} \sum_{j=1}^{n} k(x_i, x'_j),\ \{x_i \in \mathcal{S} | i = 1, \ldots m\}, \{x'_j \in \mathcal{S}' | j = 1, \ldots n\}\ . \tag{3.16}$$

The first constraint that limits the boundary of $\boldsymbol{\beta}_i \in [0, b]$ is to reflect the scope of discrepancy between $p(x)$ and $p'(x)$. The second constraint $|\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\beta}_i - 1| \leq \epsilon$ is a normalization factor over $\beta(x)$, since $p'(x) = \beta(x)p(x)$ should approximate a probability density function, where ϵ is a small number to control normalization precision.

For positive semi-definite kernel K, Eq. 3.14 formulates a convex QP problem with linear constraints. Therefore its global optimum can be obtained by using any existing QP solver [11]. The detailed steps of KMM are summarized in Algorithm 1.

Because the KMM outputs density ratios only at the sample points $\mathcal{S}$, it does not have out-of-sample ability for model/parameter selection. The original paper suggests a heuristic setting, the boundary $b = 1000$, $\epsilon = {\sqrt{m} - 1}/{\sqrt{m}}$, and kernel bandwidth σ as the median of pairwise sample distances [53]. But generally speaking, the parameter selection for KMM is still an open question.

3.2.3 Kullback-Leibler Importance Estimation Procedure

The Kullback-Leibler Importance Estimation Procedure (KLIEP) [110] models the density-ratio function as a linear combination of Gaussians as

$$\hat{\beta}(x) = \sum_{l=1}^{b} \alpha_l k(x, x_l) \ . \tag{3.17}$$

where x_l are data points taken from $\mathcal{S}'$ as reference points, and $k(\cdot, \cdot)$ is a Gaussian kernel function centered on these reference points. Usually for the feasibility of computation, a subset of $\mathcal{S}'$ is taken (i.e. $b \ll n$). The parameters $\alpha_1, \alpha_2, \ldots, \alpha_b \geq 0$ are to be learned from the two collections of data $\mathcal{S}$ and $\mathcal{S}'$.

The algorithm learning objective is to minimize the Kullback-Leibler divergence between the weighted density $\hat{\beta}(x)p(x)$ and the distribution density $p'(x)$ as

$$\begin{aligned} \mathrm{KL}\left[p'(x)|\hat{\beta}(x)p(x)\right] &= \int p'(x) \log \frac{p'(x)}{\hat{\beta}(x)p(x)} dx \\ &= \int p'(x) \log \frac{p'(x)}{p(x)} dx - \int p'(x) \log \hat{\beta}(x) dx \ . \end{aligned} \tag{3.18}$$

The first term is unrelated with respect to estimation of $\beta(x)$, so minimizing the Kullback-Leibler divergence is equivalent to maximizing the second term, defined as

$$J = \int p'(x) \log \hat{\beta}(x) dx \ . \tag{3.19}$$

Additional, because the $\hat{\beta}(x)p(x)$ is supposed to be close to a PDF $p'(x)$, a normalization constraint should be applied as

$$\int \hat{\beta}(x)p(x)dx = 1 \ . \tag{3.20}$$

Replacing the expectations in Eq. 3.18 and Eq. 3.20 with their empirical means, we

Algorithm 2 KLIEP Algorithm [110]

```
Input: $\mathcal{S} = \{x_i | i = 1, \dots, m\}$ , $\mathcal{S}' = \left\{x'_j | j = 1, \dots, n\right\}$, $b, \delta$
Output: $\hat{\beta}(x)$
Steps:
 1: Choose $b$ samples from $\mathcal{S}'$ as centers $\{x_l | l = 1, \dots, b\}$;
 2: Compute $A_{jl} \leftarrow k(x'_j, x_l)$ ;
 3: Compute $\xi_l \leftarrow \frac{1}{m}\sum_{i=1}^{m} k(x_i, x_l)$;
 4: Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{1}$;
 5: Initialize learning step $\delta, 0 < \delta \ll 1$;
 6: repeat
 7:     $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \delta A^T(\mathbf{1}./A\boldsymbol{\alpha})$;
 8:     $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + (1 - \boldsymbol{\xi}^T\boldsymbol{\alpha})\boldsymbol{\xi} / \left(\boldsymbol{\xi}^T\boldsymbol{\xi}\right)$;
 9:     $\boldsymbol{\alpha} \leftarrow \max(\mathbf{0}, \boldsymbol{\alpha})$;
10:     $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} / \left(\boldsymbol{\xi}^T\boldsymbol{\alpha}\right)$
11: until Convergence-of-$\boldsymbol{\alpha}$
12: $\hat{\beta}(x) = \sum_{l=1}^{b} \alpha_l k(x, x_l)$;
```

have the following optimization problem:

$$\begin{aligned} \hat{\boldsymbol{\alpha}} &= \arg\max_{\boldsymbol{\alpha}} \{J\} \\ &\approx \arg\max_{\boldsymbol{\alpha}} \left\{ \sum_{j=1}^{n} \log \left(\sum_{l=1}^{b} \alpha_l k(x'_j, x_l) \right) \right\} \qquad (3.21) \\ w.r.t. \quad & \frac{1}{n} \sum_{l=1}^{b} \alpha_l \sum_{i=1}^{m} k(x_i, x_l) = 1 \\ & \alpha_1, \alpha_2, \dots, \alpha_b \geq 0 \qquad (3.22) \end{aligned}$$

This formulates a convex problem with linear constraints and the global solution can be obtained through gradient searching. The KLIEP algorithm is listed in Algorithm 2.

The free parameters in KLIEP include the kernel bandwidth σ and the number of reference points b. Because the density ratio is modeled as a function, KLIEP can produce outputs for out of training samples and accordingly the parameters of the model can be tuned by Likelihood Cross-Validation (LCV) to optimize the objective value J (Eq. 3.19). Algorithm 3 summarizes the procedure of k-fold cross-validation for KLIEP to select an optimal model from candidates $\mathcal{M}$ by splitting $\mathcal{S}'$ into a section for model learning and a

Algorithm 3 Model Selection of KLIEP Algorithm [110]

Input: $\mathcal{S} = \{x_i | i = 1, \ldots, m\}$, $\mathcal{S}' = \left\{x'_j | j = 1, \ldots, n\right\}$, $\mathcal{M} = \{m_1, \ldots, m_s\}$
Output: $\hat{\beta}(x)$
Steps:

```
 1: Split $\mathcal{S}'$ into k-fold of disjoint subsets;
 2: for each model $m_i$ in $\mathcal{M}$ do
 3:     for each fold $j$ do
 4:         $\hat{\beta}(x) \leftarrow \text{KLIEP}(m_i, \mathcal{S}, \mathcal{S}' \backslash \mathcal{X}_j)$;
 5:         $\hat{J}(i, j) \leftarrow \frac{1}{|\mathcal{X}_j|} \sum_{x \in \mathcal{X}_j} \log \hat{\beta}(x)$;
 6:     end for
 7:     $\hat{J}(i) = \sum_j \hat{J}(i, j)$;
 8: end for
 9: $m^* \leftarrow \arg\max_{m_i \in \mathcal{M}} \left\{\hat{J}(i)\right\}$;
10: $\hat{\beta}(x) \leftarrow \text{KLIEP}(m^*, \mathcal{S}, \mathcal{S}')$;
```

section for model assessment.

3.2.4 Least Square Importance Fitting

Least Square Importance Fitting (LSIF) [69] is another model-based density-ratio estimation method, in which the density ratio function is also modeled as $\hat{\beta}(x) = \sum_{l=1}^{b} \alpha_l k(x, x_l)$ (the same as Eq. 3.17).

Different from the KLIEP algorithm, LSIF learns the parameter $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_b)^T$ by minimizing the squared loss of density-ratio function fitting:

$$\begin{aligned} \text{LS}\left[\hat{\beta}(x), \beta(x)\right] &= \frac{1}{2} \int \left(\hat{\beta}(x) - \beta(x)\right)^2 p(x) dx \\ &= \frac{1}{2} \int \hat{\beta}(x)^2 p(x) dx - \int \hat{\beta}(x) p'(x) dx + \frac{1}{2} \int \beta(x)^2 p(x) dx \quad (3.23) \end{aligned}$$

There are two variants of LSIF. One is the constrained Least Square Importance Fitting (cLSIF), which considers the non-negativity constraint on $\boldsymbol{\alpha}$ during the learning process. Another variant is the unconstrained Least Square Importance Fitting (uLSIF), which first outputs $\hat{\boldsymbol{\alpha}}$ by ignoring the non-negativity constraint and then modifies the output as $\tilde{\boldsymbol{\alpha}} = \max(\mathbf{0}, \hat{\boldsymbol{\alpha}})$. Substituting the learned $\tilde{\boldsymbol{\alpha}}$ into Eq. 3.17, the LSIF/uLSIF algorithm produces a function to model the density ratios.

The uLSIF algorithm involves solving linear equations only and the solution can be obtained analytically. So, the computation efficiency and stability of uLSIF make it well accepted in realistic applications [54, 59]. In the following we describe the details of uLSIF only.

The last term of Eq. 3.23 does not depend on $\hat{\beta}(x)$, which can be ignored. Then, substituting the model function $\hat{\beta}(x) = \sum_{l=1}^{b} \alpha_l k(x, x_l)$, approximating expectations with empirical means, and adding regularization term $\frac{\lambda}{2} \sum_{l=1}^{b} \boldsymbol{\alpha}_l^2$ to penalize the complexity of learned model, this leads to the following unconstrained optimization problem:

$$\hat{\boldsymbol{\alpha}} = \operatorname{argmin}_{\boldsymbol{\alpha}} \left[\frac{1}{2} \sum_{l,l'=1}^{b} \boldsymbol{\alpha}_l \boldsymbol{\alpha}_{l'} H_{ll'} - \sum_{l=1}^{b} \boldsymbol{\alpha}_l \boldsymbol{h}_l + \frac{\lambda}{2} \sum_{l=1}^{b} \boldsymbol{\alpha}_l^2 \right] , \tag{3.24}$$

where

$$H_{ll'} = \frac{1}{m} \sum_{i=1}^{m} k(x_i, x_l) k(x_i, x_{l'}) , \tag{3.25}$$

$$\boldsymbol{h}_l = \frac{1}{n} \sum_{j=1}^{n} k(x'_j, x_l) . \tag{3.26}$$

We can see Eq. 3.24 is an unconstrained convex quadratic problem, the global solution can be computed analytically as

$$\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_b)^T = (H + \lambda I)^{-1} \boldsymbol{h} . \tag{3.27}$$

To avoid the negativity of output, the final result is required to modify the output of Eq. 3.27 to

$$\tilde{\boldsymbol{\alpha}} = \max(\mathbf{0}, \hat{\boldsymbol{\alpha}}) . \tag{3.28}$$

Substituting the learned $\tilde{\boldsymbol{\alpha}}$ into Eq. 3.17, the uLSIF algorithm then produces a function to model the density ratios. Its model selection is also possible through the likelihood cross-validation, for example using k-fold cross-validation to select kernel bandwidth σ and the regularization parameter λ. The details of uLSIF with cross-validation steps are presented in Algorithm 4.

Algorithm 4 uLSIF Algorithm [69]

```
Input: $\mathcal{S} = \{x_i|i = 1, \ldots, m\}$ , $\mathcal{S}' = \left\{x'_j|j = 1, \ldots, n\right\}$ , $b$
Output: $\hat{\beta}(x)$
Steps:
 1: Choose $b$ samples from $\mathcal{S}'$ as centers $\{x_l|l = 1, \ldots, b\}$;
 2: for each $\langle\sigma, \lambda\rangle$ from candidates do
 3:     Proceed cross-validation:
 4:     Compute $H$ (Eq. 3.25);
 5:     Compute $\boldsymbol{h}$ (Eq. 3.26);
 6:     $\tilde{\boldsymbol{\alpha}} \leftarrow \max\left(\mathbf{0}, (H + \lambda I)^{-1} \boldsymbol{h}\right)$;
 7:     Compute objective function $\mathrm{LS}(\sigma, \lambda)$ (Eq. 3.23);
 8: end for
 9: $(\sigma^*, \lambda^*) \leftarrow \mathrm{argmin}\,(\mathrm{LS}(\sigma, \lambda))$;
10: Compute the final $H, \boldsymbol{h}$ with selected parameters $\sigma^*, \lambda^*$;
11: Compute the final $\tilde{\boldsymbol{\alpha}} \leftarrow \max\left(\mathbf{0}, (H + \lambda I)^{-1} \boldsymbol{h}\right)$;
12: $\hat{\beta}(x) \leftarrow \sum_{l=1}^{b} \tilde{\boldsymbol{\alpha}}_l k(x, x_l)$;
```

3.3 Applications

In this section, we review several important data analysis tasks in which the concept of density ratio has potential contributions. This includes the covariate shift adaptation [105, 109], outlier detection [59], safe semi-supervised learning [71, 114], and some others [10, 40, 54, 70].

3.3.1 Covariate Shift Adaptation

Covariate shift is a particular situation in the supervised learning setting of domain adaptation, where the test data and training data are distributed differently in the marginal input space, referring Section 2.2.1 in Chapter 2. Usually the covariate shift happens in biased sample selection scenarios.

Within the risk minimization framework, the general purpose of a supervised learning problem is to minimize the expected risk of

$$R(h, p, l) = \iint l(x, y, h)\, p(x, y)\, dxdy\,, \tag{3.29}$$

where h is a learned model, $l(x, y, h)$ is a loss function for the problem with a joint distribution $p(x, y)$.

If we are facing the case where the training data distribution $p_{tr}(x, y)$ differs from the test data distribution $p_{ts}(x, y)$, in order to obtain the optimal model in the test domain h^*_{ts}, we can derive the following reweighting scheme:

$$\begin{aligned} h^*_{ts} &= \arg\min_{h \in H} R_{ts}\left(h, p_{ts}(x, y), l(x, y, h)\right) \\ &= \arg\min_{h \in H} \iint l(x, y, h)\, p_{ts}(x, y)\, dxdy \\ &= \arg\min_{h \in H} \iint l(x, y, h)\, \frac{p_{ts}(x, y)}{p_{tr}(x, y)} p_{tr}(x, y) dxdy \\ &= \arg\min_{h \in H} R_{tr}\left(h, p_{tr}(x, y), \frac{p_{ts}(x, y)}{p_{tr}(x, y)} l(x, y, h)\right) . \end{aligned} \tag{3.30}$$

Further, covariate shift assumes that the conditional distributions are the same across the training and test data (i.e. $p_{ts}(y|x) = p_{tr}(y|x)$), but that the marginal distributions are different. Hence h^*_{ts} can be expressed as follows:

$$\begin{aligned} h^*_{ts} &= \arg\min_{h \in H} R_{tr}\left(h, p_{tr}(x, y), \frac{p_{ts}(x)}{p_{tr}(x)} l(x, y, h)\right) \\ &= \arg\min_{h \in H} R_{tr}\left(h, p_{tr}(x, y), \beta(x) l(x, y, h)\right) . \end{aligned} \tag{3.31}$$

Having the weighted training instances, there are numerous cost-sensitive learning algorithms that can be applied. Instead of minimizing the loss of misclassification, the cost-sensitive learning aims at minimizing the instance-dependent cost of wrong prediction [112, 129]. For example, Support Vector Machines [21] can naturally embed weighted samples in the training process as

$$\min_{h, \xi} \frac{1}{2} \|h\|^2_{\mathcal{F}} + c \sum_{i=1}^{n_{tr}} \beta(x_i) \xi(x_i) , \tag{3.32}$$

subject for all $(x_i, y_i) \in \mathcal{S}_{tr}$

$$\langle y_i, h(x_i) \rangle \geq 1 - \xi(x_i),\ \xi(x_i) \geq 0 ,$$

where $\mathcal{F}$ is the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel function, c is the trade off between the separation margin and the empirical error.

In comparison to the traditional SVM, the only difference in Eq. 3.32 is the weights $\beta(x_i)$ being added into the training error penalty term. It treats the training samples differently by considering their importance weights.

3.3.2 Inlier-based Outlier Detection

Hido et al. [59] proposed an inlier-based outlier detection method and defined the inlier score by using density ratios between the normal data and the test data as

$$\text{InlierScore}(x) = \frac{p_{\text{rf}}(x)}{p_{\text{ts}}(x)}\,, \tag{3.33}$$

where $p_{\text{rf}}(x)$ and $p_{\text{ts}}(x)$ are PDFs of reference normal data and the test data, respectively.

For the regions that have small inlier scores, it means the normal data density is low and the test data density is high. Thus, using the relative densities a novel (outlier) instance can be identified if its inlier score is below a threshold.

3.3.3 Safe Semi-supervised Learning

Semi-supervised Learning (SSL) is a learning paradigm that exploits the limited number of labeled data and a large number of unlabeled data to learn a strong model. The SSL algorithms improve generalization performance under their assumptions regarding population distribution and model structure. But if the associated assumptions do not hold, semi-supervised learning may even degrade estimation accuracy in comparison to supervised learning methods that simply ignore the unlabeled data.

In [71] Kawakita and Kanamori proposed a *safe* semi-supervised learning algorithm by employing a density-ratio estimator to weight the labeled samples. It is highlighted as a safe SSL since it was proved to be no worse than the supervised learning regardless of the model assumptions. Another work from Tan and Zhu [114] extended the idea of safe SSL and proposed an ensemble method by applying density ratios in the bagging manner. They proved that the density-ratio bagging method achieves less asymptotic variance than bagging and requires only weak semi-supervised learning assumptions.

Besides the three important tasks being discussed above, the concept of density ratio has also been studied for other data analysis tasks, such as change point detection [80], dimension reduction [113], and privacy preserving data publishing [40].

3.4 Summary

In this chapter, we described the problem of sample importance weighting according to density ratios and reviewed popular techniques for estimating density ratios. Table 3.1 summarizes the properties of these methods.

The two-step approach is the most straightforward as it follows the definition of density ratio, which estimates two PDFs first and then takes a ratio. Gaussian Mixture Model (GMM) and Kernel Density Estimation (KDE) are two popular methods for density function estimation. The KDE method is an especially flexible method for modeling complex data distributions and its use is computationally efficient due to its analytical form of solution. The model selection can be performed by k-fold likelihood cross-validation.

Instead of the two-step approach, the one-step approach is to estimate a density-ratio without going through the explicit estimation of two density functions. It follows the spirit of solving a problem succinctly and avoiding unnecessary steps that involve more general problems. The KMM is a milestone method that estimates a density-ratio in one-step. The basic idea of KMM is to use infinite-order moment matching by making use of universal reproducing kernels. KMM minimizes the Maximum Mean Discrepancy in the reweighted space and formulates as a quadratic convex problem. The main difficulty with KMM is the lack of a parameter selection mechanism, because the output is the values at the sampling points and there is no model being produced.

KLIEP, cLSIF, and uLSIF however model the density-ratio function as a mixture of multi-Gaussians and the mixture weights are learned by formulating different learning objectives. Because they produce the model function of the density-ratio as output, the density-ratio values for unseen data points can be obtained. Therefore, model selection by cross-validation can be implemented. This is a great advantage of these methods over KMM.

KLIEP is formulated to minimize the Kullback-Leibler divergence between the weighted data collection and the target data collection. The objective is a convex function with linear constraints and can be solved by gradient descent searching. Its computation is expensive due to the non-linear objective function used during search iterations. The output of KLIEP is not only the values at the sampling points, but also a density-ratio function. Therefore, a variant of cross-validation can be used for selecting the parameters in the model.

Constrained LSIF (cLSIF) and Unconstrained LSIF (uLSIF) have the objective to minimize the least-squared error of a density-ratio function fitting, and add a regularization

Table 3.1: State-of-the-art methods for density-ratio estimation.

Approach	**Method**	**Objective**	**Algorithmic Solution**	**Model Selection**	**Output**
Two-step approach	GMM	Maximize log-Likelihood	Expectation-Maximization	Available	Two density functions
	KDE	Maximize log-Likelihood	Analytic	Available	Two density functions
Moment matching	KMM	Minimize MMD	Convex quadratic programming	Not available	Density-ratio values at sampling points
Density function fitting	KLIEP	Minimize KL-divergence	Gradient descent	Available	Density-ratio function
Density-ratio function fitting	cLSIF	Minimize least-squared error	Convex quadratic programming	Available	Density-ratio function
	uLSIF	Minimize least-squared error	Analytic	Available	Density-ratio function

term to penalize model complexity. Constrained LSIF incorporates the non-negative constraint. So its solution can be computed using convex quadratic programming, which usually is computational demanding. Unconstrained LSIF initially drops the non-negative constraint, then enforces non-negativity over the results as an approximation. Since uLSIF is a pure quadratic problem without constraints, its solution can be analytically computed. This greatly improves computation efficiency, and many empirical studies reveal this approximation has small effects on accuracy.

The estimation of density ratios has shown potential in various machine learning and data mining applications. It has been applied for covariate shift adaptation, outlier detection, semi-supervised learning, and other challenging data analysis tasks.

Chapter 4

Discriminative Density Ratio for Domain Adaptation of Classification Problems

Domain adaptation deals with the challenging problem that arises from the distribution difference between training and test data. An effective approach is to reweight the training samples to minimize the distribution discrepancy. This chapter presents a novel Discriminative Density-Ratio (DDR) method for learning adaptive classifiers. This approach combines three learning objectives: minimizing generalized risk on the reweighted training data, minimizing class-wise distribution discrepancy, and maximizing the separation margin of the test data. To solve the DDR problem, two algorithms are presented based on the Block Coordinate Update (BCU) method.

The chapter is organized as follows. Section 4.1 introduces the domain adaptation problem and reviews related work. Section 4.2 presents the Discriminative Density-Ratio (DDR) framework. Section 4.3 describes two BCU algorithms for solving the DDR problem. The experiments and results are discussed in Section 4.4.

4.1 The Domain Adaptation Problem

There are many real world applications in which the test data exhibit distributional differences relative to the training data. For example, when building an action recognition system, training samples are collected in a university lab, where young people make up a high percentage of the population. When the system is intended to be applied in reality, it is likely that a more general population model will be required. In such cases, traditional machine learning techniques usually encounter performance degradation because their models are fitted towards minimizing the generalized error for the training data. So, it is important that the learning algorithms are able to demonstrate some degree of adaptivity to cope with distribution changes. This necessity has resulted in intensive research under the name of domain adaptation [7,25,30]. Some closely related work uses other terms such as transfer learning [93, 126], concept drift [48, 123], and covariate shift [81, 110].

Theoretical analysis of domain adaptation divides the type of distribution changes into covariate shift, prior change, and concept drift. However, realistic learning scenarios usually simultaneously include more than one type of change.

4.1.1 Problem Definition

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a d-dimension input space and $\mathcal{Y}$ be a finite set of class labels. In a typical supervised learning, given n labeled samples $\mathcal{S} = \{(x_i, y_i)|i = 1, \ldots, n\}$, we want to learn a mapping function $h : \mathcal{X} \to \mathcal{Y}$ which has minimal prediction error for unseen samples. This assumes that test data comes from the same distribution as the training data, which defines the meaning of a *domain*.

In the context of domain adaptation, the same distribution assumption does not hold. Instead, we have n_{tr} labeled training collection $\mathcal{S}_{tr} = \langle \mathcal{X}_{tr}, \mathcal{Y}_{tr} \rangle = \{(x_i, y_i)|i = 1, \ldots, n_{tr}\}$ which is drawn from a distribution $p_{tr}(x, y)$. The n_{ts} number of unlabeled test samples $\mathcal{S}_{ts} = \mathcal{X}_{ts} = \left\{x'_j|j = 1, \ldots, n_{ts}\right\}$ are from a different but related distribution $p_{ts}(x, y)$. In this scenario, the learning setting is exposed to two different domains: the training domain and the test domain (sometimes they are also referred to as the source and target domains). The domain adaptation problem aims to build an adaptive learner that can learn a model $\hat{h}_{ts}$ to fit the test data distribution using the labeled training data $\mathcal{S}_{tr}$ and the unlabeled test data $\mathcal{S}_{ts}$.

The obvious constraint which should be satisfied is the relatedness between the training data and test data. Formal studies reveal that the problem is intractable when the hypothesis set does not contain any candidate achieving a good performance on the training set [8], since the learner has to rely on the labeling function in the training data and then transfer the knowledge to the test data.

4.1.2 Related Work

Because of the importance of the problem and the need in practice, the domain adaptation has attracted a great deal of research in recent years. In the literature there are three directions of work being proposed: dynamic ensemble, feature transformation, and sample reweighting.

Dynamic ensemble. The ensemble method is a learning paradigm that combines a set of base components to model a complex problem through proper decision fusion rules. This method represents a comprehensively studied approach in the streaming data concept drift scenario [74, 75, 90]. The adaptivity is achieved by dynamic weight assignment at the decision layer. The weight of a component in the ensemble is usually decided through evaluating its performance in the most recent batch of data, while assuming the streaming data can continuously obtain the labeling information. For the cross-dataset task, Dai et al. [29] proposed an AdaBoost-style algorithm for model adaptation. It assumes a number of labeled samples available in the target dataset, and uses different weighting strategies for the training data and labeled target data in the boosting iteration.

Feature transformation. Another approach to domain adaptation is based on the assumption of the existence of a domain-invariant feature space. This approach depends on defining and quantifying a transformation to find such a feature space, and then the adaptation can be accomplished by learning a model on the new space. Pan et al. [92] proposed the Transfer Component Analysis (TCA) method, which learns the feature transformation to produce a set of common transfer components across domains in a reproducing kernel Hilbert space. In [41], shared latent structure features are learned for the problem of transferring knowledge across information networks. Blitzer et al. [9] proposed the Structural Correspondence Learning (SCL) method, which learns a common feature space by identifying correspondences among features from different domains. In [25, 50], the deep learning approach is proposed to generate robust cross-domain feature representations using the output of the intermediate layers. The work of Chen et al. [24] is along the line of feature selection, which gradually finds a subset of features that are suitable for the target domain only, instead of finding common features for both the target and source domains.

Sample reweighting. Sample reweighting approach for domain adaptation assigns sample-dependent weights for the training data with the objective of minimizing the distribution discrepancy between the training data and test data in the reweighted space. Having the weighted training samples, the cost-sensitive learning methods can be applied to produce a model that adapts to the test data distribution. Considering that ensemble-based methods need labeled test domain data while feature transformation-based methods are usually developed for a specific application, the sample reweighting is an effective and general approach. The sample reweighting can make use of the advances in many cost-sensitive learning algorithms, ranging from the single model such as cost-sensitive SVM [84] to the ensemble model such as cost-sensitive boosting [112]. The sample weighting mechanism is the core of the problem, and its estimation is formulated as the density ratio between the probability densities of the test and training data [67, 86, 109].

4.2 Discriminative Density Ratio

Existing work on using the reweighting strategy for domain adaptation is based on the simplified assumption that $p_{ts}(y|x) = p_{tr}(y|x)$ and on the estimation of the density ratio of the marginal distributions as $\beta(x) = {}^{p_{ts}(x)}/_{p_{tr}(x)}$. However, realistic domain adaptation problems are more complex than the above assumption. According to Bayes' rule, the prior, posterior, marginal, likelihood, and joint distributions are tightly related as $p(x, y) = p(y|x)p(x) = p(x|y)p(y)$. The actual learning settings usually cause more than one type of distribution change simultaneously.

Focusing on the classification tasks, the objective is to discriminatively separate the instances into different classes. However, in the conventional weighting approach dealing with adaptation, the distribution matching is performed on the whole input space. In other words, the existing algorithms focus on matching the training and test distributions without considering to preserve the separation between classes in the reweighted space.

Moreover, the effectiveness of conventional density-ratio estimation approach is limited by another constraint, the support condition (i.e. $\forall x,\ p_{tr}(x) = 0 \Rightarrow p_{ts}(x) = 0$) [62, 91]. The model, therefore, cannot generalize well for regions where $p_{ts}(x) \neq 0$ but $p_{tr}(x) = 0$.

These two problems hold back the effectiveness of the weighting methods in the domain adaptation problem severely, especially for classification tasks. Several studies have reported this problem [28, 53], but none of them have presented a clear solution.

Motivated by these observations, we propose a Discriminative Density-Ratio (DDR) approach to learn the weights of training data discriminatively by estimating the density ratio of joint distributions in a class-wise manner to preserve the separations between classes. The DDR model aims to achieve the objectives of 1) approximating test domain risk with the reweighted training data according to joint distributions, 2) minimizing the distribution discrepancy between the training and test data in a class-wise manner, and 3) guiding the decision boundary to the sparse regions of the test data.

4.2.1 Learning Objectives

Objective 1: Minimization of approximated test domain risk. First, we discuss the situation of supervised learning where there is no distribution change between the training and test data. The general purpose of supervised learning is to minimize the expected risk

$$R\left(h, p(x,y), L\left(x,y,h\right)\right) = \iint L\left(x,y,h\right) p(x,y) dxdy\ , \tag{4.1}$$

where h is the model hypothesis, $L\left(x,y,h\right)$ is a loss function, and $p(x,y)$ is the joint distribution over x and y.

Given the presence of distribution changes between the training and test data, i.e. $p_{ts}(x,y) \neq p_{tr}(x,y)$, we will seek to obtain the optimal model in the test domain by approximating the test domain risk using the following reweighting scheme:

$$\begin{aligned} R_{ts}(h, p_{ts}(x,y), L(x,y,h)) &= \iint L\left(x,y,h\right) p_{ts}(x,y) dxdy \\ &= \iint L\left(x,y,h\right) \frac{p_{ts}(x,y)}{p_{tr}(x,y)} p_{tr}(x,y) dxdy \\ &= R_{tr}\left(h, p_{tr}(x,y), \frac{p_{ts}(x,y)}{p_{tr}(x,y)} L(x,y,h)\right)\ . \end{aligned} \tag{4.2}$$

Defining weights to reflect the joint distribution ratios $w(x,y) = \frac{p_{ts}(x,y)}{p_{tr}(x,y)}$, we have

$$R_{ts} = R_{tr}(h, p_{tr}(x,y), w(x,y)L(x,y,h))\ . \tag{4.3}$$

With n_{tr} observed training samples $\mathcal{S}_{tr} = \{(x_i, y_i)|i = 1, \ldots, n_{tr}\}$ from $p_{tr}(x,y)$ and using a regularized risk scheme, the test domain risk can be approximated as

$$\begin{aligned} \tilde{R}_{ts} &\approx \hat{R}_{tr}(h, \mathcal{S}_{tr}, w(x,y)L(x,y,h)) \\ &= \frac{1}{n_{tr}} \sum_{(x_i,y_i)\in\mathcal{S}_{tr}} w(x_i,y_i)L(x_i,y_i,h) + \lambda\Omega(h)\ , \end{aligned} \tag{4.4}$$

where $\Omega(h)$ is the model complexity which serves as a regularizer to avoid overfitting to the training data, and λ is the trade-off parameter.

Objective 2: Minimization of class-wise distribution discrepancy. The conventional sample reweighting approach assumes that the posterior distributions are unchanged ($p_{ts}(y|x) = p_{tr}(y|x)$) and simplifies the weights as the density ratio of the marginal distributions of x as

$$w(x, y) = \frac{p_{ts}(x, y)}{p_{tr}(x, y)} \approx \frac{p_{ts}(x)}{p_{tr}(x)} . \tag{4.5}$$

Instead of aggressively reweighting training samples by the density ratios of the marginal distributions, our approach is to preserve the separations between classes by estimating the density ratio of joint distributions in a class-wise manner. We decompose the joint distribution from the perspective of class likelihood and class prior as

$$w(x, y) = \frac{p_{ts}(x, y)}{p_{tr}(x, y)} = \frac{p_{ts}(x|y)p_{ts}(y)}{p_{tr}(x|y)p_{tr}(y)} = \frac{p_{ts}(x|y)}{p_{tr}(x|y)} \cdot \frac{p_{ts}(y)}{p_{tr}(y)} . \tag{4.6}$$

Let $\beta(x, y) = \frac{p_{ts}(x|y)}{p_{tr}(x|y)}$ be the density ratio of class conditional distributions between the same class, and $\gamma(y) = \frac{p_{ts}(y)}{p_{tr}(y)}$ be the ratio of priors. Then, Eq. 4.6 can be written as

$$w(x, y) = \beta(x, y)\gamma(y) \tag{4.7}$$

However, the fact that the test data do not have label information means that estimating β and γ directly is not possible. The solution is to use the current model prediction on the test data $\mathcal{X}_{ts}$ to estimate β and γ. The details are given in Section 4.3.2. Here, the objective is to minimize the following class-wise distribution discrepancy

$$\begin{aligned} &\mathrm{Dcw}(\beta(x, y), p_{tr}(x, y), p_{ts}(x), h) \\ &\quad = \sum_{c \in \mathcal{C}} \mathrm{D}\left[\beta(x, y = c)p_{tr}(x|y = c), p_{ts}(x|y = c)\right] \\ &\quad = \sum_{c \in \mathcal{C}} \mathrm{D}\left[\beta(x, y = c)p_{tr}(x|y = c), \frac{p_{ts}(y = c|x)}{p_{ts}(y = c)} p_{ts}(x)\right] , \end{aligned} \tag{4.8}$$

where $p_{ts}(y = c|x)$ and $p_{ts}(y = c)$ are the posterior and prior of the test data estimated by the current model h, and

$$\mathrm{D}\left[\beta(x, y = c)p_{tr}(x|y = c), p_{ts}(x|y = c)\right]$$

is the distribution discrepancy for class c between the weighted training data $\beta(x, y = c)p_{tr}(x|y = c)$ and the test data $p_{ts}(x|y = c) = \frac{p_{ts}(y=c|x)}{p_{ts}(y=c)}p_{ts}(x)$.

With the training and test collection $\mathcal{S}_{tr}$ and $\mathcal{X}_{ts}$, the empirical class-wise distribution discrepancy can be approximated as

$$\begin{aligned}
&\mathrm{Dcw}(\beta(x,y), \mathcal{S}_{tr}, \mathcal{X}_{ts}, h) \\
&= \sum_{c \in \mathcal{C}} \mathrm{D}\left[\beta \mathcal{X}_{tr}|_{y_{tr}=c}, \frac{p_{ts}(y = c|x' \in \mathcal{X}_{ts})}{\sum_{x' \in \mathcal{X}_{ts}} p_{ts}(y = c|x')} \mathcal{X}_{ts}\right] .
\end{aligned} \tag{4.9}$$

Using different measures to express the distribution discrepancies will lead to different density-ratio estimation algorithms (see Chapter 3). For example, using Least Square Error (LSE) as the objective function results in the uLSIF-based algorithm to solve the class-wise density-ratio estimation.

Objective 3: Maximization of test data margin. For the shifted but unknown distributions, we also intend to simultaneously force the classification boundary to lie at sparse regions of the unlabeled test data. This means that it is preferable to maximize the test data margin. Making use of this characteristic over the test data can alleviate the model generalization limits on the unsupported regions where $p_{ts}(x) \neq 0$ but $p_{tr}(x) = 0$.

Maximizing the test data margin coincides with minimizing the margin loss over the test data. As a result, the idea of hinge loss from semi-supervised learning [68] can be used to express the margin loss, which is defined as

$$\mathrm{MarginLoss}(h, \mathcal{X}_{ts}) = \sum_{x'_j \in \mathcal{X}_{ts}} \max\left(0, 1 - |h(x'_j)|\right) , \tag{4.10}$$

where $h(x'_j)$ is the decision value of the model output over the given test samples.

4.2.2 DDR Optimization Problem

Combining the aforementioned three objectives, we formulate the *Discriminative Density-Ratio (DDR) Optimization Problem* as:

$$\begin{aligned}
\{h^*_{ts}, w^*\} \quad &= \quad \mathrm{argmin}_{h,\beta,\gamma,w}\{\hat{R}_{tr}(h, \mathcal{S}_{tr}, wL(x, y, h)) \\
&\qquad +\lambda_1 \mathrm{Dcw}\left(\beta(x, y), \mathcal{S}_{tr}, \mathcal{X}_{ts}, h\right) + \lambda_2 \mathrm{MarginLoss}\left(h, \mathcal{X}_{ts}\right)\} \\
s.t. \quad &\qquad w = \beta(x, y)\gamma(y) ,
\end{aligned} \tag{4.11}$$

where λ_1 and λ_2 are trade-off parameters to balance the importance of the three terms.

The DDR problem is not trivial to solve since the first two terms are convex and the last term is concave. We will present two effective solutions to the DDR problem in the next section.

4.3 Solutions via Block Coordinate Update

Having proposed the DDR model, we need practical solutions for the complicated optimization problem of Eq. 4.11. In this section, we present two effective algorithms to solve the optimization problem using the block coordinate update method [96, 117]. Block Coordinate Update (BCU) is an iterative procedure that simplifies a complex problem into several solvable blocks.

4.3.1 Preliminaries on BCU

Consider the following general optimization problem

$$\min_{x \in \mathcal{P}} F(x_1, \ldots, x_s) \;, \tag{4.12}$$

where variable x can be decomposed into s blocks of variables $x = (x_1, \ldots, x_s)$, and the set $\mathcal{P}$ is the feasible solution points.

Due to the complexity of function F, in many cases it is difficult to solve the optimization problem in terms of updating all elements of x at the same time. The Block Coordinate Update (BCU) method cyclically minimizes F over a single block of variables while fixing the remaining blocks at their last updated values. To be more specific, at iteration t, the block of variables x_i is updated by solving the following sub-problem:

$$\begin{aligned} x_i^{(t)} \;&=\; \operatorname{argmin}_{x_i \in \mathcal{P}_i} \left[F(x_1^{(t)}, \ldots, x_{i-1}^{(t)}, x_i, x_{i+1}^{(t-1)}, \ldots, x_s^{(t-1)}) \right] , \\ & \quad\; i = 1, 2, \ldots s \;. \end{aligned} \tag{4.13}$$

Usually the sub-problems decomposed by BCU are expected to be computationally feasible and efficient in comparison with the original problem. The simplicity of implementation and the advances in theoretical aspects have led to the wide use of BCU in many practical applications, such as sparse dictionary learning [3] and nonnegative matrix factorization [125]. Our proposed DDR problem also relies on the BCU technique, as discussed in the following sections.

4.3.2 Alternating Between Semi-supervised Learning and Class-wise Distribution Matching

Referring to Eq. 4.11, if we choose to use the hinge loss for the test data and combine the first and last terms together, this will create a semi-supervised learning setting. This section describes a BCU algorithm to solve the DDR problem by alternating between the use of a Transductive Support Vector Machine (TSVM) as the semi-supervised learner and uLSIF as the distribution matching method.

Semi-supervised Learning using Sample Weighted Transductive SVM

In each iteration of BCU, we first fix the sample weights w, β, γ and try to update the prediction model h. If using a cost-sensitive Support Vector Machine (SVM) as the learner, the optimization of DDR can be simplified as

$$\begin{aligned} \hat{h}_{ts} = \arg\min_h [\frac{1}{2}\|h\|^2_{\mathcal{K}} \quad &+ \quad c_1 \sum_{(x_i,y_i)\in\mathcal{S}_{tr}} w_i L(x_i, y_i, h) \\ &+ \quad \lambda_2 \sum_{x'_j\in\mathcal{X}_{ts}} \max(0, 1 - \left|h(x'_j)\right|)] \,. \end{aligned} \tag{4.14}$$

Eq. 4.14 leads precisely to the formulation of Semi-supervised SVM (S3VM) [23]. Additionally, it is embedded with sample-dependent weights of the labeled training samples. S3VM is a well-known semi-supervised learning method that learns a hyperplane with maximized class margin while penalizing the hinge loss of labeled samples and the unlabeled samples simultaneously. In Eq. 4.14, $L(\cdot)$ is the loss function defined over the training set $\mathcal{S}_{tr}$, and w_i's are the sample-dependent weights. The parameters c_1 and λ_2 are the trade-off parameters between model complexity and empirical loss on the labeled and unlabeled data, respectively.

The Transductive SVM (TSVM) is a successful example that solves a S3VM using a heuristic searching strategy, named simulated annealing [68]. It starts with a small value of λ_2 and gradually increases its value. In comparison to a conventional TSVM,

Algorithm 5 Sample-Weighted Transductive SVM [68, 79]

```
Input: $S_{tr}, X_{ts}, w_{tr}, c_1, \lambda_2$
Output: $h_{ts}$
Steps:
1: $h_{ts} = \text{svm-train}(S_{tr}, w_{tr} * c_1)$;
2: $Y_{ts} = \text{svm-predict}(X_{ts}, h_{ts})$;
3: $c_2 = 10^{-5}$;
4: while $c_2 <= \lambda_2$ do
5:     $h_{ts} = \text{svm-train}((S_{tr}, \langle X_{ts}, Y_{ts} \rangle), (w_{tr} * c_1, c_2))$;
6:     $Y_{ts} = \text{local-search}(h_{ts}, Y_{ts})$;
7:     $c_2 = c_2 * 2$;
8: end while
9: Return $h_{ts}$;
```

the only difference in the formulation of Eq. 4.14 is the sample-dependent weights, which are embedded into the loss of training samples. Algorithm 5 shows the details of the sample-weighted TSVM algorithm.

Class-wise Distribution Matching using uLSIF

Another block of the BCU includes fixing the model h and aiming to update the weights β, γ, and w for the training samples.

γ is the ratio of class priors. Following the idea of Chan and Ng [18], γ can be estimated with the posteriors of test samples using the current model h as

$$\gamma(y) = \frac{p_{ts}(y)}{p_{tr}(y)} = \frac{\frac{1}{n_{ts}} \sum_{x'_j \in \mathcal{X}_{ts}} \hat{p}_{ts}(y|x'_j)}{\frac{1}{n_{tr}} \sum_{(x_i, y_i) \in \mathcal{S}_{tr}} 1(y_i = y)} . \tag{4.15}$$

Now we discuss the estimation of β based on class-wise density-ratio estimation. In the case of a fixed model h, the optimization of DDR can be simplified as:

$$\begin{aligned}\hat{\beta}(x,y) &= \arg\min_{\beta}\{c_1 \sum_{(x_i,y_i)\in\mathcal{S}_{tr}} \beta(x_i,y_i)\gamma(y_i)L(x_i,y_i,h) + \lambda_1 \mathrm{Dcw}\,(\beta,\mathcal{S}_{tr},\mathcal{X}_{ts},h)\} \\ &= \arg\min_{\beta}\{\sum_{c\in\mathcal{C}}[\frac{c_1}{\lambda_1} \sum_{(x_i,y_i)\in\mathcal{S}_{tr}\wedge y_i=c} \beta(x_i,y_i)\gamma(y_i)L(x_i,y_i,h) \\ &\qquad + \mathrm{D}[(\mathcal{X}_{tr},\beta)\mid_{y_{tr}=c}, \frac{p_{ts}(y=c|x'\in\mathcal{X}_{ts})}{\sum_{x'\in\mathcal{X}_{ts}} p_{ts}(y=c|x')}\mathcal{X}_{ts}]]\}\,, \end{aligned} \tag{4.16}$$

where the second term estimates the density ratios for the training data of each class. When we use the Least-Square-Error (LSE) [69] as the distribution discrepancy to be minimized, Eq. 4.16 is expressed as

$$\begin{aligned}\arg\min_{\beta} \quad & \sum_{c\in\mathcal{C}}[\frac{c_1}{\lambda_1} \sum_{(x_i,y_i)\in S_{tr}\wedge y_i=c} (\beta(x_i,y=c)\gamma(y_i)L(x_i,y_i,h)) \\ & + \frac{1}{2n_{tr}^{(c)}} \sum_{(x_i,y_i)\in S_{tr}\wedge y_i=c} \beta(x_i,y=c)^2 \\ & - \frac{1}{\sum_{x'_j\in\mathcal{X}_{ts}} p_{ts}(y=c|x'_j)} \sum_{x'_j\in\mathcal{X}_{ts}} [p_{ts}(y=c|x'_j)\beta(x'_j,y=c)]]\,, \end{aligned} \tag{4.17}$$

where $n_{tr}^{(c)}$ is the number of training samples from class c.

Eq. 4.17 can be divided into the estimation of β for each class separately. The following presents a quasi-uLSIF algorithm for solving the estimation of β for each class by modifying the standard uLSIF algorithm formulation. For class c, use the linear combination of Gaussians for modeling $\beta(x,y=c)$ like the standard uLSIF algorithm:

$$\beta(x,y=c) = \sum_{l=1}^{b} \alpha_l k(x,x_l)\,. \tag{4.18}$$

Similar to uLSIF method, the quadratic term $\hat{H}_{l,l'}^{(c)}$ has the same form as

$$\hat{H}_{l,l'}^{(c)} = \frac{1}{n_{tr}^{(c)}} \sum_{(x_i,y_i)\in S_{tr}\wedge y_i=c} k(x_i,x_l)k(x_i,x_{l'})\,. \tag{4.19}$$

However, the linear term needs to be modified in order to include the prediction loss on

Algorithm 6 Class-wise quasi-uLSIF

Input: $S_{tr}, X_{ts}, h()$
Output: w_{tr}
Steps:

```
 1: $p(y|x_{ts}) = h(x_{ts}),\ x_{ts} \in \mathcal{X}_{ts}$;
 2: for $c \in \mathcal{C}$ do
 3:     Calculate $\gamma(y = c)$ (Eq. 4.15);
 4:     Calculate $H^{(c)}$ (Eq. 4.19);
 5:     Calculate $q^{(c)}$ (Eq. 4.20);
 6:     Calculate $\tilde{\alpha}$ (Eq. 4.21);
 7:     $\hat{\alpha} = \max(0, \tilde{\alpha})$;
 8:     $\beta(x_{tr}, y = c) = K(x_{tr}^{(y=c)}, x_l)\hat{\alpha}$;
 9: end for
10: Return $w_{tr} = \beta \cdot \gamma$;
```

the training samples $\frac{c_1}{\lambda_1} \sum_{(x_i,y_i) \in S_{tr} \wedge y_i = c} (\beta(x_i, y_i)\gamma(y_i)L(x_i, y_i, h))$ and the soft decision on the test samples belonging to a class $p_{ts}(y = c|x'_j)$. Therefore, define $\hat{q}_l^{(c)}$ as

$$\begin{aligned}\hat{q}_l^{(c)} &= \frac{1}{\sum_{x'_j \in \mathcal{X}_{ts}} p_{ts}(y = c|x'_j)} \sum_{x'_j \in \mathcal{X}_{ts}} [p_{ts}(y = c|x'_j)k(x'_j, x_l)] \\ &\quad - \frac{c_1}{\lambda_1}[\sum_{(x_i,y_i) \in S_{tr} \wedge y_i = c} \gamma(y_i)L(x_i, y_i, h)] \,. \end{aligned} \tag{4.20}$$

Then, for class c, the coefficients of the model $\beta(x, y = c)$ can be analytically obtained as

$$\tilde{\alpha} = (\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_b)^T = \left(\hat{H}^{(c)} + \lambda I_b\right)^{-1} \hat{q}^{(c)} \tag{4.21}$$

Further, by enforcing the non-negativity constraint, we have the solution as $\hat{\alpha} = \max(0, \tilde{\alpha})$. Substituting $\hat{\alpha}$ into Eq. 4.18, the density ratios of the training samples in class c can be estimated with this quasi-uLSIF algorithm (see Algorithm 6). We demonstrate how the class-wise density-ratio estimation can be implemented based on the uLSIF metric. It is worthy to note that the same idea can be easily extended to other density-ratio estimation methods.

Algorithm 7 DDR Solution with TSVM (DDR-TSVM)

```
Input: $S_{tr}, X_{ts}$
Output: $h^*_{ts}, w^*_{tr}$
Steps:
 1: Initialize:
 2: t = 1;
 3: $w_{tr}(t) = 1$;
 4: $\epsilon$ = small number;
 5: while not-converged do
 6:     $h(t)$ = Cost-sensitive-TSVM($S_{tr}, w_{tr}(t), X_{ts}$) (call Algorithm 1);
 7:     $w_{tr}(t+1)$ = Class-wise-Quasi-uLSIF($S_{tr}, X_{ts}, h(t)$) (call Algorithm 2);
 8:     Converge-condition = $(\frac{1}{n_{tr}} \|w_{tr}(t+1) - w_{tr}(t)\| < \epsilon)$;
 9:     t = t + 1;
10: end while
11: Return $h^*_{ts} = h(t)$;
12: Return $w^*_{tr} = w_{tr}(t)$;
```

The Algorithm

With the BCU optimization procedure, we derive Algorithm 7, which alternatively updates the model and sample weights using the two blocks: the TSVM and the quasi class-wise distribution matching. The procedure is repeated until convergence is reached. The proof of convergence is shown in the next section.

The initial weights for the training samples are set to one. It means that the prediction performance is expected to be reasonable by applying the training domain model to the new test domain data directly. This assumption is common for successful domain adaptation [27].

Convergence Analysis

In order to show that Algorithm 7 is guaranteed to converge, let's consider the two steps of the BCU procedure. First, the cost-sensitive TSVM is guaranteed to converge. As shown in Theorem-2 of [68], the TSVM algorithm is proved to converge in a finite number of steps. The embedded weights on the labeled samples modify the linear term of the TSVM, but this does not change the optimization process of the TSVM, and accordingly, the cost-sensitive

TSVM will converge in a finite number of steps. Second, the class-wise quasi-uLSIF is a quadratic problem and can be analytically solved to obtain a global optimum.

Theorems 2.1 and 3.1 of [82] state that the BCU procedure on quadratic problems converges at least Q-linearly on its objective function. Since both the TSVM and quasi-uLSIF are quadratic problems, this means that the BCU procedure of Algorithm 7 is guaranteed to converge to a stationary point of the DDR objective function.

4.3.3 Alternating Between Supervised Learning and Class-wise Distribution Matching with Early Stopping

In this section, we present the second solution for the DDR problem based on the BCU procedure with an early stopping criterion. The development of this solution is motivated by two observations: 1) the simulated annealing used for semi-supervised learning has a high computation demand; 2) there are many learning algorithms which can not be naturally extended to handle the semi-supervised setting.

Revisiting the DDR problem of Eq. 4.11, if we use the third term as a stopping criterion for the BCU procedure, the algorithm will only optimize the first two terms which represent the cost-sensitive supervised learning and the class-wise distribution matching. This alleviates the computational cost of semi-supervised learning and allows for the use of traditional classification algorithms that were previously used for the same problem. We have shown experimentally that this algorithm still achieves very good classification accuracies.

In order to define the stopping criterion, we use mutual information to express the separation of classes in the test data. Mutual Information (MI) has been studied in the context of discriminative clustering [35] and semi-supervised learning [39]. Maximizing this criterion implicitly means that the current model's output, $p_{ts}(y|x'_j)$, has the least amount of confusing labeling and classification decisions located in the sparse regions [104]. Using the MI as the early stopping criterion is helpful because it is a smoother indicator than the hinge loss, and therefore helps to escape the local optimum caused by the limited number of samples.

The Mutual Information (MI) between the test samples $\mathcal{X}_{ts}$ and their estimated labels $\hat{\mathcal{Y}}_{ts}$ using the posterior outputs of model h is defined as

$$\mathrm{MI}\left(h, \mathcal{X}_{ts}\right) = -H(\boldsymbol{p}_0) + \frac{1}{n_{ts}} \sum_{x'_j \in \mathcal{X}_{ts}} H(\boldsymbol{p}_{ts}(y|x'_j)) \,, \tag{4.22}$$

Algorithm 8 DDR Solution with Early Stopping (DDR-ES)

Input: S_{tr}, X_{ts}
Output: h^*_{ts}, w^*_{tr}
Steps:

```
1:  Initialize:
2:  t = 1;
3:  $w_{tr}(t) = 1$;
4:  $\epsilon$ = small number;
5:  **while** not-converged **do**
6:      $h(t)$ = Cost-sensitive-Learner$(S_{tr}, w_{tr}(t))$;
7:      MI$(t)$ = Calculate-MI$(h(t), X_{ts})$ (Eq. 4.22);
8:      $w_{tr}(t+1)$ = Class-wise-Quasi-uLSIF$(S_{tr}, X_{ts}, h(t))$ (call Algorithm 2);
9:      Converge-condition = $(\frac{1}{n_{tr}} \|w_{tr}(t+1) - w_{tr}(t)\| < \epsilon)$;
10:     t = t + 1;
11: **end while**
12: t-chosen = $\arg\max_t$MI$(t)$;
13: Return $h^*_{ts} = h$(t-chosen);
14: Return $w^*_{tr} = w_{tr}$(t-chosen);
```

where $\boldsymbol{p}_{ts}(y|x'_j)$ is the posterior vector for a test sample x'_j using the current model's output, defined as

$$\boldsymbol{p}_{ts}(y|x'_j) = \left[p_{ts}(y=1|x'_j), \ldots, p_{ts}(y=c|x'_j)\right]^T \ , \tag{4.23}$$

and $\boldsymbol{p}_0$ is the class prior vector, defined as $\boldsymbol{p}_0 = \frac{1}{n_{ts}} \sum_{x'_j \in \mathcal{X}_{ts}} \boldsymbol{p}_{ts}(y|x'_j)$. The function $H(\cdot)$ is the information entropy defined over the probability vector as

$$H(\boldsymbol{p}_{ts}) = -\sum_{i=1}^{c} \boldsymbol{p}_{ts}(i) \ln(\boldsymbol{p}_{ts}(i)) \ . \tag{4.24}$$

Algorithm 8 presents the BCU procedure between the two blocks: the cost-sensitive classification and the class-wise distribution matching. The early stopping criterion is based on the objective of the third term, which uses the MI to express the margin on the unlabeled test data.

Convergence Analysis

The convergence of Algorithm 8 can be proved by showing that the optimization problem is multiconvex [125]. When the third term of Eq. 4.11 is taken out as an early stopping criterion, we have the following optimization problem to solve:

$$\{h_{ts}^*, w^*\} = \text{argmin}_{h,\beta,\gamma,w}\{\hat{R}_{tr}(h, \mathcal{S}_{tr}, w(x,y)L(x,y,h)) + \lambda_1 \text{Dcw}\,(\beta, \mathcal{S}_{tr}, \mathcal{X}_{ts}, h)\}\,. \quad (4.25)$$

It can be shown that the supervised learning algorithm has a convex objective function and the use of quasi-uLSIF for distribution matching is also convex. Based on Theorem 4.1 of [117], the BCU procedure on multiconvex optimization problems is guaranteed to converge to a stationary point.

It is worthy to mention that even though the sub-functions of the two blocks are convex and the global optimum is obtainable at each block, the overall solution of the BCU algorithm is typically a local optimum, because there is a common term for the two blocks in the DDR problem.

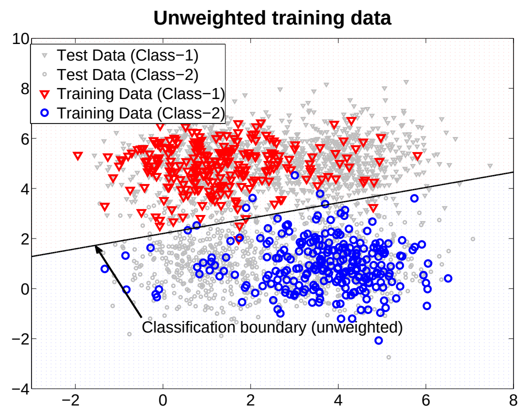
4.4 Experiments

This section presents the results of three experiments to evaluate the performance of the proposed DDR approach. The first experiment is conducted on a synthetic 2-class 4-cluster dataset. The second experiment evaluates the sampling bias scenarios on different benchmark datasets. The third experiment is a more challenging cross-dataset learning task. In addition, empirical studies are conducted on the parameter sensitivity and the convergence behavior of the proposed methods.

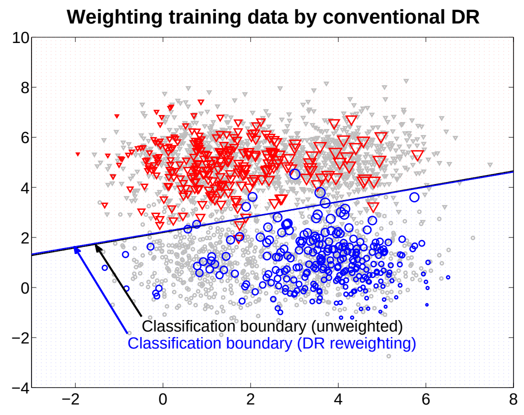
4.4.1 Illustrative Example of Synthetic Data

The first experiment is designed with samples generated from 2-dimensional Gaussian mixture models, in which both the class priors and likelihoods exhibit changes. The 2-class 4-cluster distributions of the training and test data are given in Table 4.1.

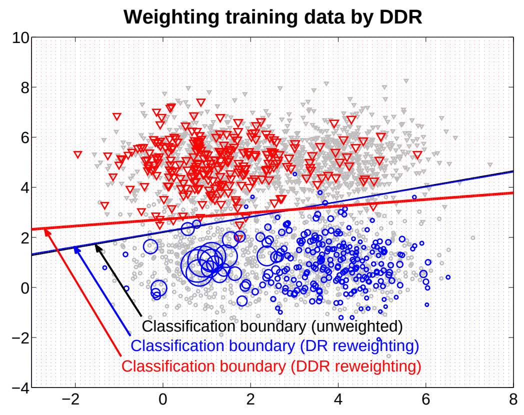
Fig. 4.1 illustrates the difference of importance estimation results between the unweighted approach, the conventional DR, and the proposed DDR methods. The figure also shows the corresponding classification boundaries. We can observe that the conventional DR method assigns higher importance weights to the misclassified blue points because they



(a) Training and test data



(b) Conventional Density-Ratio (DR)



(c) Discriminative Density-Ratio (DDR)

Figure 4.1: Weighted training data and classification boundaries for the synthetic data. The plotted sizes of training samples are proportional to the logarithm of estimated density ratios.

Table 4.1: The distributions of the training and test data of a synthetic 2-class 4-cluster problem.

		Prior	**Likelihood**
p_{tr}	class-1	0.5	$0.9 \times \mathcal{N}\left(\begin{bmatrix}1\\5\end{bmatrix}, I\right) + 0.1 \times \mathcal{N}\left(\begin{bmatrix}4\\5\end{bmatrix}, I\right)$
	class-2	0.5	$0.1 \times \mathcal{N}\left(\begin{bmatrix}1\\1\end{bmatrix}, I\right) + 0.9 \times \mathcal{N}\left(\begin{bmatrix}4\\1\end{bmatrix}, I\right)$
p_{ts}	class-1	0.6	$0.5 \times \mathcal{N}\left(\begin{bmatrix}1\\5\end{bmatrix}, I\right) + 0.5 \times \mathcal{N}\left(\begin{bmatrix}4\\5\end{bmatrix}, I\right)$
	class-2	0.4	$0.5 \times \mathcal{N}\left(\begin{bmatrix}1\\5\end{bmatrix}, I\right) + 0.5 \times \mathcal{N}\left(\begin{bmatrix}4\\5\end{bmatrix}, I\right)$

lie in a dense region of test points (Fig. 4.1-b), while our proposed DDR method assigns small importance weights to these points and, accordingly, learns a much better decision boundary (Fig. 4.1-c). Fig. 4.1-a clearly shows that classification using the unweighted approach is biased to training samples and that leads to a suboptimal model on the test data.

4.4.2 Experiments on Sampling Bias Benchmark Data

This experiment evaluates the proposed DDR method on a set of benchmark datasets. The datasets 'breast cancer', 'diabetes', 'ionosphere', 'wdbc', 'image segmentation', 'mushroom' are from the UCI Machine Learning Repository[1].

The sampling bias classification tasks are formulated by using a deliberately biased selection procedure to split the training and test data, following the setup of [28]. In all experiments, before any further processing, all the data are normalized to the range $[-1, 1]^d$. Then, half of data are uniformly sampled to form the testing section. The rest of data are sub-sampled to form the biased training set with the probability of $P(s = 1|x) = e^v/(1 + e^v)$, where $s = 1$ means the sample is included in the training set, $v = 4(\boldsymbol{\omega^t}(x - \overline{x}))/\sigma_{\boldsymbol{\omega^t}(x-\overline{x})}$, and $\boldsymbol{\omega} \in R^d$ is a projection vector randomly chosen from $[-1, 1]^d$. For each run, ten values of $\boldsymbol{\omega}$ are randomly generated, and we select the vector $\boldsymbol{\omega}$ which maximizes the difference between the unweighted method and the weighted method with ideal sampling weights.

[1]These datasets are downloaded from http://archive.ics.uci.edu/ml/datasets.html

The classifiers being used include importance-weighted Support Vector Machine (iwSVM) [21] and importance-weighted Least-Squares Probabilistic Classifier (iwLSPC) [54]. For iwSVM, the RBF kernel is used, and the default parameter value as adopted. For iwLSPC, we took the same data splits generated in the iwSVM setting. The number of kernel basis functions was set to 100 by random sampling from the test data. The other parameters (the kernel width σ and regularization parameter λ) were chosen by 5-fold importance-weighted Cross-Validation (iwCV) [109]. The hyper parameters were set as $\lambda_1 = 1$ and $\lambda_2 = 0.1$.

We evaluated the performance of our DDR method by comparing with the conventional density-ratio estimation methods using the same setup. The classification results using the model learned from the unweighted training data are included as the baseline. Because of the deliberately biased sampling selection procedure, we know that the probability of each sample being included into the training section was $P\,(s = 1|x)$. Therefore, the ideal sample importance is the reciprocal of the probability of being selected, i.e., $\mathrm{imp}(x) = {}^{1}\!/_{P\,(s\,=\,1|x)}$. The results of using the oracle importance weights are reported in the 'Oracle-weight' column in Table 4.3 and 4.4. In the KMM density-ratio method, the kernel width was set as the median of sample pair distances. The normalization constraint was set as suggested by the original authors [53] as $\epsilon = {}^{(\sqrt{n_{tr}}\,-\,1)}\!/_{\sqrt{n_{tr}}}$. The parameters of KLIEP and uLSIF were chosen by their equipped likelihood cross-validation.

All experiments were repeated 30 times with different training-test data splits. The significance of improvement in classification accuracy was tested using the signed rank test and the Friedman test [33]. The results are summarized in Table 4.2, Table 4.3 and Table 4.4. In Table 4.3 and 4.4, the results of DR method are the best among the methods of KMM, KLIEP and uLSIF. It can be observed that the proposed DDR approach outperforms the unweighted method and the conventional density-ratio estimator in almost all cases. There are four cases where the accuracies are improved by more than 10%. Significantly, the DDR approach sometimes achieved higher accuracy than the corresponding methods even when the corresponding methods used the ideal weights. This is because the weights are ideal in the sense of reflecting the density-ratio over x only. Comparing the two algorithms for DDR, the early stopping method seems to perform slightly better than the semi-supervised method. We believe that this is due to the characteristics of the sampling bias problem, which usually does not dramatically change the classification models. This means the initial model learned from the training data directly is located at a region near the global optimum in the solution space. Thus, the early stopping algorithm will give more favorable performance.

Table 4.2: Classification accuracies of sampling bias data with density-ratio methods. Comparing with the baseline method that uses unweighted samples, those significantly better accuracies through the signed rank test at 5% significance level are presented in italics.

Dataset	**Classifier**	**Unweighted**	**KMM**	**KLIEP**	**uLSIF**
breastCancer	iwSVM	0.8674±.1176	*0.9271±.0587*	*0.9283±.0602*	*0.9300±.0603*
	iwLSPC	0.8166±.1471	*0.8744±.1367*	*0.8389±.1465*	*0.8880±.1270*
diabetes	iwSVM	0.6683±.0343	*0.7095±.0352*	0.6714±.0366	0.6684±.0345
	iwLSPC	0.7100±.0399	0.6775±.0467	0.6986±.0383	0.7041±.0360
ionosphere	iwSVM	0.8269±.0305	0.8159±.0485	0.8189±.0399	0.7788±.0726
	iwLSPC	0.7352±.0478	0.7362±.0743	0.7288±.0536	0.7331±.0529
wdbc	iwSVM	0.7580±.0468	*0.8878±.0504*	0.7558±.0492	0.7522±.0493
	iwLSPC	0.9135±.0301	0.9235±.0513	0.9115±.0328	0.9090±.0318
imageSegment	iwSVM	0.8547±.0111	*0.8965±.0230*	*0.8633±.0198*	*0.8788±.0328*
	iwLSPC	0.9151±.0212	0.9175±.0145	0.9120±.0257	0.9100±.0244
mushroom	iwSVM	0.9697±.0074	*0.9751±.0178*	*0.9776±.0143*	*0.9800±.0161*
	iwLSPC	0.8360±.1223	0.7770±.1416	0.8479±.1228	0.7951±.1189

Table 4.3: Classification accuracies of sampling bias data with the DDR and iwSVM. Comparing with the baseline method that uses unweighted samples, those significantly better accuracies through the signed rank test at 5% significance level are presented in italics. The best results (the Friedman test at a 95% confidence interval) in each group are highlighted in bold. The results of DR method are the best among the methods of KMM, KLIEP and uLSIF. The 'Oracle-weight' is a reference and is not involved to comparison.

Dataset	**Unweighted**	**DR**	**DDR-TSVM**	**DDR-ES**	**Oracle-weight**
breastCancer	0.8674±.1176	*0.9300±.0603*	***0.9634±.0108***	*0.9586±.0144*	0.9357±.0496
diabetes	0.6683±.0343	*0.7095±.0352*	*0.7252±.0295*	***0.7345±.0314***	0.7016±.0433
ionosphere	0.8269±.0305	0.8189±.0399	*0.8485±.0685*	***0.8589±.0455***	0.8258±.0413
wdbc	0.7580±.0468	*0.8878±.0504*	*0.9389±.0083*	***0.9540±.0145***	0.8353±.0675
imageSegment	0.8547±.0111	*0.8965±.0230*	*0.9053±.0500*	***0.9155±.0225***	0.9183±.0310
mushroom	0.9697±.0074	*0.9800±.0161*	*0.9767±.0098*	***0.9852±.0159***	0.9845±.0166

Table 4.4: Classification accuracies of sampling bias data with the DDR and iwLSPC. The semi-supervised setting is not applicable for the iwLSPC classifier.

Dataset	**Unweighted**	**DR**	**DDR-ES**	**Oracle-weight**
breastCancer	0.8166±.1471	*0.8880±.1270*	***0.9295±.0881***	0.9250±.0898
diabetes	0.7100±.0399	0.7041±.0360	**0.7181±.0279**	0.6982±.0353
ionosphere	0.7352±.0478	0.7362±.0743	***0.7477±.0348***	0.7458±.0674
wdbc	0.9135±.0301	**0.9235±.0513**	0.9219±.0283	0.9199±.0485
imageSegment	0.9151±.0212	0.9175±.0145	**0.9184±.0255**	0.9170±.0277
mushroom	0.8360±.1223	0.8479±.1228	***0.9341±.0111***	0.8504±.1288

4.4.3 Experiments on Cross-dataset Digits Recognition

Training a model with samples from one dataset and adapting the model to another dataset which has been collected under different conditions is usually seen as a challenging problem. We evaluated our DDR approach for the cross-dataset classification task using the two handwritten digits recognition datasets: USPS and MNIST. The USPS dataset contains 9,298 handwritten digit images with the size of 16×16. The MNIST dataset has a total of 70,000 handwritten digit images (the first 20,000 samples are used in our experiment). The size of each image is 28×28.

Because the two datasets have different image sizes and intensity levels, a preprocessing step was applied first: 1) resize the image size of MNIST from 28×28 into the same size as USPS, 16×16, and 2) normalize the feature (intensity of pixel) into the range of $[-1, 1]$. Then, we conducted two experiments: one using the USPS data for training and the MNIST data for testing, and the other using the MNIST data for training and the USPS data for testing. The classification method used was a SVM with linear kernels. The parameter c in the SVM is a trade-off between model generalization and training error, and its value was chosen using 5-fold importance-weighted Cross-Validation (iwCV). The hyper parameters were once again set as $\lambda_1 = 1$ and $\lambda_2 = 0.1$.

Because the uLSIF algorithm has been shown to outperform other density-ratio methods, such as KMM and KLIEP, we included the uLSIF in our comparison in this series of experiments. Table 4.5 and 4.6 present the average and standard deviations of the classification accuracies of 30 runs for the cross-dataset tasks. Each run was based on bootstrapping the data by randomly selecting 90% samples from the designated training dataset and 90% samples from the designated test dataset.

The reported results show that the DDR method significantly improves recognition

Table 4.5: Cross-dataset tasks: classification accuracies of training on the USPS dataset and test on the MNIST dataset. For each test case, comparing with the baseline method that uses unweighted samples, those significantly better accuracies through the signed rank test at 5% significance level are presented in italics. The best results (the Friedman test at a 95% confidence interval) in each group are highlighted in bold, and the second best results are underlined.

	USPS to MNIST			
Test Case	**Unweighted**	**DR(uLSIF)**	**DDR-TSVM**	**DDR-ES**
0 vs 1	0.8717±.0614	0.8879±.0631	***0.9968±.0017***	*0.9836±.0054*
1 vs 2	0.5961±.0281	0.5925±.0287	0.6001±.0352	**0.6253±.0627**
2 vs 3	0.7314±.0854	0.7575±.0948	***0.8830±.0507***	*0.8635±.0281*
3 vs 4	0.7898±.0483	*0.8284±.0288*	***0.9398±.0139***	*0.8435±.0270*
4 vs 5	0.6254±.0574	*0.6937±.0472*	***0.8104±.0505***	*0.7456±.0377*
5 vs 6	0.5552±.0592	0.5074±.0313	*0.5775±.0560*	***0.5978±.0692***
6 vs 7	0.6219±.0857	0.6287±.0843	***0.9955±.0012***	*0.6809±.0504*
7 vs 8	0.6153±.0594	0.6139±.0545	***0.7477±.1388***	0.6591±.0677
8 vs 9	0.6965±.0743	*0.7411±.0888*	***0.8931±.0468***	*0.8335±.0309*
9 vs 0	0.9195±.0243	0.9185±.0244	0.9175±.0040	0.9175±.0244

accuracies. Compared to the conventional DR approach, for the scenario 'USPS to MNIST' nine out of ten test cases achieve an improvement, and seven cases record improvement in accuracy by increments of more than 10%. For the scenario 'MNIST to USPS', all ten test cases gain an improvement in accuracy of 4% to 26%. Comparing the two algorithms for DDR, the semi-supervised learning setting outperforms the early stopping algorithm in most cases. For the cross-dataset tasks, the distributions between the two data collections exhibit great differences. It is even expected that there exist regions in the test data with no representative training data. This would lead to the initial model falling into an unsatisfactory region in the solution space, and the early stopping solver becoming easily trapped at a local solution far from the global optimum. In these scenarios, the algorithm that uses the semi-supervised learning would be able to overcome this limitation and obtain a better local-optimal solution.

Table 4.6: Cross-dataset tasks: classification accuracies of training on the MNIST dataset and test on the USPS dataset.

	MNIST to USPS			
Test Case	**Unweighted**	**DR(uLSIF)**	**DDR-TSVM**	**DDR-ES**
0 vs 1	0.9461±.0085	0.9499±.0089	***0.9981±.0009***	*0.9848±.0036*
1 vs 2	0.9048±.0266	*0.9161±.0279*	***0.9758±.0059***	*0.9533±.0205*
2 vs 3	0.6383±.0367	0.6561±.0308	***0.8051±.0107***	0.6597±.0284
3 vs 4	0.7915±.0313	0.7767±.0318	***0.8647±.0419***	*0.8266±.0217*
4 vs 5	0.8374±.0338	*0.8480±.0360*	***0.8893±.0398***	*0.8836±.0091*
5 vs 6	0.5810±.0331	0.5676±.0317	*0.7288±.0244*	***0.7382±.0359***
6 vs 7	0.5114±.0299	0.5102±.0274	*0.6103±.1528*	***0.6718±.0554***
7 vs 8	0.6199±.0477	0.6097±.0412	*0.7956±.0294*	***0.8298±.0352***
8 vs 9	0.8249±.0622	0.7934±.0628	***0.8605±.0327***	*0.8480±.0505*
9 vs 0	0.6419±.1105	0.6735±.1598	***0.9367±.0713***	*0.9269±.0349*

4.4.4 Study on the Hyper Parameters

In this section, the results of an empirical study on the effect of the two hyper parameters λ_1 and λ_2 in the DDR model are presented. The experiments described in this section were conducted on the DDR-TSVM algorithm only because the DDR with early stopping is a simplification of DDR-TSVM. We examined four cases in the 'MNIST to USPS' scenario: '0 vs 1', '1 vs 2', '2 vs 3', and '5 vs 6'.

First, we explored the impact of parameter λ_1 while fixing parameter λ_2 at $0.1 \times c_1$. The parameter c_1 is the cost factor in the SVM formulation, which was decided by 5-fold cross-validation. The relative accuracy was calculated by setting the respective baseline method as 100% for visualizing the margins of performance improvement. Fig. 4.2 plots the relative accuracies for varying values of λ_1 over the range $\lambda_1 = [0, 0.1 : 0.1 : 1, 2 : 1 : 10]$. Next, we studied the impact of λ_2 while fixing parameter λ_1 at 1. The performance was calculated by varying values of λ_2 over the range $\lambda_2 = [0, 0.01 : 0.01 : 0.1, 0.2 : 0.1 : 1] \times c_1$.

As shown in Fig. 4.2, the DDR algorithm performs with high stability for a wide range of parameter selections, except for the extreme setups that $\lambda_1 = 0$ or $\lambda_2 = 0$. When the parameter λ_1 is set to 0, the DDR model in fact is reduced to the unweighted semi-supervised learning that assumes the training data and the test data are from the same distributions. When the parameter λ_2 is set to 0, the information on the test data margin

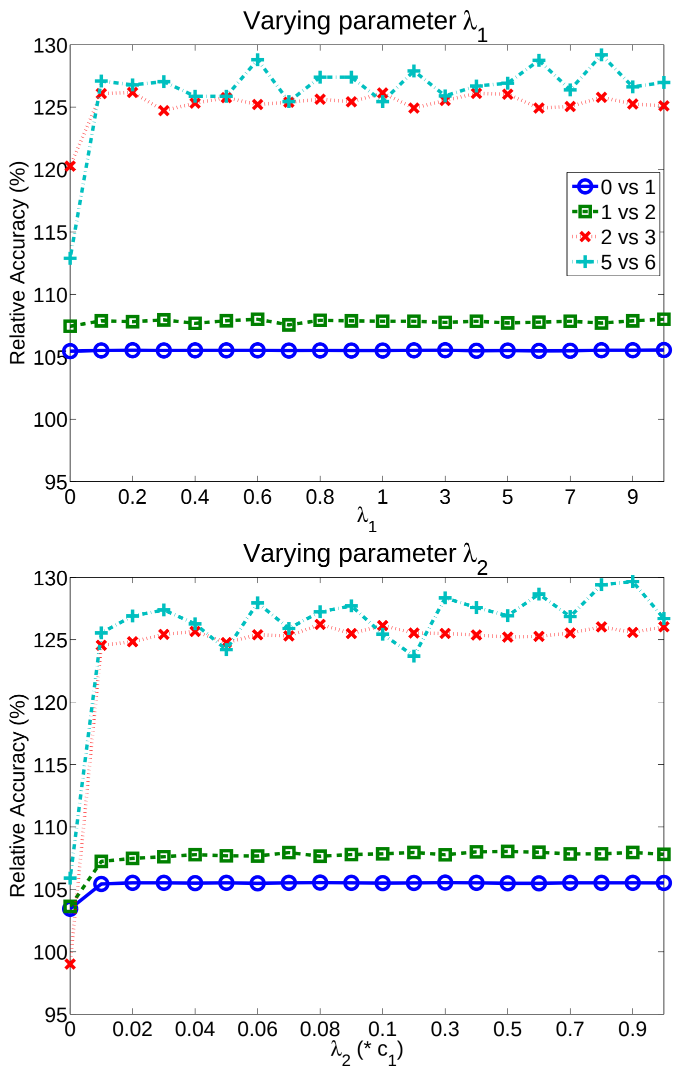


Figure 4.2: Relative accuracies of DDR for varying the setting of parameter λ_1 and λ_2 on the two easy tasks (‘0 vs 1’, ‘1 vs 2’) and two difficult tasks (‘2 vs 3’, ‘5 vs 6’).

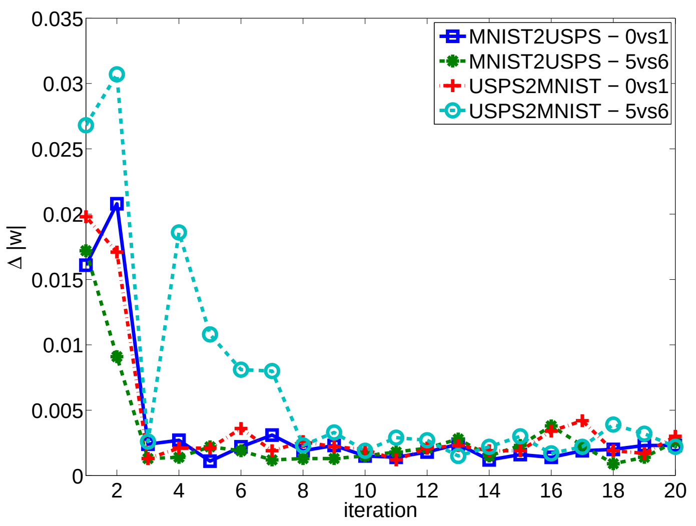


Figure 4.3: The change of Δw with respect to the number of iterations shows the convergence behavior of the DDR algorithm.

is ignored, and often the solution deteriorates into a worse local optima.

4.4.5 Convergence Behavior

Because the proposed DDR-based algorithms are iterative, their convergence behavior is an important concern. Here we check the algorithm's convergence speed empirically on the cross-dataset recognition tasks.

Fig. 4.3 shows the change of sample weights $\Delta w = \frac{1}{n_{tr}} \|w(t+1) - w(t)\|$ with respect to the number of iterations for two easy tasks ('0 vs 1' of 'MNIST2USPS' and 'USPS2MNIST') and two difficult tasks ('5 vs 6' of 'MNIST2USPS' and 'USPS2MNIST'). As shown in the figure, the proposed method converges in just a few iterations. The DDR algorithm is a special case of using the block coordinate update method, in which each block of algorithms is solved with a local optimal solution. This approach is in opposition with many general BCU algorithms, which usually have a slow convergence rate because the updates of each block are based on gradient searching with a small step width.

Chapter 5

Locally-Adaptive Density Ratio for Novelty Detection

In non-stationary environments, along with the changing of existing concepts which are handled by domain adaptation techniques, we also face the occurrence of new concepts. The detection of novelties is another crucial problem in non-stationary data mining. In dynamic data there naturally exist two types of novelties: emerging and evolving. Emerging novelties are represented by concepts which are very different from the previously seen ones, while evolving novelties are characterized by relatively new aspects of existing concepts. In real situations, these two types of novelties are not easily distinguishable, and sometimes a truly novel concept does not fit perfectly under one of these categories. In this chapter, a locally-adaptive kernel density-ratio method is proposed to capture the two characteristics in one formula.

The chapter is organized as follows. Section 5.1 provides an overview and a motivation example. Section 5.2 describes the details of the proposed locally-adaptive density ratio method. Section 5.3 relates the proposed method to some previous works. Section 5.4 presents experimental results. Section 5.5 describes an application of the proposed method to social media analysis. Lastly, computation complexity is discussed in Section 5.6.

5.1 Overview

Novelty detection is the task of identifying *abnormal* instances in new data that differ in one or more aspects from a previous collection of *normal* data [94]. The detection of novel instances is a crucial task in data mining and machine learning and has a variety of applications, such as the early detection of defects and failures in industrial systems, the early discovery of frauds, disease outbreaks, natural crises, and learning of emerging topics in news and online discussions. The novelty detection problem has also been addressed in the literature under other names such as outlier detection [1,12] and anomaly detection [20, 26].

A common approach to novelty detection is to first build a model for normality, and then recognize any deviation from that model as novelty. There is no agreed-upon definition of what is normal and what is novel, and different existing methods for novelty detection capture different aspects of novelty. For instance, some methods, such as the Local Outlier Factor (LOF) [13] and One-class SVM (OSVM) [100], focus on detecting novel instances that emerge in new data which are completely different from previously-seen instances. Other methods, such as relative [106] and density-ratio based [59] novelty detection, focus on detecting instances which are not new by themselves, but their intensities are considerably different from previously-seen data. We refer to these two types of novelties as *emerging* and *evolving*, respectively. These two terms are borrowed from the topic detection community [98], where the first term refers to absolutely new topics that have never been seen before, and the second refers to topics that have evolved smoothly over time.

While previous approaches for novelty detection exhibit appealing successes in specific applications, methods that focus on detecting emerging novelties are quite limited in identifying evolving novelties, and vice versa. In real situations, many of the truly novel concepts do not perfectly fit under one of these two categories, and normally have both emerging and evolving aspects. For instance, an online discussion about a new disease outbreak involves reference to comparable outbreaks in the past that have had similar effects.

To address these limitations, a new approach is presented for novelty detection that combines the strength of both categories of methods and recognizes both emerging and evolving novelties. The basic idea behind the proposed approach is to define a novelty factor that combines the emerging and evolving aspects of a new instance. The factor measures how novel an instance is relative to its neighborhood structure, as well as how novel it is relative to the reference data. Accordingly, we derive a locally-adaptive approach for density ratio estimation and use it to measure the novelty of an instance. The use of density-ratio provides a measure of how evolving the novelty is, while the local neighborhood captures how emerging it is.

5.1.1 Existing Approaches

Here we briefly review the state-of-the-art of novelty detection approaches and analyze their strengths and shortcomings.

Due to the lack of abnormal samples and the diversity of abnormal modes, it is more practical to construct novelty detection methods based on modeling the normal class of data only, and identifying any deviation from that normal class as novel [49, 94]. In this research, we are mainly concerned with the novelty detection problem under the assumption that *normal* data have been previously identified, and numerous examples are provided as a reference set. This is an assumption that has been adopted in many well-recognized works on the novelty detection task [13, 59, 72, 100, 106].

In the aforementioned setting, the model of normality $M(x)$ is learned from the given normal examples

$$\mathcal{X}_{\text{rf}} = \{x_i | i = 1, ..., n_{\text{rf}}\} ,$$

as the reference collection. Then, in the test stage, previously-unseen instances

$$\mathcal{X}_{\text{ts}} = \{x_j | j = 1, \ldots, n_{\text{ts}}\} ,$$

are tested against the model M, and the corresponding novelty scores are calculated. The novelty score for a test sample x, i.e. $m = M(x)$, is compared to a decision threshold τ. If $m \leq \tau$, this instance x is classified as normal. Otherwise, the instance is classified as novel, or abnormal.

A variety of approaches for novelty detection have been proposed with different inspirations. Existing work can generally be classified into four categories, as explained in the following sections.

Probabilistic-based approach. Probabilistic-based novelty detection methods estimate the Probability Density Function (PDF) of the normal data, and assume that the low-density areas have high probability of being novel [72]. A variety of PDF estimation techniques can be employed to learn a model of the normal data. This includes Gaussian Mixture Model (GMM) [103] and Kernel Density Estimation (KDE) methods [116]. The GMM method is parametric and assumes the data are generated from a weighted mixture of Gaussian distributions, while KDE is a non-parametric approach which is closely related to histograms, but enhanced with the properties of smoothness and continuity due to the use of kernel functions. PDF-based novelty detection tends to incorrectly identify the majority of data instances in sparse regions as novelties.

Domain boundary-based approach. The boundary-based approaches model the boundary of normal data and assume that samples located outside of this boundary are novel. Examples for these methods include the One-class SVM (OSVM) [100] and the conceptually-similar Support Vector Data Description (SVDD) [115].

The OSVM method is an extension of the SVM algorithm that separates a data space into inliers and outliers by modeling an inlier boundary that contains a fraction of the training data. Formally, an OSVM formulates a quadratic optimization problem as

$$\begin{aligned} & \min_{\alpha} \frac{1}{2}\alpha^T K \alpha \\ \text{s.t.} \quad & 0 \le \alpha_i \le \frac{1}{vn} \text{ for } i = 1, \dots, n \\ \text{and} \quad & \sum_{i=1}^{n} \alpha_i = 1 \, , \end{aligned}$$

where $K_{ij} = \kappa(x_i, x_j)$ is the kernel matrix over the reference data, and v is the maximum fraction of outliers.

This boundary-based approach avoids distribution estimation and is typically insensitive to sampling noise. However, the performance of the OSVM and SVDD methods depend heavily on the choice of kernel and its parameters. Furthermore, the output novelty scores ignore the relative densities in different regions of the inliers. A trained OSVM is based on the boundary of the majority of the reference data and therefore can not be used to identify evolving novelties which overlap with the reference data.

Neighborhood-based approach. Neighborhood-based methods analyze the k-nearest neighbors of data instances, and identify novel instances as those relatively far from their neighbors. For example, the Local Outlier Factor (LOF) [13] is a well-known method for detecting outliers based on this local-neighborhood analysis. LOF calculates an outlier score for a test sample x as

$$\mathrm{LOF}_k(x) = \frac{1}{k} \sum_{i=1}^{k} \frac{\mathrm{lrd}_k\left(\mathrm{Nearest}_i\left(x\right)\right)}{\mathrm{lrd}_k(x)} \, ,$$

where $\mathrm{Nearest}_i(x)$ represents the i-th nearest neighbor of x, and $\mathrm{lrd}_k(x)$ is the local reachability density of x, defined as the inverse of the average distance from k nearest neighbors of x. If x lies far away from a much denser region, $\mathrm{lrd}_k(x)$ tends to be much smaller than $\mathrm{lrd}_k\left(\mathrm{Nearest}_i\left(x\right)\right)$. Therefore, it will result in a large value for $\mathrm{LOF}_k(x)$.

In this formulation, LOF captures the local neighborhood of data instances while estimating the novelty level, and has shown effective performance in some applications [78]. Although LOF and similar approaches are very effective in detecting emerging novelties, they are very limited in identifying evolving novelties that overlap with reference data, due to ignoring the relationship among these test instances.

Density ratio-based approach. Density-ratio estimation is a common approach that has been previously employed in a variety of data mining tasks [87]. The use of density-ratio estimation for novelty detection has been recently explored [59,106]. The idea behind this approach is to compare the densities of the test and reference data, and then identify the novel instances based on the ratio between these two densities. The estimation of a density-ratio can be done directly without estimating the density functions of test and reference data.

Examples of this approach include the work of Smola et al. [106], who proposed the concept of relative novelty and modified the One-class SVM formulation to incorporate the reference densities as density ratios. This work was followed by the work of Hido et al. [59], who defined an inlier score by using density ratios between normal data and test data. At a test instance x, if the density of normal data is low and that of test data is high, the instance is considered novelty and its inlier score will be small.

Since this approach depends mainly on the ratio of test and reference data densities, it is very effective in detecting relative (or evolving) novelties. It is, however, limited in identifying emerging novelties that are very dissimilar to the reference data.

5.1.2 A Motivation Example

In order to illustrate the limitations of existing novelty detection methods and motivate the need for developing a new approach, we start by providing a synthetic example of 2-D points that has emerging and evolving novelties.

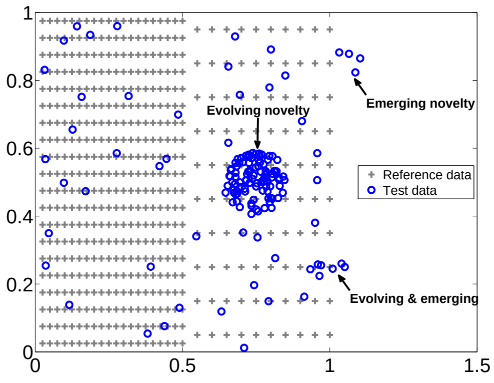
As shown in Fig. 5.1a, the reference data (gray crosses) contains normal data only and consists of three uniform regions: a dense region, a sparse region and an empty region. The test data to be examined for novelty (blue circles) contains normal as well as novel instances. The novel instances appear in three forms: (1) emerging novel instances that are far away from the distribution of the reference data in the empty region, (2) evolving novel instances that overlap with old instances but appear in a group of higher density, and (3) the clustered instances at the bottom-right corner that include both emerging and evolving novelties.

We examined different state-of-the-art approaches, as well as our proposed method, using this data. The probabilistic-based approach (Fig. 5.1b) identified all instances lying in the sparse region as novel, which usually causes a high false alarm rate. The domain boundary approach (One-class SVM) (Fig. 5.1c) assigned instances as novel if they are out of the boundary of a learned normality model, and cannot distinguish relatively-novel instances that reside within the boundary of normality. The neighborhood-based approach (local outlier factor) compares the neighborhood structure of each test instance with the reference instances separately, while ignoring the relationship among these test instances. As shown in Fig. 5.1d, the occurrence of a relatively dense region in the test data (evolving novelties) is not considered a novelty in this case. The density-ratio approach captured the relative density difference between the reference and the test collection (evolving novelties), while ignoring the not-high volume of emerging novelties at the boundary of the normal data. As shown in Fig. 5.1e, a few points located at the upper-right and bottom-right corners are not identified. The proposed method, on the other hand, identified both evolving and locally emerging novelties, as shown in Fig. 5.1f.

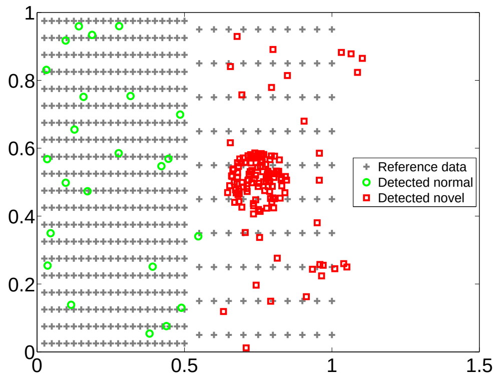
5.2 Locally-Adaptive Density Ratio Method

In the literature, existing techniques for novelty detection focus on either (1) detecting novel instances that are completely different from existing instances in the reference data, or (2) identifying a group of novel instances that are not completely new in themselves, but rather appear with a density which is different from that of similar instances in the reference data. As highlighted in the introduction, we refer to these two types of novelties as emerging and evolving, respectively. The objective of this work is to identify the limitations of each of these methods and introduce a method that is capable of identifying both emerging and evolving novelties.

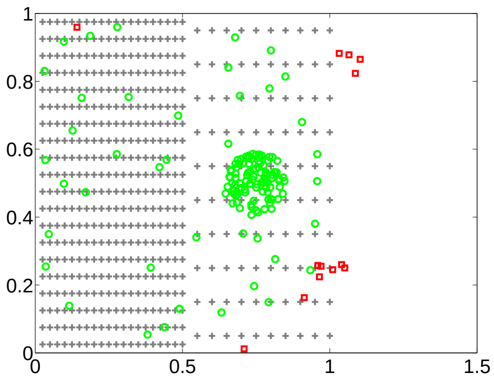
The basic idea behind the proposed method is to start with the effective kernel mean matching method for density-ratio estimation, which has been very successful in identifying evolving novelties (Fig. 5.1e), and to augment this method to capture emerging novelties. By inspecting a variety of problem instances, we observed that the density ratio as a novelty measure is very unreliable when encountering a few novelties in completely new areas of the feature space. In this case, the density-ratio estimation depends mainly on the absolute dissimilarity between the potentially novel instances and other reference instances, without considering information about how these reference instances are similar to each other. We propose to alleviate this problem by adaptively estimating the density ratio based on the neighborhood of both the new instance as well as the reference instances.



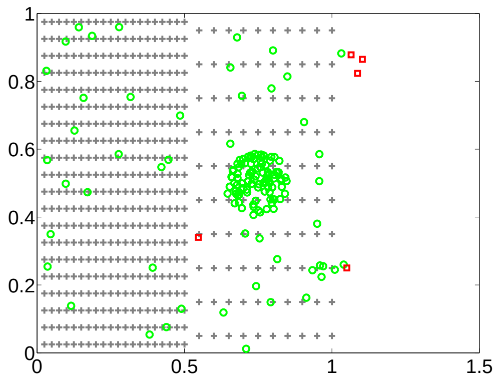
(a) Reference and test data



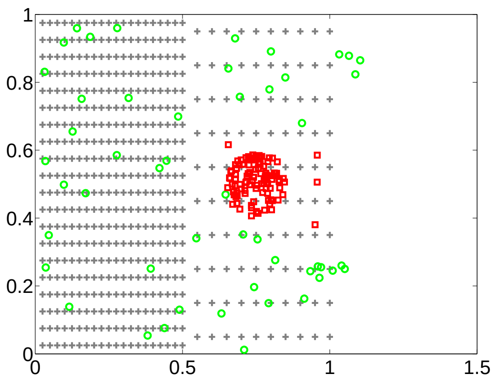
(b) PDF-based novelty detection



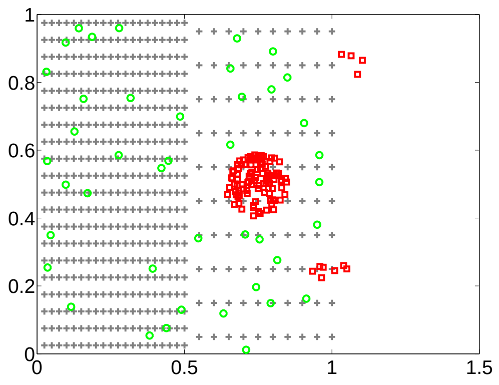
(c) One-class SVM



(d) Local Outlier Factor (LOF)



(e) Density ratio-based approach



(f) Locally-adaptive density ratio

Figure 5.1: An illustrative example of novelty detection methods.

In the rest of this section, we first formalize the aforementioned intuition by analyzing the kernel mean matching algorithm for density ratio estimation and identify the different components of the novelty score that quantify how emerging/evolving the novelty is. Then, we analyze how the component responsible for detecting emerging novelties is inaccurate. After that, we propose the use of a locally-adaptive kernel that results in an accurate estimation of emerging novelty scores.

5.2.1 Differentiate Emerging and Evolving Novelties

The kernel mean matching method for novelty detection proceeds as follows. Let $r(x)$ be a normality score for a data point x, defined as the density ratio between the reference and test data:

$$r(x) = \frac{p_{\mathrm{rf}}(x)}{p_{\mathrm{ts}}(x)} \,, \tag{5.1}$$

where $p_{\mathrm{rf}}(x)$ and $p_{\mathrm{ts}}(x)$ are the PDFs of the reference and test data, respectively. For a data point x, if the test density is relatively higher than the reference density, then the normality score will be very small, and this point is more likely to represent a novelty.

To derive a formula for $r(x)$, we use the theorem of Kernel Mean Matching (KMM) [62] and minimize the Maximum Mean Discrepancy (MMD) between the weighted distribution $r(x) * p_{\mathrm{ts}}(x)$ and the reference distribution $p_{\mathrm{rf}}(x)$ in a Reproducing Kernel Hilbert Space (RKHS) $\phi(x) : x \to \mathcal{F}$,

$$\mathrm{MMD}^2\left(\mathcal{F}, \left(r(x), p_{\mathrm{ts}}(x)\right), p_{\mathrm{rf}}(x)\right) = \left\| E_{x\sim p_{\mathrm{ts}}(x)}\left[r(x)\cdot\phi(x)\right] - E_{x\sim p_{\mathrm{rf}}(x)}\left[\phi(x)\right]\right\|^2 \,. \tag{5.2}$$

Using the empirical means of $\mathcal{X}_{\mathrm{rf}}$ and $\mathcal{X}_{\mathrm{ts}}$ to replace the expectations, and defining a vector $\boldsymbol{r} = [r(x_1), \ldots, r(x_{n_{\mathrm{ts}}})]^T$, we can obtain the following quadratic optimization problem:

$$
\begin{aligned}
\hat{\boldsymbol{r}} &= \operatorname{argmin}_{\boldsymbol{r}} \| \frac{1}{n_{\text{ts}}} \sum_{j=1}^{n_{\text{ts}}} r(x_j)\phi(x_j) - \frac{1}{n_{\text{rf}}} \sum_{i=1}^{n_{\text{rf}}} \phi(x_i) \|^2 \\
&= \operatorname{argmin}_{\boldsymbol{r}} [\frac{1}{n_{\text{ts}}^2} \sum_{i,j=1}^{n_{\text{ts}}} r(x_i)\kappa(x_i, x_j) r(x_j) \\
&\quad - \frac{2}{n_{\text{ts}} n_{\text{rf}}} \sum_{i=1}^{n_{\text{ts}}} \sum_{j=1}^{n_{\text{rf}}} r(x_i)\kappa(x_i, x_j) + \frac{1}{n_{\text{rf}}^2} \sum_{i,j=1}^{n_{\text{rf}}} \kappa(x_i, x_j)] \\
&= \operatorname{argmin}_{\boldsymbol{r}} \left[\frac{1}{2} \boldsymbol{r}^T K_{x_{\text{ts}}, x_{\text{ts}}} \boldsymbol{r} - \frac{n_{\text{ts}}}{n_{\text{rf}}} \boldsymbol{r}^T K_{x_{\text{ts}}, x_{\text{rf}}} \mathbf{1} \right] ,
\end{aligned}
\tag{5.3}
$$

where

$$
\begin{aligned}
&K_{x_{\text{ts}}, x_{\text{ts}}}(i, j) = \kappa(x_i, x_j),\ \{x_i, x_j \in \mathcal{X}_{\text{ts}}\} , \\
&K_{x_{\text{ts}}, x_{\text{rf}}}(i, j) = \kappa(x_i, x_j),\ \{x_i \in \mathcal{X}_{\text{ts}}, x_j \in \mathcal{X}_{\text{rf}}\} , \\
&\mathbf{1} = [1, \ldots 1]^T .
\end{aligned}
\tag{5.4}
$$

The optimal solution of Eq. 5.3 without imposing constraints on $\boldsymbol{r}$ can be analytically obtained as:

$$
\hat{\boldsymbol{r}} = \frac{n_{\text{ts}}}{n_{\text{rf}}} K_{x_{\text{ts}}, x_{\text{ts}}}^{-1} K_{x_{\text{ts}}, x_{\text{rf}}} \mathbf{1} .
$$

For a test point $x \in \mathcal{X}_{\text{ts}}$,

$$
\hat{r}(x) = \frac{n_{\text{ts}}}{n_{\text{rf}}} \sum_{x_i \in \mathcal{X}_{\text{rf}}} \kappa(x, x_i) - \sum_{x_j \in \mathcal{X}_{\text{ts}} \setminus x} r(x_j)\kappa(x, x_j) . \tag{5.5}
$$

This means that the normality index $r(x)$ is the difference between two terms r_1 and r_2 defined as

$$
\begin{aligned}
r_1(x) &= \frac{n_{\text{ts}}}{n_{\text{rf}}} \sum_{x_i \in \mathcal{X}_{\text{rf}}} \kappa(x, x_i) , \\
r_2(x) &= \sum_{x_j \in \mathcal{X}_{\text{ts}} \setminus x} r(x_j)\kappa(x, x_j) .
\end{aligned}
$$

These two terms affect the novelty of x as follows: The first term r_1 captures the similarity between the test instance x and all reference instances $\mathcal{X}_{\text{rf}}$ (based on $\kappa(x, x_i)$).

This quantifies how x is different from the previously-seen instances and, accordingly, measures how emerging it is. If x is very dissimilar to all reference instances, then x is an emerging novelty and the value of r_1 will be very small. On the other hand, if x is very similar to many reference instances, then x is a normal instance and the value of r_1 will be very large.

The second term r_2 captures the similarity between the test instance x and other test instances $\mathcal{X}_{\text{ts}}$. This quantifies how x is a novelty relative to other new instances, depending on how similar these instances are (based on $\kappa(x, x_j)$) and how novel they are (based on $r(x_j)$). This term is a key indicator for detecting evolving concepts. To understand how, let us consider the case where a test instance x appears very close to reference data $\mathcal{X}_{\text{rf}}$. In this case, using r_1 only results in the conclusion that this point is not novel. However, if x appears within a tight cluster of other test instances, the large score of r_1 will propagate through the calculation of r for the instances of this cluster, and result in a very large value for r_2. Accordingly, r_2 will reduce the overall score $r(x)$ and lead to the conclusion that x is an evolving novelty.

5.2.2 Limitations of Density Ratio-based Measures

Although $r(x)$ captures the emerging and evolving aspects of novelty, the score of emerging novelty r_1 is very inaccurate, as it mainly depends on the absolute similarity between x and other data instances. For instance, supposing that x has an average similarity of s to all the reference instances, we cannot conclude anything about how novel x is unless we learn about how similar reference instances are to each other. If s is a common similarity in the subspace of reference instances, then x should be considered normal regardless of the absolute value of s.

To illustrate this argument, Fig. 5.2 shows two cases for a test instance x (black cross) and a set of reference instances (green triangles). For the two cases, the value of r_1 is exactly the same. However, comparing with their neighboring structures, x should be considered normal in Case A and novel in Case B. A similar argument was discussed by Breunig et al. [13].

In the next section, we discuss how to address these limitations through the use of a locally-adaptive kernel.

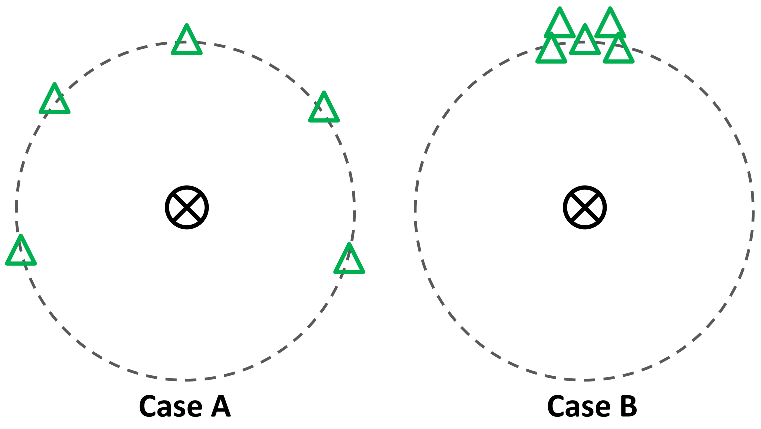


Figure 5.2: Two cases with identical r_1.

5.2.3 Locally-Adaptive Kernel Mean Matching

Emerging and evolving novelties. We categorize novelties into emerging and evolving based on these two terms. Emerging novelties are represented by concepts which are very different from the previously seen ones, while evolving novelties are characterized by relatively new aspects of existing concepts. Additionally, there are novelty of clustered instances, among which some are similar to previous and some are different. These differences on novelty are listed as:

- Evolving novelty: a tightly clustered set of test instances which shows novelty due to the relative high intensity of occurrences, but individually they are similar to previous seen ones.
- Emerging novelty: test instances which are very different from previous seen normal data.
- Novelty with a mixture of emerging and evolving aspects: a tightly clustered set of instances, among which some are similar to old concepts and some are new.

Locally-adaptive kernel. In order to address the aforementioned limitations,we need to incorporate information about the similarity of x and y to other neighboring instances in the calculation of how novel is x with respect to y. Since the novelty score is mainly based on the kernel function between x and y, one indirect way to modify the novelty score is to adjust the kernel function to reflect how x is truly dissimilar (i.e., novel) to y with respect to their neighborhood.

For instance, in the case of Gaussian kernels, we can adjust the kernel width σ to be adaptive to the local neighborhood of each pair of instances. In the rest of this section, we will focus on developing a locally-adaptive Gaussian kernel based on this idea. The same idea can be directly extended to other types of kernels by normalizing the value of the kernel function $\kappa(x, y)$ using the kernel between x and y and other neighboring points.

The Gaussian kernel function between two samples x_i and x_j is defined as

$$\kappa_\sigma(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2\sigma^2}\right) , \tag{5.6}$$

where $d(x_i, x_j)$ is a distance function between the two samples x_i and x_j, and σ is the scaling factor (bandwidth) that decides the smoothness of the kernel.

Instead of choosing one scaling factor σ for measuring similarity between all data points in the kernel space, we propose the use of a locally-adaptive kernel that captures the local density statistics of x_i and x_j. One locally-adaptive kernel that was successfully used for enhancing spectral clustering [131] is defined as:

$$\kappa_l(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2d(x_i, N_k(x_i))d(x_j, N_k(x_j))}\right) , \tag{5.7}$$

where $N_k(x_i)$ is the k-th nearest neighbor of x_i. In other words, the bandwidth of $\kappa_l(x_i, x_j)$ is the geometric mean of the k-th nearest neighbor distances for x_i and x_j:

$$\sigma_{ij} = \sqrt{d(x_i, N_k(x_i))d(x_j, N_k(x_j))} .$$

Using this locally-adaptive kernel while calculating $K_{x_{\text{ts}}, x_{\text{ts}}}$ and $K_{x_{\text{ts}}, x_{\text{rf}}}$ of Eq. 5.3, the new normality score takes into consideration two factors: relativity to normality and relativity to neighborhood. In comparison to previous approaches, the traditional density-ratio approach takes care of the first factor only, while the LOF method just focuses on the second factor.

5.2.4 Approximation Using Diagonal Shifting

According to [88], a valid reproducing kernel should satisfy the following two properties:

1. *Hermitian*: For finite data observations of real entries, the Gram matrix should be symmetric ($K_{ij} = K_{ji}$).

2. *Positive Semi-Definite (PSD)*: For finite data observations, the Gram matrix should be positive semi-definite.

As seen from Eq. 5.7, the local kernel satisfies the symmetric condition, i.e. $\kappa_l(x_i, x_j) = \kappa_l(x_j, x_i)$. But, because the matrix $K_{x_{\text{ts}},x_{\text{ts}}}$ is constructed from a locally-adaptive kernel (Eq. 5.7), the Positive Semi-Definite (PSD) of the kernel may be violated. In order to address this issue and enforce PSD, we adopt the following approximation using diagonal shifting [34]:

$$\tilde{K}_{x_{\text{ts}},x_{\text{ts}}} = K_{x_{\text{ts}},x_{\text{ts}}} + (\delta + \epsilon)\, I\,, \tag{5.8}$$

where δ is the absolute value of the minimum negative eigenvalue, and ϵ is a small value for compensating numerical error. Through diagonal shifting, the only changes to the kernel matrix are elements representing self-similarity, while all other pairs of similarities are not affected. Therefore, our intuitive idea of expressing similarity based on neighborhood structure is still preserved in $\tilde{K}_{x_{\text{ts}},x_{\text{ts}}}$; meanwhile, the PSD and symmetry are both satisfied.

As the PSD of $\tilde{K}_{x_{\text{ts}},x_{\text{ts}}}$ is held, the locally-adaptive Kernel Mean Matching (*loc*KMM) optimization is still a convex quadratic problem. Our implementation includes a boundary constraint on r and uses the well-known '`interior-point-convex`' algorithm in the Matlab toolbox as the Quadratic Programming (QP) solver. The complete *loc*KMM algorithm is outlined in Algorithm 9.

5.3 Related Work

The concept of localized kernels has been explored in some earlier works in the research community. For support vector machines, Gönen and Alpaydin [51] proposed the use of a weighted combination of multiple kernels which optimizes different weights for different regions with the localized intuition. For spectral clustering, Zelnik-Manor and Perona [131] proposed using a locally-scaling factor to construct the affinity graph for clustering purposes.

Emerging concept discovery [41] is a related task which is concerned with the modeling of new concepts or classes into previously-learned classifiers. This task is conceptually different from the novelty detection task as the former uses labeled data to learn a classification model and identify deviations from learned classes as emerging concepts (or classes).

Algorithm 9 Locally-adaptive Kernel Mean Matching

```
Input: $\mathcal{X}_{\text{rf}},\ \mathcal{X}_{\text{ts}},\ k, \epsilon, b$
Output: $\boldsymbol{r} = [r(x_1), \ldots, r(x_{n_{\text{ts}}})]^T$
Steps:
1: $\sigma_{ij} = \sqrt{d(x_i, N_k(x_i))d(x_j, N_k(x_j))}, \forall x_i \in \mathcal{X};$
2: $K_{x_{\text{ts}},x_{\text{rf}}}(i,j) = \kappa_l(x_i, x_j), \forall x_i \in \mathcal{X}_{\text{ts}}, x_j \in \mathcal{X}_{\text{rf}}$
3: $K_{x_{\text{ts}},x_{\text{ts}}}(i,j) = \kappa_l(x_i, x_j), \forall x_i, x_j \in \mathcal{X}_{\text{ts}}$
4: if not PSD($K_{x_{\text{ts}},x_{\text{ts}}}$) then
5:     $\tilde{K}_{x_{\text{ts}},x_{\text{ts}}} = K_{x_{\text{ts}},x_{\text{ts}}} + (\delta + \epsilon)\, I$ (Eq. 5.8);
6: else
7:     $\tilde{K}_{x_{\text{ts}},x_{\text{ts}}} = K_{x_{\text{ts}},x_{\text{ts}}}$
8: end if
9: $\boldsymbol{r} \leftarrow$ QP_solver $\left(\tilde{K}_{x_{\text{ts}},x_{\text{ts}}}, K_{x_{\text{ts}},x_{\text{rf}}}, \epsilon, b\right)$(Eq. 5.3).
```

In [85], both concept drift and novel class detection were considered in streaming data scenarios with a delayed labeling process. The concept drift problem was addressed by continuously updating an ensemble of classifiers to include the most recent changes. The novel class detection was handled by enriching each classifier in the ensemble with a novelty descriptor. In [41], active learning was employed to discover new categories based on user feedback and learn models of them, while pursuing minimal labeling efforts.

5.4 Experiments

The empirical study to evaluate the proposed novelty detection method is considered under two scenarios, emerging and evolving novelty detection, using two real life datasets. The scenario that contains both emerging and evolving novelties will be described in the next section in the context of social media analysis task.

5.4.1 Evaluation Criteria

Evaluation measures play a crucial role in assessing the novelty detection performance and guiding the model fitting. In the literature of novelty detection, Receiver Operating Characteristic (ROC) curves, Area Under ROC Curves (AUC), and Precision-at-t (Prec@t) are commonly used evaluation measures.

Receiver Operating Characteristic (ROC) curve. An ROC curve captures the tradeoff between true positives and false alarms [45]. When a novelty detector assigns scores for test instances, the prediction of novelty can be changed by varying the threshold. Each threshold value generates a pair of measurements, False Positive rate (FPrate) and True Positive rate (TPrate). By linking these measurements with the FPrate as the x-axis and the TPrate as the y-axis, a ROC curve is plotted. The ideal model is the one that obtains TPrate $= 1$ and FPrate $= 0$. The ROC curve gives a summary of the performance of a detection model.

Area Under an ROC Curve (AUC). To compare several models using ROC curves, it is hard to declare a winner unless one curve clearly dominates the others over the entire space. The Area Under an ROC Curve (AUC) provides a single measure on detecting performance for evaluating which model is better on average.

Precision at t (Prec@t). The Precision-at-t (Prec@t) [106] is defined as the precision rate among the t top-ranked novel instances, where t is the number of ground truth novel instances. The Prec@t originated from the literature of ranking-based information retrieval, where the first few results are the most important for users. The Prec@t measure depicts the starting section of ROC curves.

5.4.2 Experimental Setup

Datasets. The USPS handwritten digits collection and the spam emails collection were used to evaluate different novelty detection algorithms. The properties of the two datasets are summarized in Table 5.1.

The USPS dataset is from the libSVM archive[1], which contains 9,298 handwritten digit images. Each sample is of size 16×16 pixels with intensity levels in the range $[-1, 1]$. This dataset has a standard split of 7,291 training samples and 2,007 test samples. We synthesized the novel digit detection tasks as follows. The reference data $\mathcal{X}_{\text{rf}}$ was constructed from the original training set by excluding one digit in each setup, while all 2,007 samples in the test set were used as the test data $\mathcal{X}_{\text{ts}}$. Thus, the extra digit shown in the test collection is the novel concept to be detected.

The spam email collection is from the UCI archive[2]. The dataset contains 4,601 samples. Each sample is described by 57 features, representing the frequencies of particular words

[1]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[2]https://archive.ics.uci.edu/ml/datasets/Spambase

Table 5.1: The property of datasets used in novelty detection.

Dataset	**# Samples**	**# Feature**	$n_{\mathbf{rf}}$	$n_{\mathbf{ts}}$	**Detection Task**
USPS	9,298	256	7,291	2,007	images of new digit
SPAM	4,601	57	$\rho\%$	$(100-\rho)\%$	spam emails

or characters. The reference data $\mathcal{X}_{\text{rf}}$ were formed by taking a percentage ρ of instances that contain only the non-spam emails. The test data $\mathcal{X}_{\text{ts}}$ were formed from the remaining instances that contain both spam and non-spam emails. The task was to detect the spam emails with the reference non-spam collection.

Comparison methods. The following novelty detection methods were included for comparison. Their implementation details are set as follows.

- **OSVM:** One-class Support Vector Machine [100]. The LibSVM library[3] was used. The Gaussian kernel was adopted, the kernel width was set to the median distance between samples, and the parameter v was set to 0.1.
- **LOF:** Local Outlier Factor method [13]. The neighborhood size k was set to 7, the same as the *loc*KMM method.
- **KDE:** Kernel Density Estimator [116]. This method was used to estimate the PDF of the reference data, and then the novelty indicator was derived as the inverse of the PDF for the test data. The Gaussian kernel was used, and its width was selected using 10-fold cross validation.
- **uLSIF:** unconstrained Least Squares Importance Fitting [59]. This is a well-known density-ratio estimation method. The Gaussian kernel was used and its width was selected using 10-fold cross validation.
- **KMM:** Kernel Mean Matching [62]. This is another well-known density-ratio estimation method. As in [62], the Gaussian kernel was used and its kernel width was set to the median distance between samples.
- ***loc*KMM:** locally-adaptive Kernel Mean Matching. This is the proposed method. The neighborhood size k was set to 7. The effect of other values of k is reported in Section 5.4.5. For KMM and *loc*KMM, we set $b = 1000$ and $\epsilon = 1e - 10$.

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Table 5.2: AUC of different novelty detection methods on USPS dataset.

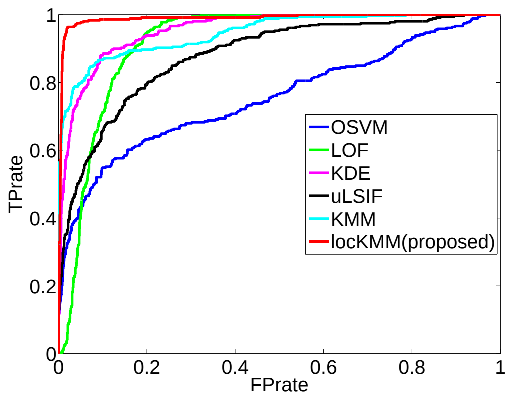
TargetClass	OSVM	LOF	KDE	uLSIF	KMM	locKMM
0	0.7545	0.9186	0.9548	0.8792	0.9463	**0.9891**
1	0.9120	0.7248	0.2939	0.7156	0.9147	**0.9881**
2	0.6422	0.9586	**0.9592**	0.7557	0.8303	0.9463
3	0.4820	0.8575	0.9154	0.6349	0.8365	**0.9558**
4	0.6989	0.9586	0.8119	0.7058	0.8599	**0.9701**
5	0.5585	0.9249	**0.9291**	0.6393	0.8121	0.9076
6	0.4607	0.9635	0.8970	0.6685	0.8226	**0.9781**
7	0.7419	0.9564	0.7685	0.6864	0.8484	**0.9744**
8	0.3040	**0.9211**	0.8842	0.5299	0.8104	0.9204
9	0.3469	0.8375	0.6202	0.5124	0.8336	**0.9551**
Avg	*0.5902*	*0.9021*	*0.8034*	*0.6728*	*0.8515*	*0.9585*
#wins	-	*1*	*2*	-	-	*7*

5.4.3 Emerging Novelty Detection

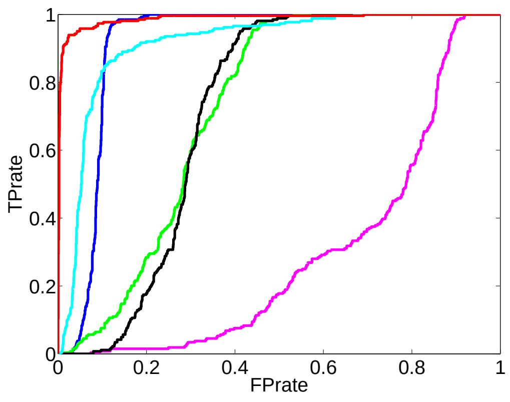
Results of novel digit detection. Introducing each digit as a novel concept in the test collection, Fig. 5.3 and 5.4 plot the ROC curves of different novelty detection algorithms. As can be observed, the proposed *loc*KMM method outperformed the other methods except LOF and KDE for the digit 2, 5, and 8, for which it had tied performance. Especially, the *loc*KMM dominated during the beginning stage in all cases. This merit is clearly shown in the quantitative results when evaluated with AUC and Prec@t. As listed in Table 5.2 and 5.3, the *loc*KMM achieved the best performance in 7 cases out of 10 in terms of AUC, and the highest Prec@t in all cases.

Results of spam email detection. This task uses non-spam emails as the reference collection, and the emerging novelties to detect are the spam emails. The ROC detection curves are plotted in Fig. 5.5, with changing the percentage of samples that were reserved for the reference collection, $\rho = 30\%, 50\%, 70\%$. In all the three cases, the proposed *loc*KMM method produced the best ROC curves.

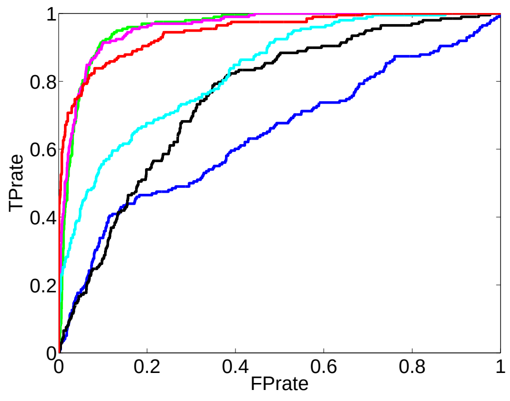
The quantitative results of AUC and Prec@t are listed in Table 5.4. Experiments were repeated 30 times by randomly splitting the data according to the different percentage of the three cases. The significance of improvement was tested using the Friedman test [33].



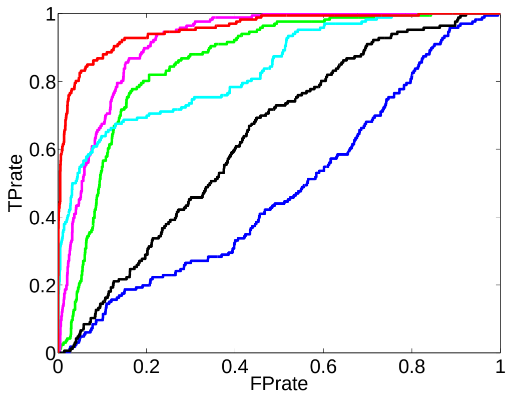
(a) Digit 0



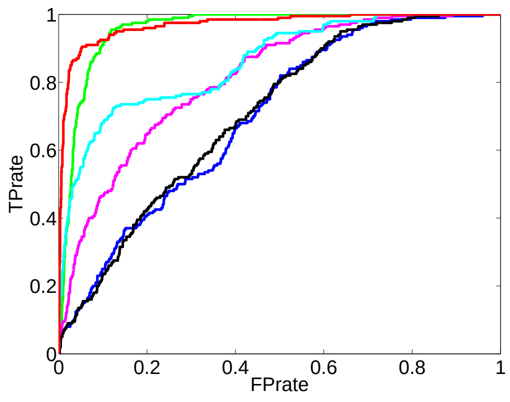
(b) Digit 1



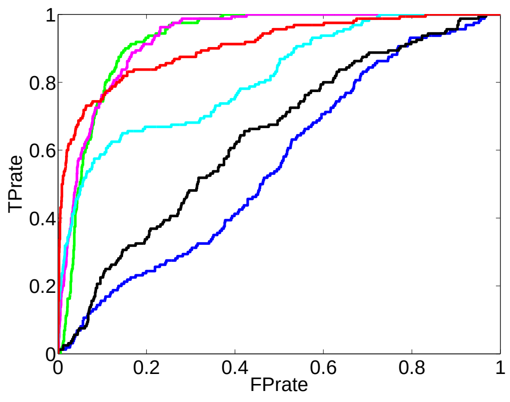
(c) Digit 2



(d) Digit 3

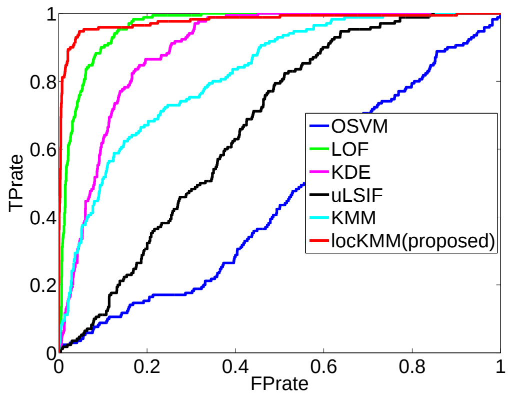


(e) Digit 4

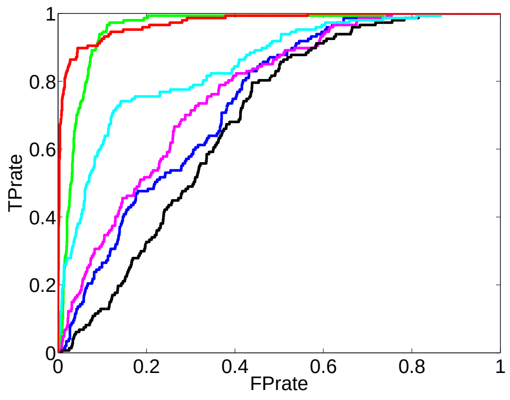


(f) Digit 5

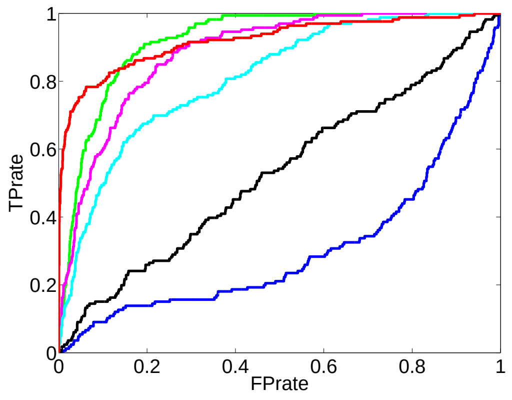
Figure 5.3: The ROC curves of different novelty detection methods for digit 0 to 5.



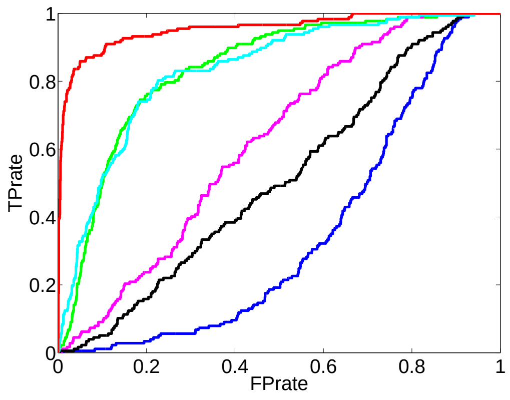
(a) Digit 6



(b) Digit 7



(c) Digit 8



(d) Digit 9

Figure 5.4: The ROC curves of different novelty detection methods for digit 6 to 9.

Table 5.3: Prec@t of different novelty detection methods on USPS dataset.

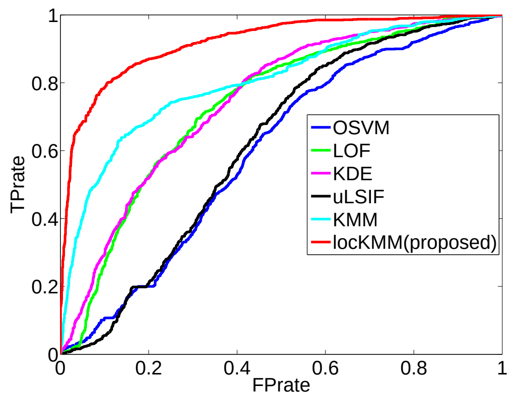
TargetClass	**OSVM**	**LOF**	**KDE**	**uLSIF**	**KMM**	**locKMM**
0	0.5460	0.6351	0.7632	0.6086	0.7939	**0.9359**
1	0.4356	0.1174	0.0152	0.0587	0.6174	**0.9091**
2	0.2778	0.6768	0.6768	0.2576	0.4545	**0.7222**
3	0.0843	0.3072	0.4518	0.1386	0.5181	**0.7470**
4	0.2100	0.6650	0.3900	0.1875	0.5500	**0.8000**
5	0.1250	0.4750	0.5063	0.1281	0.4625	**0.6313**
6	0.0765	0.6647	0.3824	0.0971	0.3765	**0.8353**
7	0.1905	0.5782	0.2313	0.0918	0.3810	**0.8095**
8	0.0904	0.5120	0.4398	0.1446	0.3554	**0.7048**
9	0.0113	0.3277	0.0791	0.0395	0.3559	**0.7740**
Avg	*0.2047*	*0.4959*	*0.3936*	*0.1752*	*0.4865*	*0.7869*
#wins	-	-	-	-	-	*10*

It can be observed that the proposed *loc*KMM method consistently outperformed the other competitors with a great advantage. The AUC is increased by at least 12% and the Prec@t is increased by at least 14% in comparison with the second best method. This is different from the above novel digits detection tasks. Even spam emails show abnormal concepts with respect to normal emails, their vocabularies usually have high amounts of overlap. The inseparability in the representation space indicates that the spam emails contain the characteristic of both emerging and evolving novelties. Thus, the LOF method, usually a great performer for emerging novelty detection, loses the ability to identify spam emails effectively.

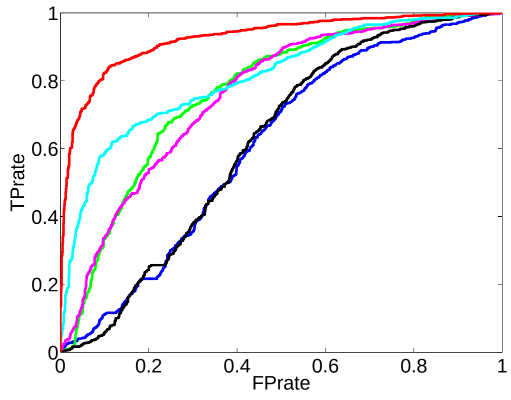
5.4.4 Evolving Novelty Detection

To explicitly evaluate the detection of evolving novelties, a set of experiments based on the spam emails collection was organized as follows. We intentionally added a small fraction of spam emails into the reference data. Thus, the spam emails in the test collection would not show as being absolutely novel. Instead, a much higher proportion in the test collection simulates the evolving novel. This may demonstrate a realistic scenario of having noisy samples in the reference collection in training one-class classifiers for novelty detection.

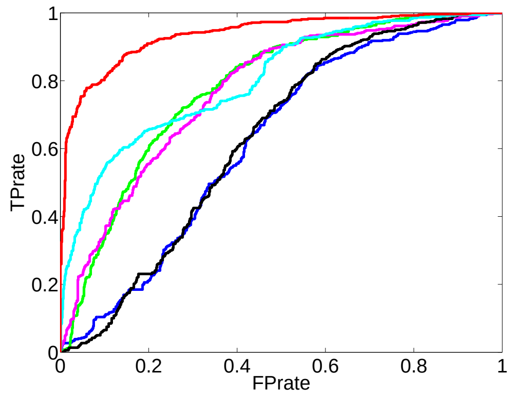
With the reference/test data split as $\rho = 50\%$ and different fractions of spam emails



(a) $\rho = 30\%$



(b) $\rho = 50\%$



(c) $\rho = 70\%$

Figure 5.5: The ROC curves of different methods for spam email detection with data splits as ρ =30% (up), 50% (middle), 70% (bottom).

Table 5.4: The quantitative results of spam email detection. The best performing methods (according to the Friedman test at a confidence interval of 95%) are highlighted in bold.

	ρ	**OSVM**	**LOF**	**KDE**	**uLSIF**	**KMM**	**locKMM**
AUC	30%	0.6074±.0140	0.7234±.0134	0.7449±.0054	0.6198±.0106	0.8008±.0165	**0.9258±.0092**
	50%	0.6112±.0104	0.7584±.0105	0.7592±.0055	0.6216±.0109	0.8041±.0114	**0.9362±.0067**
	70%	0.6119±.0122	0.7729±.0101	0.7697±.0102	0.6255±.0175	0.8023±.0156	**0.9307±.0088**
Prec@t	30%	0.4651±.0153	0.5941±.0166	0.6032±.0110	0.4755±.0137	0.6858±.0206	**0.8266±.0146**
	50%	0.4661±.0150	0.6275±.0144	0.6159±.0114	0.4765±.0142	0.6787±.0171	**0.8431±.0116**
	70%	0.4688±.0152	0.6410±.0143	0.6283±.0150	0.4807±.0203	0.6611±.0184	**0.8330±.0148**

being included into the reference collection, $\tau = 1\%, 2\%, 5\%, 10\%$, the results are presented in Table 5.5. As a reference, the table also includes the results of the case of no spam emails in the reference collection, i.e. $\tau = 0$. Experiments of each test case were repeated 30 times and results were tested using the Friedman significance test.

From the table, it is not surprising that the performance of all methods declines when increasing the fraction of added spam emails. However, we can categorize these methods into two groups. The methods of LOF, KDE, and KMM are highly affected by the amount of added spam email, while the methods of OSVM, uLSIF, and locKMM are more robust to the impurity of reference data.

5.4.5 Effect of Neighborhood Size

The performance of the *loc*KMM and LOF algorithms relies on the neighborhood size k, which was heuristically set to 7 in all previous experiments. To examine the sensitivity of these methods to different values of k, we varied k in the range from 1 to 100 for the spam email dataset with $\rho = 50\%$ setting. Experiments were repeated 30 times for each setup.

Fig. 5.6 plots the average AUC versus the neighborhood size k. It can be observed that the AUC of *loc*KMM is very stable in a wide range of k from 4 to 30 compared to LOF. When k is extremely small ($k \leq 3$), the performance degrades for both methods, as the density ratio estimation overfits the data and it is easily affected by noisy samples. When k is too large ($k \geq 40$), the novelty of each sample is affected by many unrelated samples which also negatively affect both methods. This demonstrates that the proposed *loc*KMM method is insensitive to changes in the neighborhood size.

Table 5.5: The performance of spam email detection with simulated evolving novelty as $\tau = 0, 1\%, 2\%, 5\%, 10\%$. The best performing methods (according to the Friedman test at a confidence interval of 95%) are highlighted in bold.

	τ	**OSVM**	**LOF**	**KDE**	**uLSIF**	**KMM**	**locKMM**
AUC	0	0.6112±.0104	0.7584±.0105	0.7592±.0055	0.6216±.0109	0.8041±.0114	**0.9362±.0067**
	1%	0.6085±.0109	0.7511±.0111	0.7483±.0065	0.6189±.0115	0.7862±.0119	**0.9349±.0068**
	2%	0.6071±.0106	0.7435±.0110	0.7399±.0072	0.6220±.0132	0.7735±.0144	**0.9332±.0071**
	5%	0.6018±.0105	0.7180±.0148	0.7150±.0083	0.6207±.0132	0.7403±.0171	**0.9283±.0072**
	10%	0.5937±.0121	0.6762±.0179	0.6851±.0103	0.6177±.0105	0.6998±.0203	**0.9194±.0079**
	↓	1.75%	8.23%	7.40%	0.39%	10.44%	1.68%
Prec@t	0	0.4661±.0150	0.6275±.0144	0.6159±.0114	0.4765±.0142	0.6787±.0171	**0.8431±.0116**
	1%	0.4643±.0152	0.6181±.0163	0.6059±.0124	0.4740±.0162	0.6581±.0152	**0.8400±.0124**
	2%	0.4634±.0145	0.6091±.0168	0.5998±.0122	0.4761±.0172	0.6423±.0200	**0.8376±.0120**
	5%	0.4599±.0143	0.5784±.0213	0.5797±.0143	0.4757±.0166	0.6039±.0235	**0.8292±.0123**
	10%	0.4536±.0163	0.5369±.0241	0.5514±.0155	0.4731±.0136	0.5571±.0276	**0.8144±.0126**
	↓	1.25%	9.06%	6.45%	0.34%	12.16%	2.87%

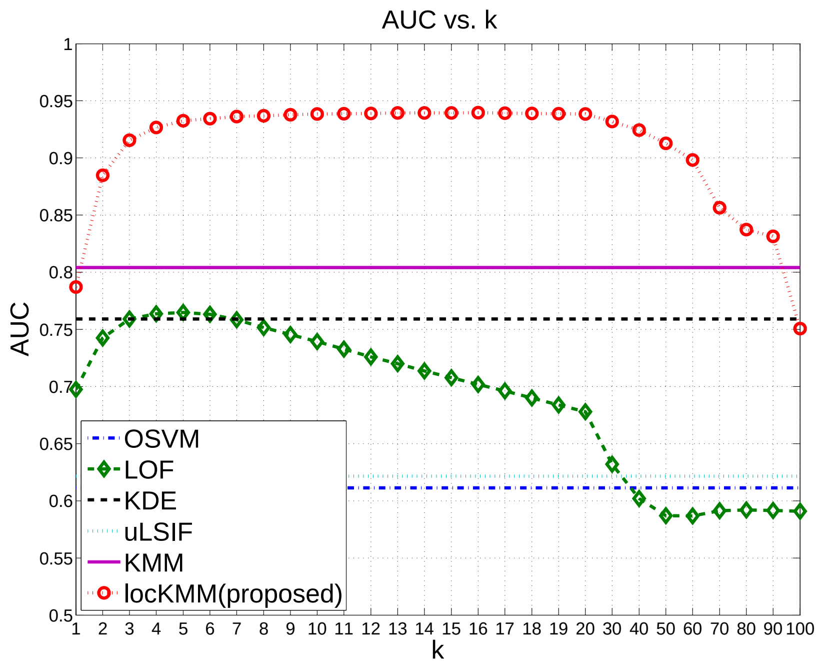


Figure 5.6: The performance of spam detection vs. neighborhood size.

5.5 Application to Social Media Analysis

With the increasing popularity of social networks, a huge amount of diverse and dynamic information is continually being generated. Mining this rich information and analyzing the trend of topics in social media has the potential to be useful in many aspects, such as for helping political parties and companies to understand people's opinions, responding to customer needs, or even discovering natural or social disasters as early as possible [89]. Accurately detecting novel content from this short text in a timely fashion is an important task, which involves identifying novel instances that include new topics or have sudden increases of intensity on existing topics in comparison to the past.

As the time continuity of social media streams, the novelty is usually characterized by the combination of emerging and evolving. One reason is the existence of large common vocabularies between different topics. Another reason is that there is a high possibility of topics being continuously discussed in sequential batches of collections, but showing different level of intensity. Thus, the social media data is a perfect example to examine the presented novelty detection algorithms.

5.5.1 Semantic Representation of Very Short Text

The conventional text representation is the Vector Space Model (VSM), which represents documents with a document-term matrix. A big disadvantage of the VSM representation is its sparsity along with very high dimensionality. This becomes extremely severe when dealing with short social media data. For example, the maximum length of twitter messages is 140 characters only, with the average length being approximately 15 words per tweet [95]. In English texts, the dimension can easily reach to 10K even when the techniques of stop word removing and stemming are employed.

To address the extreme sparse and high dimension problem in short social media data, the application of low-rank semantic kernels is proposed. This method first builds a semantic kernel based on term-term correlation [44]. Then, a Nyström approximation [42] is applied to extract low-rank representations. In our following experiments, the low rank number is set to 500.

5.5.2 Tweet Data and Results

Dataset and test scenario. The tweet dataset used in this experiments is described in [136], which includes 369K tweets spreading over 10 topics. Two test scenarios are

Table 5.6: The tweet dataset and test scenario.

Scenario		Topic	n_{rf}	n_{ts}
S1	Stable	fashion	1000	1000
	Emerging	investing	0	1000
	Evolving	media	100∼200	1000
S2	Stable	music, religion, shopping	3000	3000
	Emerging	sports, technology	0	2000
	Evolving	travel, video-games	200∼400	2000

formulated to include both emerging and evolving topics, as listed in Table 5.6.

Baseline methods. To detect both the emerging and evolving novelties, the straightforward cascaded approach is formulated as baselines, that is step-1: using LOF or OSVM to detect the emerging novelties, and step-2: using density-ratio (KMM or uLSIF) method to detect the evolving novelties. Therefore, these combinations lead to four baseline methods as: LOF+uLSIF, LOF+KMM, OSVM+uLSIF, and OSVM+KMM.

In addition, the four novelty detection algorithms LOF, OSVM, KMM, uLSIF are also included for comparison. The KDE method is not included because it is not scalable for the high dimension tweet data, even when its dimensionality is greatly reduced to 500 using the low-rank semantic kernel representation as described above. The detection results in terms of AUC and Prec@t on the two scenarios are presented in Table 5.7.

Further, we evaluate the detection results by precision, recall, F1, and F0.5 by thresholding the novelty scores. With a contingency table (Table 5.8), defining

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} ,$$

F1 is the harmonic mean of precision and recall as

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} .$$

Table 5.7: Average AUC and Prec@t of 10 runs on the tweet data.

Test Scenario	**S1**		**S2**	
	AUC	**Pre@t**	**AUC**	**Pre@t**
LOF	0.6001±.0163	0.7309±.0080	0.5190±.0102	0.5953±.0099
OSVM	0.8255±.0152	0.8100±.0141	0.7173±.0096	0.7069±.0068
uLSIF	0.8676±.0067	0.8494±.0079	0.7087±.0323	0.7018±.0231
KMM	0.8170±.0083	0.8174±.0056	0.7553±.0084	0.7387±.0063
LOF+uLSIF	0.7737±.0115	0.8246±.0050	0.5940±.0138	0.6477±.0070
LOF+KMM	0.7718±.0117	0.8157±.0061	0.7061±.0106	0.7228±.0070
OSVM+uLSIF	0.8749±.0065	0.8511±.0063	0.7312±.0302	0.7145±.0226
OSVM+KMM	0.8332±.0101	0.8202±.0064	0.7617±.0072	0.7406±.0066
locKMM	**0.9339±.0043**	**0.9016±.0069**	**0.8421±.0166**	**0.7975±.0161**

Table 5.8: The contingency table of prediction.

Ground Truth	Positive	Negative
Predicted Positive	true-positive (tp)	false-positive (fp)
Predicted Negative	false-negative (fn)	true-negative (tn)

F0.5 is a measure biased to precision by setting $\beta = 0.5$ in the equation of

$$F_\beta = \frac{(1+\beta^2)\cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} .$$

Table 5.9 reports the average detection performance of 10 runs. Each method is by varying the threshold of novelty scores and reports the best result in terms of F0.5. From Table 5.7 and 5.9, the superiority of *loc*KMM method is obvious, while the simple cascading methods do not have satisfactory performance. The main reason lies in the fact that tweets from emerging topics may also demonstrate evolving aspects, which include overlaps with previous concepts. Another observation is that the *loc*KMM method maintains a high level of accuracy with reasonable recall values. This is a preferable feature in information retrieval applications as the novelty detection. The relative low recall values mean that a number of tweets from novel topics are not distinguishable from existing topics. This is likely caused by the limitation of the representation model.

Table 5.9: Detection performance in terms of precision, recall, F1, and F0.5.

Test Scenario	**S1**			
	P	**R**	**F1**	**F0.5**
LOF	0.7148±.0066	0.9179±.0225	0.8036±.0108	0.7478±.0074
OSVM	0.8428±.0082	0.7880±.0264	0.8142±.0153	0.8311±.0094
uLSIF	0.8882±.0116	0.7553±.0366	0.8158±.0197	0.8576±.0091
KMM	0.8324±.0055	0.7730±.0188	0.8015±.0101	0.8197±.0056
LOF+uLSIF	0.8254±.0069	0.8191±.0336	0.8218±.0157	0.8239±.0060
LOF+KMM	0.8280±.0068	0.7756±.0219	0.8007±.0121	0.8168±.0072
OSVM+uLSIF	0.8966±.0112	0.7448±.0296	0.8133±.0163	0.8612±.0085
OSVM+KMM	0.8863±.0117	0.6575±.0295	0.7546±.0202	0.8283±.0129
locKMM	0.9558±.0041	0.7593±.0154	**0.8462±.0096**	**0.9087±.0052**
Test Scenario	**S2**			
	P	**R**	**F1**	**F0.5**
LOF	0.6003±.0061	0.8489±.0145	0.7032±.0073	0.6376±.0061
OSVM	0.7258±.0089	0.6585±.0138	0.6905±.0101	0.7112±.0089
uLSIF	0.6650±.0305	0.8466±.0445	0.7435±.0123	0.6941±.0208
KMM	0.7515±.0072	0.6963±.0104	0.7228±.0066	0.7397±.0061
LOF+uLSIF	0.6258±.0050	0.8569±.0543	0.7225±.0193	0.6611±.0058
LOF+KMM	0.7267±.0077	0.7092±.0106	0.7178±.0065	0.7231±.0064
OSVM+uLSIF	0.6883±.0479	0.8050±.0827	0.7372±.0233	0.7058±.0218
OSVM+KMM	0.7518±.0067	0.7052±.0108	0.7277±.0066	0.7419±.0058
locKMM	0.8261±.0187	0.7228±.0102	**0.7710±.0133**	**0.8031±.0163**

5.6 Discussion on Computation Complexity

This section analyzes the computation complexity of the proposed locKMM algorithm and the comparable methods. Following, the experimental running time with the tweets data is reported.

We first setup the computational complexity for several common modules. Let n be the number of samples and d be the dimensions. The k-NN algorithm has computational complexity of $\mathcal{O}(nd+kn)$. The computation of kernel matrix has the complexity of $\mathcal{O}(n^2d)$. The computation of matrix inverse has the complexity of $\mathcal{O}(n^3)$. The convex quadratic programming problem has the complexity of $\mathcal{O}(n^3)$. We noticed that there are some variances which have improved efficiency [6]. The complexities presented here act as the worst boundaries.

Let $n_{\text{rf}}, n_{\text{ts}}, d$ be the number of samples in the reference collection, test collection, and the dimension of the data, respectively. For the LOF algorithm, with k denoting the neighborhood size the complexity for testing a single data point is $\mathcal{O}(kn_{\text{rf}}d + k^2n_{\text{rf}}d)$. Then, for all test points the complexity is $\mathcal{O}(n_{\text{ts}}n_{\text{rf}}d(k^2+k))$. As can be seen, if k takes a large value the LOF method would be computationally expensive. This is due to the recursive searching of k nearest neighbors in the LOF method.

The OSVM method includes a training phase and a test phase. In the training phase, the kernel matrix needs a computation of $\mathcal{O}(n_{\text{rf}}^2d)$. The QP problem has the complexity of $\mathcal{O}(n_{\text{rf}}^3)$. In the test phase, by denoting n_{sv} as the number of support vectors, the complexity to test all n_{ts} samples takes $\mathcal{O}(n_{\text{ts}}n_{\text{sv}}d)$. Taking the worst case that $n_{\text{sv}} \approx n_{\text{rf}}$, the test complexity is $\mathcal{O}(n_{\text{ts}}n_{\text{rf}}d)$. Then, the total computational complexity for OSVM is $\mathcal{O}(n_{\text{rf}}^3 + n_{\text{rf}}^2d + n_{\text{ts}}n_{\text{rf}}d)$.

The uLSIF method also includes a model training phase and a test phase. Refer to Section 3.2.4 in Chapter 3, the computation of $H_{ll'}$ needs $\mathcal{O}(b^2n_{\text{ts}}d)$ and computation of h_l needs $\mathcal{O}(bn_{\text{rf}}d)$, where b is the number of Gaussian bases in modeling the density-ratio functions. The computation of $\hat{\boldsymbol{\alpha}}$ involves matrix inverse and has the complexity of $\mathcal{O}(b^3 + b^2)$. Be aware that uLSIF has its model selection process. Assume there are s candidates for model selection, the total complexity in the training phase is $\mathcal{O}(s(b^2n_{\text{ts}}d + bn_{\text{rf}}d + b^3 + b^2))$. The complexity of test phase is $\mathcal{O}(n_{\text{rf}}bd)$. The total computational complexity for uLSIF is $\mathcal{O}(s(b^2n_{\text{ts}}d + bn_{\text{rf}}d + b^3 + b^2) + n_{\text{rf}}bd)$.

For the KMM algorithm, the computation of the kernel matrix $K_{x_{\text{ts}},x_{\text{ts}}}, K_{x_{\text{ts}},x_{\text{rf}}}$ is $\mathcal{O}(n_{\text{ts}}^2d + n_{\text{ts}}n_{\text{rf}}d)$. The QP problem has the complexity of $\mathcal{O}(n_{\text{ts}}^3)$. Then, the total computational complexity for KMM is $\mathcal{O}(n_{\text{ts}}^3 + n_{\text{ts}}^2d + n_{\text{ts}}n_{\text{rf}}d)$.

Table 5.10: The computational complexity of different novelty detection algorithms.

	Train	**Test**	**Total**
LOF	-	-	$\mathcal{O}(n_{\mathrm{ts}}n_{\mathrm{rf}}d(k^2+k))$
OSVM	$\mathcal{O}(n_{\mathrm{rf}}^3+n_{\mathrm{rf}}^2d)$	$\mathcal{O}(n_{\mathrm{ts}}n_{\mathrm{rf}}d)$	$\mathcal{O}(n_{\mathrm{rf}}^3+n_{\mathrm{rf}}^2d+n_{\mathrm{ts}}n_{\mathrm{rf}}d)$
uLSIF	$\mathcal{O}(s(b^2n_{\mathrm{ts}}d+bn_{\mathrm{rf}}d+b^3+b^2))$	$\mathcal{O}(n_{\mathrm{rf}}bd)$	$\mathcal{O}(s(b^2n_{\mathrm{ts}}d+bn_{\mathrm{rf}}d+b^3+b^2)+n_{\mathrm{rf}}bd)$
KMM	-	-	$\mathcal{O}(n_{\mathrm{ts}}^3+n_{\mathrm{ts}}^2d+n_{\mathrm{ts}}n_{\mathrm{rf}}d)$
locKMM	-	-	$\mathcal{O}(n_{\mathrm{ts}}^3+n_{\mathrm{ts}}^2d+n_{\mathrm{ts}}n_{\mathrm{rf}}d+(n_{\mathrm{rf}}+n_{\mathrm{ts}})d+k(n_{\mathrm{rf}}+n_{\mathrm{ts}}))$

In the proposed locKMM algorithm, there is an overhead for computing the local bandwidth σ_{ij}, which is $\mathcal{O}((n_{\mathrm{rf}}+n_{\mathrm{ts}})d+k(n_{\mathrm{rf}}+n_{\mathrm{ts}}))$. The other components have the same computation needs as KMM. Therefore, the total computational complexity for locKMM is $\mathcal{O}(n_{\mathrm{ts}}^3+n_{\mathrm{ts}}^2d+n_{\mathrm{ts}}n_{\mathrm{rf}}d+(n_{\mathrm{rf}}+n_{\mathrm{ts}})d+k(n_{\mathrm{rf}}+n_{\mathrm{ts}}))$.

Table 5.10 summarizes the computational complexities of these five methods. The OSVM and uLSIF are model-based methods which include two phases, model training and the test, while LOF, KMM and locKMM are non-parametric methods, which do not produce explicit models. Assuming $n_{\mathrm{rf}}=\mathcal{O}(n_{\mathrm{ts}})=n, b=\mathcal{O}(\sqrt{n}), k=\mathcal{O}(\sqrt{n}), d<<n, s<<n$, the uLSIF method has the lowest complexity as $\mathcal{O}(n^2)$, the other four methods have the similar level of complexity as $\mathcal{O}(n^3)$. For the four cascaded methods (LOF+KMM, LOF+uLSIF, OSVM+KMM, OSVM+uLSIF), their computational complexities are straightforward, that is the sum of two corresponding algorithms.

Table 5.11 reports experimental running time of the scenario 'S1' and 'S2' on the tweet data (Section 5.5.2). The results are based on a commercial desktop using one core of its i-7 CPU. All methods are implemented in Matlab except the OSVM. It can be seen that these results are consistent with the above complexity analysis. The OSVM has the same level of computational complexity as KMM and locKMM. Because OSVM is based on a C++ implementation taken from the LibSVM library [21], the execution time for OSVM and the cascaded methods involved with OSVM are not included for comparison.

Table 5.11: Running time of the tweet data experiments (average of 10 runs, in seconds).

Test Scenario	**S1**	**S2**
LOF	14.8	110.6
uLSIF	3.6	11.6
KMM	27.4	172.3
locKMM	27.3	182.8
LOF+uLSIF	19.1	123.3
LOF+KMM	43.1	284.6

Chapter 6

Parameter Tuning for Kernel Mean Matching

Although the work on estimating density ratios in a discriminative manner considerably enhances the performance of density-ratio estimators, the category of kernel mean matching methods depend on the choice of a kernel whose parameters are hard to estimate. In order to address this problem, we proposed an auto-tuning method for the kernel-based non-parametric density-ratio estimators by introducing a novel measure for assessing the quality of candidate choices.

The chapter is organized as follows. Section 6.1 introduces the problem of parameter selection in the kernel mean matching density-ratio estimators. Section 6.2 presents a parameter tuning method based on the new defined Normalized Mean Squared Error (NMSE) measure. Section 6.3 presents an empirical evaluation. Lastly, the method is examined by extending it to polynomial kernels in Section 6.4.

6.1 Introduction

In the context of covariate shift, there is distributional divergence between the training and test data ($p_{tr}(x) \neq p_{ts}(x)$, but $p_{tr}(y|x) = p_{ts}(y|x)$). To match changed distributions, the Kernel Mean Matching (KMM) method reweights sample importance by minimizing the mean discrepancy in a Reproducing Kernel Hilbert Space (RKHS) [62].

The KMM algorithm shows elegance in theory, and is not specific to any distribution or density-ratio model. However, KMM lacks a systematic mechanism for tuning parameters.

The heuristic choices, such as adopting the median of sample pairwise distances as a Gaussian kernel's width has neither strong theoretical justification nor has it been supported in practice [53, 108]. Furthermore, there does not exist a clue regarding the choices for other types of kernels, such as a polynomial kernel.

Yu et al. [128] analyzed the convergence rate of KMM and revealed that the selection of kernel highly affects the performance of KMM. This study concludes that in order to maintain a certain level of accuracy on the matching results, KMM would require a huge volume of training and test samples if the kernel does not interact well with the given data.

In order to choose a good kernel, one traditional approach is to use weighted Cross-Validation (CV) applied to the subsequent learners. This approach, however, does not achieve good results. The reason behind this was explored in the previous work of [110], who have shown that in learning covariate shift adaptation systems the model selection of importance estimation should be separated from the model selection of subsequent learners. If combining the two steps of model selection by the weighted CV based on the final learning system, the CV score would be estimated with bias inside the loop and accordingly the result is highly unreliable.

In the following sections, we present a new quality measure for conducting kernel or parameter tuning.

6.2 Proposed Auto-tuning Method

6.2.1 Parameters in KMM

For the convenience of following descriptions, we quickly review the formulation of the Kernel Mean Matching algorithm. KMM estimates density ratios by minimizing the Maximum Mean Discrepancy (MMD) [52] between the weighted distribution $p_{tr}(x)$ and the distribution $p_{ts}(x)$ in a Reproducing Kernel Hilbert Space (RKHS) $\phi(x) : x \to \mathcal{F}$,

$$\mathrm{MMD}^2\left(\mathcal{F}, \left(\beta, p_{tr}\right), p_{ts}\right) = \left\| E_{x \sim p_{tr}(x)}\left[\beta(x) \cdot \phi(x)\right] - E_{x \sim p_{ts}(x)}\left[\phi(x)\right] \right\|^2 . \tag{6.1}$$

Using empirical means of $\mathcal{X}_{tr}$ and $\mathcal{X}_{ts}$ to replace the expectations, we can obtain a

Quadratic Programming (QP) problem as

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \text{argmin}_{\boldsymbol{\beta}} \left[\text{MMD}^2\left(\mathcal{F}, (\beta, p), p'\right)\right] \\ &= \text{argmin}_{\boldsymbol{\beta}} \left[\frac{1}{2}\boldsymbol{\beta}^T K \boldsymbol{\beta} - \boldsymbol{k}^T \beta\right] ,\end{aligned} \tag{6.2}$$

with respect to two constraints

$$\begin{gathered}\boldsymbol{\beta}_i \in [0, b]\ i = 1, \ldots, n_{tr},\ \text{and} \\ \left|\tfrac{1}{n_{tr}} \textstyle\sum_{i=1}^{n_{tr}} \boldsymbol{\beta}_i - 1\right| \leq \epsilon ,\end{gathered}$$

where K is a kernel matrix defined on $\mathcal{X}_{tr}$ as

$$K_{ij} = k(x_i, x_j),\ \{x_i, x_j \in \mathcal{X}_{tr}\} , \tag{6.3}$$

and $\boldsymbol{k}$ is a vector defined on the kernel between $\mathcal{X}_{tr}$ and $\mathcal{X}_{ts}$ as

$$\boldsymbol{k}_i = \frac{n_{tr}}{n_{ts}} \sum_{j=1}^{n_{ts}} k(x_i, x'_j),\ \{\{x_i \in \mathcal{X}_{tr},\ x'_j \in \mathcal{X}_{ts}\} . \tag{6.4}$$

The KMM algorithm involves the following factors: the boundary b, the normalization precision ε, and most importantly the kernel parameters.

The boundary b. This reflects the discrepancy between the two distributions to be matched, and acts as a constraint that limits the range of the estimation to $\beta \in [0, b]$. Therefore, the parameter b is expected to be different case by case, according to the given problem. A setting of $b = 1000$ is reasonable for most applications as suggested by [62].

The normalization precision ε. Referring to Lemma 3 in [62], the normalization constraint $\int \beta(x) p_{tr}(x) dx = 1$ is applied and the empirical estimate is used as $|\frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta_i - 1| \leq \varepsilon$, where the parameter ε reflects the normalization precision. [62] explained the convergence consideration of ε should be $\mathcal{O}\left({}^{b}/_{\sqrt{n_{tr}}}\right)$, and suggested KMM to adopt the setting as $\varepsilon = {}^{(\sqrt{n_{tr}} - 1)}/_{\sqrt{n_{tr}}}$.

The kernel and kernel parameters. KMM uses the kernel trick $k(x_i, x_j)$ to express the moment matching concept in the kernelized space. But the choices of kernel and the kernel parameters, which are the most important setting affecting the performance of the

KMM algorithm, have not been well-studied in the literature. To provide a representation of infinite moments in the kernelized space, a Gaussian kernel is the most often used kernel type. The bandwidth of a Gaussian kernel is still not easy to decide.

6.2.2 The Limitation of Objective Function MMD

Before we start to describe a method for tuning KMM, one direct question that might be asked: If the KMM algorithm takes MMD as its objective function, then why can't MMD serve as the criteria for parameter selection? We first answer this question and support our answer with an illustrative example.

Referring to Eq. 6.1, MMD also has its own parameters, related to the choice of kernel, to be determined, which shares the same parameters with KMM. For any defined kernel used to calculate the MMD, the KMM which minimizes the MMD will fall into the same kernel space and lead to the same choice. To demonstrate this difficulty, we use an illustrative example. Suppose that KMM is used to match two one-dimension Gaussians, where the distributions differ at the means from 0 to 1, and the variances are the same ($\sigma^2 = 1$). Fig. 6.1 shows the results of KMM and the corresponding MMD value with different kernel widths (X-axis). We can see that using different parameters for calculating MMD will also lead the KMM to arrive at an optimum at different bandwidths. The shared parameter and hence the dependency between MMD and KMM imply that the MMD (i.e., the objective function of KMM) is not suitable to act as an objective criterion for KMM to process model selection.

6.2.3 Tuning KMM Using NMSE

Inspired by the work of Kanamori et al. [69], we propose to introduce the Normalized Mean Squared Error (NMSE) to assess the goodness of parameter settings. The key idea is that while each KMM procedure minimizes the mean discrepancy between the weighted training samples and the test samples, an overhead parameter selection procedure minimizes the NMSE, which is computed from the matching results. A NMSE-based quality measure for estimating the goodness of candidate parameter values can be derived as follows.

The NMSE between the ground-truth and approximate density ratios is defined as:

$$\mathrm{NMSE} = \frac{1}{n_{tr}} \sum_{x \in \mathcal{X}_{tr}} \left(\frac{\tilde{\beta}(x)}{\sum_{z \in \mathcal{X}_{tr}} \tilde{\beta}(z)} - \frac{\beta(x)}{\sum_{z \in \mathcal{X}_{tr}} \beta(z)} \right)^2, \tag{6.5}$$

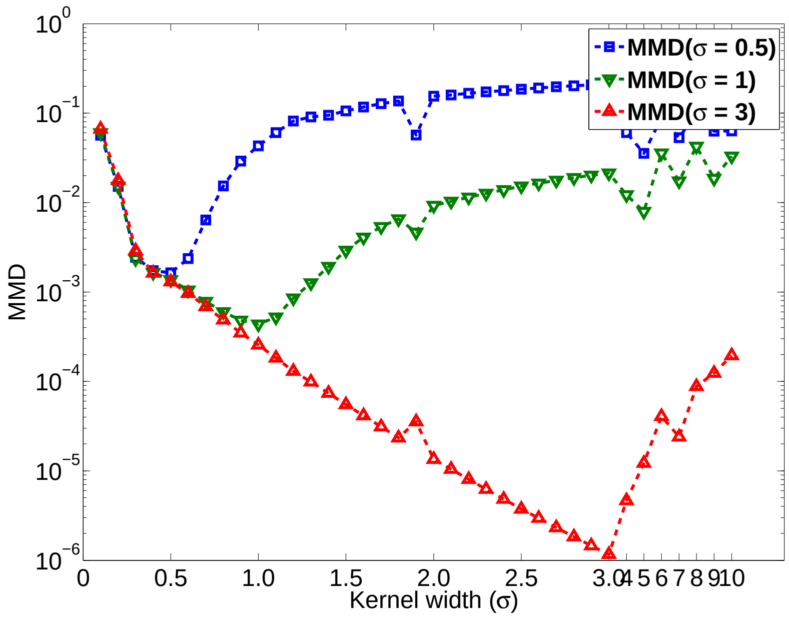


Figure 6.1: KMM with different kernel widths and the corresponding MMD values.

where $\beta(x)$ is the ground-truth density-ratio for a training sample x, $\tilde{\beta}(x)$ is the approximate density-ratio for x estimated using the KMM, $\mathcal{X}_{tr}$ is the set of training samples, and n_{tr} is the number of training samples.

Our goal is to define a criterion based on NMSE that can be used to assess the quality of a given parameter value, such that the best parameter value is the one that minimizes NMSE. Without loss of generality, we can assume that $\beta(x)$ and $\tilde{\beta}(x)$ are normalized over the training samples: $\sum_{x\in\mathcal{X}_{tr}}\tilde{\beta}(x) = \sum_{x\in\mathcal{X}_{tr}}\beta(x) = 1$. In this case, the NMSE can be calculated as:

$$\text{NMSE} = \frac{1}{n_{tr}}\sum_{x\in\mathcal{X}_{tr}}\left(\tilde{\beta}(x) - \beta(x)\right)^2 . \tag{6.6}$$

Eq. 6.6 is the empirical average corresponding to the following integral:

$$\begin{aligned} E[\text{NMSE}] &= \int\left(\tilde{\beta}(x) - \beta(x)\right)^2 p_{tr}(x)\,dx \\ &= \int\left(\tilde{\beta}^2(x) - 2\tilde{\beta}(x)\,\beta(x)\right)p_{tr}(x)\,dx + \int\beta^2(x)\,p_{tr}(x)dx\,. \end{aligned} \tag{6.7}$$

The term $\int\beta^2(x)\,p_{tr}(x)dx$ does not depend on the density-ratio estimation method and

Algorithm 10 Parameter Tuning for KMM

```
1: Input: $\mathcal{X}_{tr}$, $\mathcal{X}_{ts}$, $\mathcal{M}$ (a family of setups to be selected)
2: Output: chosen parameters $m^*$, and the estimates $\tilde{\boldsymbol{\beta}}^*$
3: for each $m$ in $\mathcal{M}$ do
4:     $\tilde{\boldsymbol{\beta}}_i^{(m)} \leftarrow \text{KMM}\left(\mathcal{X}_{tr}, \mathcal{X}_{ts}, m\right)$ for $x_i \in \mathcal{X}_{tr}$;
5:     Estimate $\tilde{\boldsymbol{\beta}}_j^{(m)}$ for $x_j \in \mathcal{X}_{ts}$ using RLS;
6:     $J(m) \leftarrow$ Eq. 6.10
7: end for
8: $m^* \leftarrow \arg\min_{m \in \mathcal{M}} J(m)$;
9: $\tilde{\boldsymbol{\beta}}^* \leftarrow \tilde{\boldsymbol{\beta}}^{(m*)}$.
```

accordingly the choice of the method parameters. This means that the parameter values that minimize NMSE will also lead to the minimization of a new score J, which can be defined as:

$$J = \int \left(\tilde{\beta}^2(x) - 2\tilde{\beta}(x)\,\beta(x)\right) p_{tr}(x)\,dx\ . \tag{6.8}$$

Substituting with $\beta(x) = p_{ts}(x)/p_{tr}(x)$ in Eq. 6.8, the J score can be simplified as follows:

$$J = \int \tilde{\beta}^2(x)\,p_{tr}(x)\,dx - 2\int \tilde{\beta}(x)\,p_{ts}(x)dx\ . \tag{6.9}$$

Using the empirical averages corresponding to the integrals, the right-hand of Eq. 6.9 can be expressed as:

$$J = \frac{1}{n_{tr}} \sum_{x \in \mathcal{X}_{tr}} \tilde{\beta}^2(x) - \frac{2}{n_{ts}} \sum_{x \in \mathcal{X}_{ts}} \tilde{\beta}(x)\ . \tag{6.10}$$

The first section of the J score can use the estimated $\tilde{\beta}$ at the training samples given by KMM directly. The second section considers the $\tilde{\beta}$ scores at the test samples, which are not available. We formulate this scenario as a regression problem to model the $\tilde{\beta}$ and then deduce values at the test samples. In this work, we use the Regularized Least Square (RLS) [109] as the regression method.

At this point, we have all the necessary components to calculate the J score of Eq. 6.10. The parameter tuning procedure of KMM is conducted by minimizing J, as summarized in Algorithm 10. This mechanism gives the KMM the ability to be tuned based on evaluating the goodness of density-ratio estimation from a different perspective, which minimizes the NMSE as the objective.

Table 6.1: Three cases of distribution shifting.

	p_{tr}	p_{ts}
Case-1	$\mathcal{N}(0, 1^2)$	$\mathcal{N}(1, 1^2)$
Case-2	$\mathcal{N}(0, \left(\frac{1}{2}\right)^2)$	$\mathcal{N}(0, \left(\frac{1}{4}\right)^2)$
Case-3	$\mathcal{N}(0, 1^2)$	$\mathcal{N}(1, \left(\frac{1}{2}\right)^2)$

It should be noted that using the estimation of $\tilde{\beta}$ at the training samples and extending the estimation to the test samples creates another regression model with covariate shift problem. However, we observed that since the second term (an average over $\tilde{\beta}$) is relatively small compared to the first term (an average over $\tilde{\beta}^2$), slight errors in estimating the second term due to covariate shift is not going to affect the values of the final quality measure. This is especially true when we have a relatively reasonable number of training samples, referring to Theorem 4 in [128].

6.3 Experiments

In this section, first the performance of the proposed method is demonstrated using three synthetic examples. Then, the method is evaluated on covariate shift tasks over benchmark datasets.

6.3.1 Synthetic Data

To demonstrate the ability of the proposed parameter tuning mechanism, we use one-dimension Gaussians and examine three distribution drifting cases as shown below. The ground truth density-ratio $\boldsymbol{\beta}$ can be obtained using the known underlying distribution functions. Therefore we can calculate the quality of the importance estimates $\tilde{\boldsymbol{\beta}}$ by the NMSE [109], as defined in Eq. 6.5.

Table 6.1 lists three cases to be examined where the training and test distributions are drifting either by a shifting of means, a shifting of variances, or both. In all three cases, 200 training samples and 1000 test samples are randomly generated from the distributions as given. The KMM using a Gaussian kernel was studied regarding different settings of kernel width.

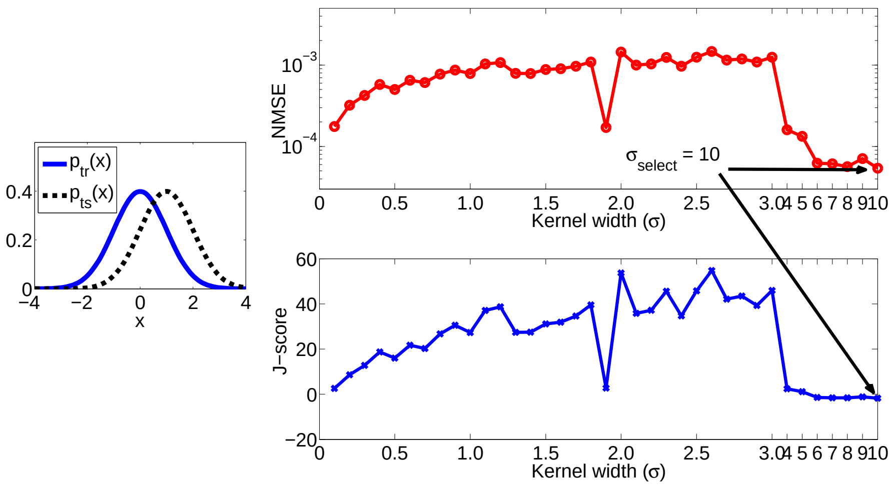


Figure 6.2: Case-1: the minimum J score leads to the choice of kernel width $\sigma = 10$.

The J scores calculated from the matching results and the NMSE calculated from ground-truth are plotted in Fig. 6.2, 6.3, and 6.4, corresponding to the three cases. From these results, some general facts can be observed:

1. The optimal parameter of KMM kernel width differs in different scenarios. The optimal parameters σ for the three cases are 10, 0.3 and 2.8, respectively.
2. This infers that any predefined value of the parameter may work in some cases, while failing in others. If we do not have a strong prior knowledge on the given tasks, an automatic parameter tuning method is greatly needed.
3. The J scores calculated from the outputs reflect the goodness of the KMM's matching results to a great extent, and usually lead to a proper choice of the parameters, even though it may not be the most optimal.

6.3.2 Covariate Shift of Benchmark Datasets

Further experiments have been conducted on ten benchmark datasets, whose properties are summarized in Table 6.2. These frequently used datasets are from the UCI[1] and LibSVM[2]

[1]UCI datasets: http://archive.ics.uci.edu/ml/datasets.html

[2]LibSVM datasets: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. For Cod-RNA and Adult-a1a, the first 3000 samples are taken.

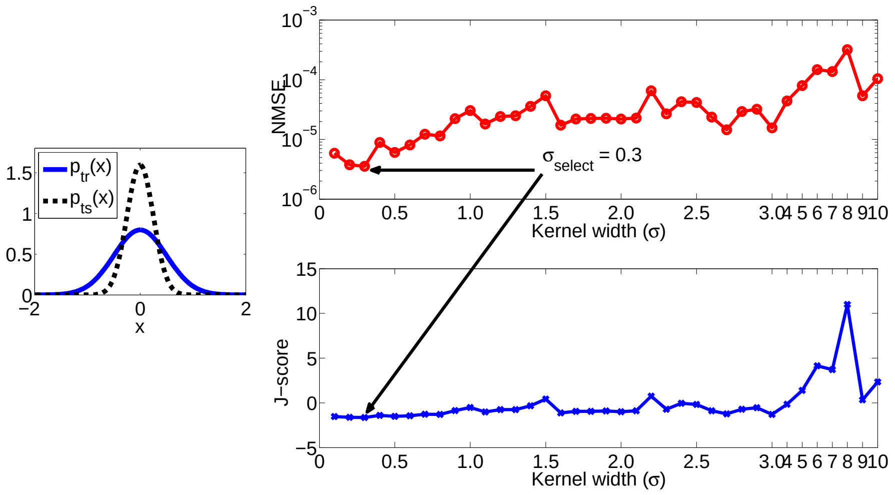


Figure 6.3: Case-2: the minimum J score leads to the choice of kernel width $\sigma = 0.3$.

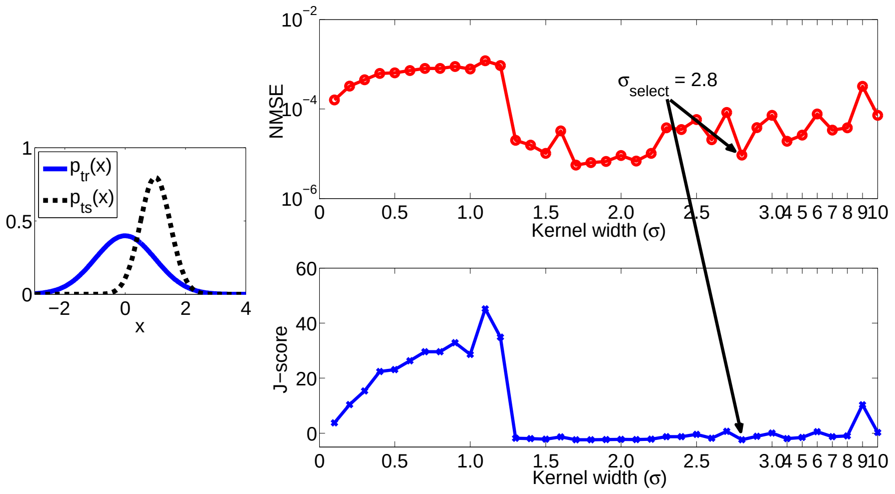


Figure 6.4: Case-3: the minimum J score leads to the choice of kernel width $\sigma = 2.8$.

Table 6.2: Overview of datasets and the training test split.

Dataset	**#Samples**	**#Features**	n_{tr}	n_{ts}
ImageSeg	2310	18	716	770
BreastCancer	683	9	283	228
Diabetes	768	8	266	256
PenDigits(6vs8)	1498	16	530	499
USPS(6vs8)	1508	256	483	503
GermanCredit	1000	24	334	333
Cod-RNA	3000	8	1015	1000
Splice	3175	60	1025	1058
Australian	690	14	235	230
Adult-a1a	3000	123	1009	1000

archives.

In our experiments, before any further processing, all the data were normalized to the range $[-1,1]^d$. The covariate shift classification tasks were formulated with the deliberately biased sampling procedures by following the work of [28]. First, one third of the data is uniformly sampled to form the test partition. Then, the rest of data is sub-sampled to form the the biased training set with probability $P(s=1|x) = {}^{e^v}\!/_{1+e^v}$, where $P(s=1)$ means that the sample x is included in the training set, and $v = {}^{4\boldsymbol{w}^{\boldsymbol{T}}(x-\overline{x})}\!/_{\sigma_{\boldsymbol{w}^{\boldsymbol{T}}(x-\overline{x})}}$. $\boldsymbol{w} \in \mathbb{R}^d$ is a projection vector randomly chosen from $[-1,1]^d$. For each run, we randomly generate ten values of $\boldsymbol{w}$ and select the $\boldsymbol{w}$ which maximizes the difference between the unweighted method and the weighted method with ideal sampling weights. The typical reserved number of training samples and the number of test samples are listed in Table 6.2.

The baseline method was set as fitting a model on the training set without any modifications and predicting the test samples. The following state-of-the-art sample importance estimation methods were included for comparison:

- **KDE:** Using Kernel Density Estimator [116] to estimate the training PDF and test PDF separately, then dividing the two densities.
- **KLIEP:** The Kullback-Leibler Importance Estimation Procedure [110], which models density-ratio as multi-Gaussians and minimizes the Kullback-Leibler divergence.
- **uLSIF:** unconstrained Least Squares Importance Fitting [69], which models density-ratio as multi-Gaussians and minimizes LSIF. The above three methods have out-

of-sample ability, and the parameters are chosen using the likelihood 10-fold Cross-Validation.

- **KMM(med):** KMM algorithm with the kernel width being set as the median of pairwise distances of all training and test samples.
- **KMM($\sqrt{d/2}$):** KMM algorithm with the kernel width being set as $\sqrt{d/2}$, where d is the number of features of the data. This setup was used by [28].
- **KMM(auto-tune):** The proposed method with Gaussian kernels, using NMSE to tune the parameter of kernel width, scanning kernel widths ($[0.1 : 0.1 : 3, 4 : 1 : 10] * \sigma_{med}$, where σ_{med} is the median of sample pairwise distances, and taking the choice with the minimal J score. In the above KMM methods, the other parameters are set to $\varepsilon = {}^{(\sqrt{n_{tr}}-1)}/_{\sqrt{n_{tr}}}$, $b = 1000$.

After acquiring the instance-dependent weights using the above importance estimation methods, we trained an importance-weighted Least-Squares Probabilistic Classifier (iwLSPC) [109] and evaluated its prediction accuracy on the test sets respectively. Each setup was repeated 30 times and the average performance measures are reported.

6.3.3 Results and Discussion

Similar to previous work [110], the Normalized Error (NE) is adopted to show the effectiveness of a method by considering the error of the baseline unweighted method as one and calculate the metric as:

$$\mathrm{NE}_{method} = \frac{\mathrm{Err}_{method}}{\mathrm{Err}_{baseline}} \times 100\% \tag{6.11}$$

Table 6.3 reports the Normalized Error on the ten datasets by using different importance estimation methods. The significance of improvement was tested using the Friedman test [33]. It shows that the KMM equipped with the proposed auto-tuning mechanism outperforms other methods in mostly all the cases. As mentioned earlier, the KDE method is a two-step approach of importance estimation, which has an inherent likelihood Cross-Validation mechanism for parameter selection. In low dimensionality cases, it performs well. But, when encountering a high-dimensional problem, the weakness of this method is noticeable. On the other hand, the conventional heuristic setups of KMM, which uses the median of sample pairwise distances or $\sqrt{d/2}$, do not have strong evidence of effectiveness. This is consistent with the findings of other reported results [28, 53, 108].

Table 6.3: The normalized testing error of different importance estimation methods. For each dataset, the best performing group of methods (according to the Friedman test at a confidence interval of 95%) are highlighted in bold. The second-best method is underlined. The absolute error rate of the baseline is also reported in the first column.

	Baseline		**KDE**	**KLIEP**	**uLSIF**	**KMM**	**KMM**	**KMM**
Dataset	**Err_abs**	**Err_norm**				**(med)**	**($\sqrt{d/2}$)**	**(tuned)**
ImageSeg	0.1869	100.00	<u>59.03</u>	61.44	66.63	62.44	62.99	**55.70**
BreastCancer	0.3115	100.00	109.57	42.56	70.44	21.59	**21.21**	<u>21.49</u>
Diabetes	0.3469	100.00	98.24	95.46	<u>95.42</u>	98.39	99.36	**95.38**
PenDigits(6vs8)	0.0140	100.00	22.86	22.38	<u>**21.90**</u>	31.90	26.19	**21.43**
USPS(6vs8)	0.1262	100.00	81.73	<u>25.98</u>	36.90	30.50	30.34	**25.30**
GermanCredit	0.3160	**100.00**	106.37	103.99	<u>103.64</u>	105.16	107.92	104.91
Cod-RNA	0.3316	100.00	<u>85.81</u>	89.03	86.65	**86.16**	87.52	**83.77**
Splice	0.3792	100.00	125.23	90.95	113.93	85.30	**81.84**	<u>83.60</u>
Australian	0.2354	100.00	93.90	**72.60**	79.86	84.73	86.45	<u>79.68</u>
Adult-a1a	0.2640	100.00	118.71	<u>94.38</u>	100.37	**90.49**	96.94	96.72
Average		100.00	90.14	69.88	77.58	<u>69.67</u>	70.08	**66.80**

An interesting observation can be noted for the German Credit dataset. In this case, there is no improvement in classification performance when comparing all importance weighting methods with the simple unweighted approach. In such a scenario, the distribution changes are probably far away from the decision boundaries. And any reweighting strategy will not be effective in dealing with the shift.

6.4 Extension to Other Kernels

In this section, we extend the proposed auto-tuning method to another type of kernels, the polynomial kernels. We observed that when using the polynomial kernel with classification problems (in a setup as the one explored in Section 6.3.2), the classification accuracy does not change that much with different parameter values. This however does not reduce the usefulness of applying the auto-tuning method to polynomial kernels. This is because KMM is essentially a method for importance estimation which could have applicability in other machine learning tasks such as anomaly detection. Based on this observation, we use a different approach to evaluate how the auto-tuning method works with polynomial kernels. Specifically, we used the NMSE to evaluate how good the estimated density-ratio is using different parameter values in comparison to using the auto-tuning method.

The following two commonly used polynomial kernels are to be investigated:

1. The polynomial kernel of degree 2: $k(x_i, x_j) = (x_i^T x_j + c)^2$;
2. The polynomial kernel of degree 3: $k(x_i, x_j) = (x_i^T x_j + c)^3$.

The parameter c in these kernels is to be tuned using the proposed method.

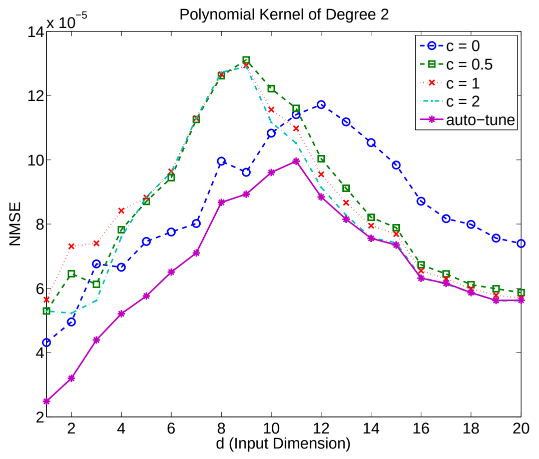
Taking the same setup as [110], experiments were conducted based on the setup of Case-1 listed in Table 6.1. We fixed the number of test samples as $n_{ts} = 1000$, and considered the following two scenarios:

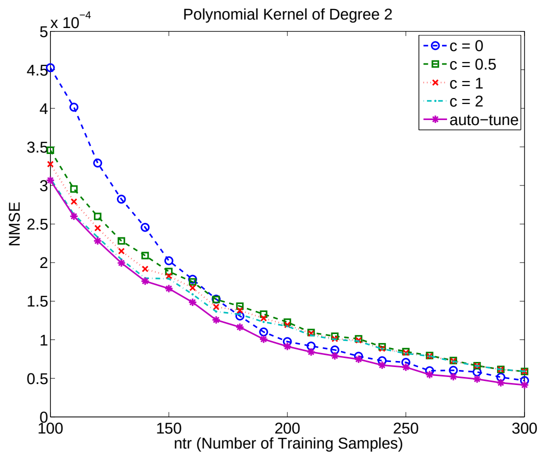
1. Fix the number of training samples as $n_{tr} = 200$, and change the input dimension as $d = 1, 2, ..., 20$;
2. Fix the input dimension as $d = 10$, and increase the training sample size as $n_{tr} = 100 : 10 : 300$.

For each setting, the experiments were repeated 100 times. The matching quality was evaluated by the normalized mean square error (Eq. 6.5). For the parameter tuning method, the c is automatically chosen from the range of $[0, 2]$ with increments of 0.1.

Fig. 6.5 and 6.6 show the average NMSE when using different parameter values for polynomial kernels of degree 2 and 3, respectively. From Fig. 6.5a, we can find that small values of the parameter c gives better matching scores for lower values of d. On the contrary, for high dimensional cases it can be observed that large values of c tend to produce better matching results. The degree 3 kernel (Fig. 6.6a) shows its own characteristics in response to dimension changes, but the effects of parameter c demonstrate the same trend. From these observations, we can conclude that using a fixed value for the c parameter is not going to give the best results for all dimensions, while using the auto-tuning method achieves the best matching results for all dimensions and outperforms the performance of the fixed values.

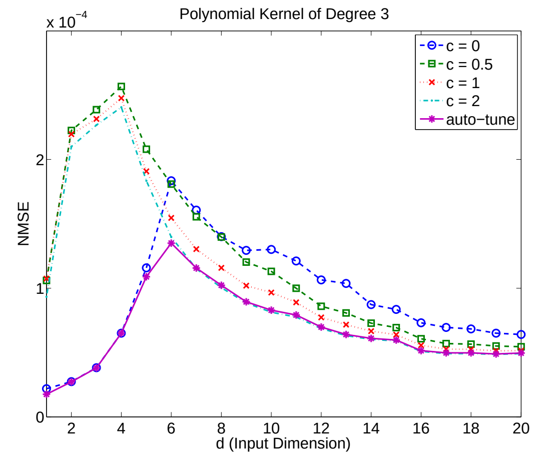
As observed from Fig. 6.5b and 6.6b, it is not a surprise that in general the matching errors are shrinking as the training sample size increases. For the degree 2 kernel, the interesting point is that large values of c seem to perform better when there is a small number of training samples. On the other hand, small values of c are more suitable for large numbers of training samples. For the degree 3 kernel, large values of c tend to perform well

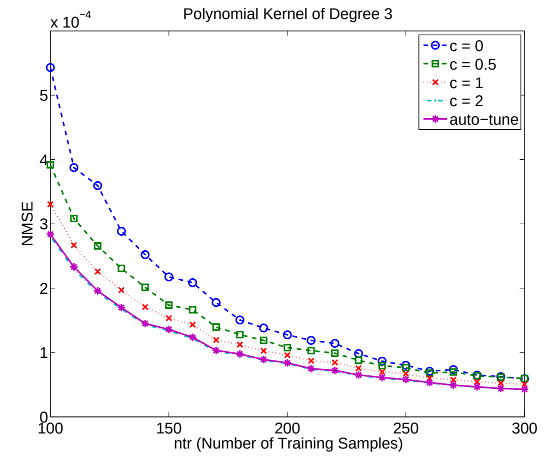




(a) Changing the input dimension from 1 to 20. (b) Changing the training sample size from 100 to 300.

Figure 6.5: The polynomial kernel of degree 2.





(a) Changing the input dimension from 1 to 20. (b) Changing the training sample size from 100 to 300.

Figure 6.6: The polynomial kernel of degree 3.

with different number of training samples. Similarly, we can observe that using fixed values for the parameter c does not always achieve the best matching performance while using the auto-tuning method achieves the best matching scores for different sizes of training samples.

Overall, the parameter tuning method was consistently found to be effective in all the circumstances studied.

Chapter 7

Conclusions and Future Work

This chapter concludes this dissertation and outlines future work. Section 7.1 presents a summary of contributions and concluding remarks. Section 7.2 discusses potential directions for extending the current study.

7.1 Conclusions

This dissertation analyzed the non-stationary data mining problem and proposed a set of methods for reweighting the training data. The proposed algorithms advance the distribution matching scheme by developing novel density-ratio methods to solve different non-stationary data analysis tasks differently.

For domain adaptation classification problems, this dissertation first presented a novel Discriminative Density-Ratio (DDR) method for learning adaptive classifiers. To minimize the discrepancy between training and test data, many methods are based on estimating density ratios over the marginal distributions and apply to both regression and classification problems. Although these methods work well for regression problems, their performance on classification problems is not satisfactory. This is due to a key observation that these methods focus on matching the sample marginal distributions without paying attention to preserving the separation between classes in the reweighted space. The proposed DDR method addressed the problem by estimating the density ratio of joint distributions in a class-wise manner.

In non-stationary environments, along with the changing of existing concepts which are handled by the domain adaptation techniques, we are also facing the occurrence of new concepts. The detection of novelties is another crucial problem in non-stationary data mining. In this dissertation, a locally-adaptive kernel density-ratio method has been proposed that captures both the emerging and evolving aspects of novelties. Traditional algorithms are limited to detect either emerging novel instances which are completely new, or evolving novel instances whose distribution are different from previously-seen ones. The proposed algorithm builds on the success of the idea of using density ratio as a measure of evolving novelty and augments this with structural information about each data instance's neighborhood. This makes the estimation of density ratio more reliable, and results in detection of emerging as well as evolving novelties.

In addition, the proposed locally-adaptive kernel novelty detection method was applied in the social media analysis for novel concepts detection. Social media data, such as the messages in Twitter, is characterized by extreme sparsity and high dimensionality if using traditional document-term vector space representation. To overcome the difficulty, the low-rank approximation based on term-term semantic kernels was adopted in the proposed solution. The low dimensional semantic kernel representation embeds the most important concepts in the short text. As the time continuity of social media streams, the novelty is usually characterized by the combination of emerging and evolving novelties. One reason is the existence of large common vocabularies between different topics. Another reason is that there is a high possibility of topics being continuously discussed in sequential batches of collections, but showing different levels of intensity. Thus, the presented novelty detection algorithm demonstrated its effectiveness in the social media data analysis application.

Although the work on estimating density ratios in a discriminative manner considerably enhances the performance of density-ratio estimators, the category of kernel mean matching methods depend on the choice of a kernel whose parameters are hard to estimate. In order to address this problem, we proposed an auto-tuning method for kernel mean matching by introducing a novel measure for assessing the quality of candidate choices. The proposed quality measure reflects the normalized mean square error between the estimated importance weights and the ratio of the estimated test and training densities, which is a different perspective from the objective function used by the kernel mean matching algorithm. The proposed auto-tuning method does not depend on the learner in the following step and accordingly allows the model selection procedures for importance estimation and prediction model learning to be completely separated.

Non-stationary data is pervasive in a large number of data mining applications and poses great challenges for many traditional learning algorithms. This dissertation focused on the sample reweighting approach and developed a set of strategies that effectively handle dynamic data analysis tasks.

7.2 Future Work

The work presented in this research can be extended in different directions and applied in many interesting applications. A short list of possible directions are highlighted as follows:

Scalable adaptation algorithms for big data platform. Large scale data is a dominant characteristic in many of todays applications. We have seen platforms being proposed to handle large scale data by using distributed storage and processing the data among cluster of machines, such as the MapReduce [31] and Spark [130] framework. The presented methods in this dissertation are based on single machine execution. How to develop optimized distributed extensions to these methods is worth further investigation.

Efficient adaptation algorithms for resource limited environment. Another promising direction is to develop adaptation algorithms that can continually adapt to new data while reducing both time and space complexities. This has important applications in the resource limited environment. For example, intelligent sensor and sensor networks are imposed with many constraints, including limited accessibility of computation, storage, and power.

Personalized data mining to achieve both efficiency and effectiveness. Most of the algorithms for supervised and unsupervised learning are in direct interaction with thousands or even millions of system users. The variety of users and their needs generate a challenging aspect of non-stationarity in the data. The new data associated with a new user or group of users can be used to adapt the learning models to perform well on these data. This leads to personalized systems with objectives to achieve both efficiency and effectiveness by selecting and modeling a small portion of data from the whole data space. The personalized data mining techniques can be explored in interesting applications such as intelligent personal assistants and personalized recommendations.

Identification and modeling of emerging and volatilizing topics in social media. As an application case, we have presented solutions for novelty detection. The ability of tracking both emerging and volatilizing topics in a social media stream is needed, which can help us understanding the full view of social events.

7.3 Publications

From the research the following publications are produced.

- Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel (2015). "Ensemble Kernel Mean Matching". *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*: 15 pages. Submitted in April 2015.

- Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel (2015). "Locally Adaptive Density Ratio for Detecting Novelty in Twitter Streams". *To Appear in Proceedings of the 6th International Workshop on Modeling Social Media (MSM) - Behavioral Analytics in Social Media, Big Data and the Web*: 6 pages. 2015.

- Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel (2014). "A Novel Approach For Domain Adaptive Classification". *Machine Learning Journal (Springer)*: 27 pages. Submitted in November 2014.

- Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel (2014). "Discriminative Density-ratio Estimation". *In Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, pages 830-838, 2014.

- Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel (2014). "Class-wise Density-ratios for Covariate Shift". *The Workshop on New Directions in Transfer and Multi-Task at the Neural Information Processing Systems (NIPS)*: 5 pages. - *Best Abstract Award.*

- Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel (2013). "Auto-tuning Kernel Mean Matching". *In Proceedings of the Workshop on Incremental Clustering, Concept Drift and Novelty Detection at the IEEE 13th International Conference on Data Mining (ICDM)*, pages 560-567, 2013.

- Yun-Qian Miao, Rodrigo Araujo, and Mohamed S. Kamel (2012). "Cross-Domain Facial Expression Recognition Using Supervised Kernel Mean Matching". *In Proceedings of the IEEE 11th International Conference on Machine Learning and Applications (ICMLA)*, pages 326-332, 2012.

Bibliography

[1] Charu C. Aggarwal and Philip S. Yu. Outlier detection with uncertain data. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 483–493. SIAM, 2008.

[2] Ferit Akova, Murat Dundar, V. Jo Davisson, E. Daniel Hirleman, Arun K. Bhunia, J. Paul Robinson, and Bartek Rajwa. A machine-learning approach to detecting unknown bacterial serovars. *Statistical Analysis and Data Mining*, 3(5):289–301, 2010.

[3] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[4] Andrew G. Barto. *Reinforcement learning: An introduction.* MIT press, 1998.

[5] Safaa M. Bedawi and Mohamed S. Kamel. Urban land-cover classification based on swarm intelligence from high resolution remote sensing imagery. In *Proceedings of the 2011 International Conference on Remote Sensing, Environment and Transportation Engineering (RSETE)*, pages 5617–5620. IEEE, 2011.

[6] Richard Bellman. *Introduction to matrix analysis*, volume 960. SIAM, 1970.

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

[8] Shai Ben-David and Ruth Urner. On the hardness of domain adaptation and the utility of unlabeled target samples. In *Algorithmic Learning Theory*, pages 139–153, 2012.

[9] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 120–128, 2006.

[10] R.P. Jagadeesh Chandra Bose, Wil M.P. van der Aalst, Indrė Žliobaitė, and Mykola Pechenizkiy. Handling concept drift in process mining. In *Advanced Information Systems Engineering*, pages 391–405. Springer, 2011.

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

[12] 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.

[13] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, pages 93–104. ACM, 2000.

[14] Serhat S. Bucak and Bilge Gunsel. Incremental subspace learning via non-negative matrix factorization. *Pattern Recognition*, 42(5):788–797, 2009.

[15] Xavier Carreras, Llu S. Marquez, and Jordi Girona Salgado. Boosting trees for anti-spam email filtering. In *Proceedings of the 4th International Conference on Recent Advances in Natural Language Processing (RANLP)*, 2001.

[16] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 554–560. ACM, 2006.

[17] Sanjay Chakraborty and N.K. Nagwani. Analysis and study of incremental k-means clustering algorithm. In *High Performance Architecture and Grid Computing*, pages 338–341. Springer, 2011.

[18] Yee Seng Chan and Hwee Tou Ng. Word sense disambiguation with distribution estimation. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1010–1015, 2005.

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[20] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 24(5):823–839, 2012.

[21] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[22] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-supervised learning*, volume 2. MIT press, Cambridge, 2006.

[23] Olivier Chapelle, Vikas Sindhwani, and Sathiya S. Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.

[24] Minmin Chen, Kilian Q. Weinberger, and John Blitzer. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 2456–2464, 2011.

[25] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 767–774, 2012.

[26] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven A. Klooster. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 413–424. SIAM, 2009.

[27] Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014.

[28] Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. Sample selection bias correction theory. In *Algorithmic Learning Theory*, pages 38–53, 2008.

[29] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 193–200, 2007.

[30] Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126, 2006.

[31] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[32] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[33] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[34] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report TR-04-25, University of Texas at Austin, 2005.

[35] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.

[36] Thomas G. Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.

[37] Gregory Ditzler and Robi Polikar. An ensemble based incremental learning framework for concept drift and class imbalance. In *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[38] Fatemeh Dorri and Ali Ghodsi. Adapting component analysis. In *Proceedings of the IEEE 12th International Conference on Data Mining (ICDM)*, pages 846–851. IEEE Computer Society, 2012.

[39] Ran El-Yaniv and Oren Souroujon. Iterative double clustering for unsupervised and semi-supervised learning. In *Machine Learning: ECML 2001*, pages 121–132. Springer, 2001.

[40] Charles Elkan. Preserving privacy in data mining via importance weighting. In *Privacy and Security Issues in Data Mining and Machine Learning*, pages 15–21. Springer, 2011.

[41] Meng Fang, Jie Yin, and Xingquan Zhu. Transfer learning across networks for collective classification. In *Proceedings of the IEEE 13th International Conference on Data Mining (ICDM)*, pages 161–170, 2013.

[42] Ahmed K. Farahat, Ali Ghodsi, and Mohamed S. Kamel. A novel greedy algorithm for Nyström approximation. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 269–277, 2011.

[43] Ahmed K. Farahat, Ali Ghodsi, and Mohamed S. Kamel. Efficient greedy feature selection for unsupervised learning. *Knowledge and Information Systems*, 35(2):285–310, 2013.

[44] Ahmed K. Farahat and Mohamed S. Kamel. Statistical semantics for enhancing document clustering. *Knowledge and Information Systems*, 28(2):365–393, 2011.

[45] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[46] Dimitar P. Filev and Finn Tseng. Novelty detection based machine health prognostics. In *Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems*, pages 193–199. IEEE, 2006.

[47] George S. Fishman. *Monte Carlo: concepts, algorithms, and applications*. Springer, 1996.

[48] Joao Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, March 2014.

[49] Giorgio Giacinto, Roberto Perdisci, Mauro Del Rio, and Fabio Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, 9(1):69–82, 2008.

[50] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 513–520, 2011.

[51] Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 352–359. ACM, 2008.

[52] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel method for the two sample problem. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 513–520, Cambridge, MA, 2007. MIT press.

[53] 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*, pages 131–160. MIT press, Cambridge, MA, 2009.

[54] Hirotaka Hachiya, Masashi Sugiyama, and Naonori Ueda. Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomputing*, 80:93–101, 2012.

[55] Jiawei Han and Jing Gao. Research challenges for data mining in science and engineering. In *Next Generation of Data Mining*, chapter 1, pages 3–28. Chapman & Hall, 2009.

[56] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques (3rd edition)*. Morgan Kaufmann, 2011.

[57] Wolfgang Härdle, Axel Werwatz, Marlene Müller, and Stefan Sperlich. Introduction. *Nonparametric and semiparametric models*, pages 1–18, 2004.

[58] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer Series in Statistics, second edition, 2009.

[59] 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.

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

[61] Timothy M. Hospedales, Shaogang Gong, and Tao Xiang. Finding rare classes: Active learning with generative and discriminative models. *Knowledge and Data Engineering, IEEE Transactions on*, 25(2):374–386, 2013.

[62] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 601–608, 2007.

[63] Norden E. Huang, Man-Li Wu, Wendong Qu, Steven R. Long, and Samuel S.P. Shen. Applications of Hilbert-Huang transform to non-stationary financial time series analysis. *Applied Stochastic Models in Business and Industry*, 19(3):245–268, 2003.

[64] Jean Jacod. Multivariate point processes: predictable projection, Radon-Nikodym derivatives, representation of martingales. *Probability Theory and Related Fields*, 31(3):235–253, 1975.

[65] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[66] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002.

[67] Jing Jiang. A literature survey on domain adaptation of statistical classifiers. online, `http://sifaka.cs.uiuc.edu/jiang4/domain_adaptation/survey/da_survey.pdf`, Last modified by author in March 2008. Retrieved in January 2015.

[68] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, volume 99, pages 200–209, 1999.

[69] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. Efficient direct density ratio estimation for non-stationarity adaptation and outlier detection. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 809–816, 2008.

[70] Yoshinobu Kawahara and Masashi Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of 2009 SIAM International Conference on Data Mining (SDM)*, pages 389–400, 2009.

[71] Masanori Kawakita and Takafumi Kanamori. Semi-supervised learning with density-ratio estimation. *Machine Learning*, 91(2):189–209, 2013.

[72] Michael Kemmler, Erik Rodner, Esther-Sabrina Wacker, and Joachim Denzler. One-class classification with gaussian processes. *Pattern Recognition*, 46(12):3507–3518, 2013.

[73] Gwangseob Kim and Ana P. Barros. Space-time characterization of soil moisture from passive microwave remotely sensed imagery and ancillary data. *Remote Sensing of the Environment*, 81(2):393–403, 2002.

[74] J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[75] Ludmila I. Kuncheva. Classifier ensembles for changing environments. In *Multiple Classifier Systems*, pages 1–15. Springer, 2004.

[76] Rocco Langone, Carlos Alzate, and Johan A.K. Suykens. Kernel spectral clustering with memory effect. *Physica A: Statistical Mechanics and its Applications*, 392(10):2588–2606, 2013.

[77] Rocco Langone, Oscar Mauricio Agudelo, Bart De Moor, and Johan A.K. Suykens. Incremental kernel spectral clustering for online learning of non-stationary data. *Neurocomputing*, 139:246–260, 2014.

[78] Jaeshin Lee, Bokyoung Kang, and Suk-Ho Kang. Integrating independent component analysis and local outlier factor for plant-wide process monitoring. *Journal of Process Control*, 21(7):1011–1021, 2011.

[79] Yu-Feng Li, James T. Kwok, and Zhi-Hua Zhou. Cost-sensitive semi-supervised support vector machine. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, volume 10, pages 500–505, 2010.

[80] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[81] Victoria López, Alberto Fernández, and Francisco Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1–13, 2014.

[82] Zhi-Quan Luo and Paul Tseng. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46(1):157–178, 1993.

[83] Heikki Mannila. Data mining: machine learning, statistics, and databases. In *International Conference on Scientific and Statistical Database Management*, pages 2–9. IEEE Computer Society, 1996.

[84] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Risk minimization, probability elicitation, and cost-sensitive svms. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 759–766, 2010.

[85] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Classification and novel class detection in concept-drifting data streams

under time constraints. *Knowledge and Data Engineering, IEEE Transactions on*, 23(6):859–874, 2011.

[86] Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel. Auto-tuning kernel mean matching. In *Workshop on Incremental Clustering, Concept Drift and Novelty Detection at the IEEE 13th International Conference on Data Mining (ICDM)*, pages 560–567, 2013.

[87] Yun-Qian Miao, Ahmed K. Farahat, and Mohamed S. Kamel. Discriminative density-ratio estimation. In *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, pages 830–838, 2014.

[88] Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *The Journal of Machine Learning Research*, 7:2651–2667, 2006.

[89] Stuart Middleton, Lee Middleton, and Stefano Modafferi. Real-time crisis mapping of natural disasters using social media. *Intelligent Systems, IEEE*, 29(2):9–17, Mar 2014.

[90] Leandro L. Minku and Xin Yao. DDD: A new ensemble approach for dealing with concept drift. *Knowledge and Data Engineering, IEEE Transactions on*, 24(4):619–633, 2012.

[91] Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.

[92] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, 22(2):199–210, 2011.

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

[94] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[95] Oxford University Press. Rt this: OUP dictionary team monitors twitterer's tweets. `http://blog.oup.com/2009/06/oxford-twitter/`, Retrieved in January 2015.

[96] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[97] Douglas A. Reynolds and Richard C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *Speech and Audio Processing, IEEE Transactions on*, 3(1):72–83, 1995.

[98] Ankan Saha and Vikas Sindhwani. Learning evolving and emerging topics in social media: a dynamic NMF approach with temporal regularization. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 693–702. ACM, 2012.

[99] Peter Sarlin. Self-organizing time map: an abstraction of temporal multivariate patterns. *Neurocomputing*, 99:496–508, 2013.

[100] Bernhard Schölkopf, Robert C. Williamson, Alex J. Smola, John Shawe-Taylor, and John C. Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, pages 582–588, 1999.

[101] David W Scott. *Multivariate density estimation: theory, practice, and visualization*, volume 383. John Wiley & Sons, 2009.

[102] Burr Settles. Active learning literature survey. Technical Report 1648, Computer Sciences, University of Wisconsin, Madison, 2010.

[103] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 465–472, 2004.

[104] Yuan Shi and Fei Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1079–1086, 2012.

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

[106] Alexander J. Smola, Le Song, and Choon Hui Teo. Relative novelty detection. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 536–543, 2009.

[107] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, pages 187–198, 2006.

[108] Masashi Sugiyama, Takafumi Kanamori, Taiji Suzuki, Shohei Hido, Jun Sese, Ichiro Takeuchi, and Liwei Wang. A density-ratio framework for statistical data processing. *IPSJ Transactions on Computer Vision and Applications*, 1(0):183–208, 2009.

[109] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *The Journal of Machine Learning Research*, 8:985–1005, 2007.

[110] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 1433–1440, 2008.

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

[112] Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[113] Taiji Suzuki and Masashi Sugiyama. Sufficient dimension reduction via squared-loss mutual information estimation. *Neural Computation*, 25(3):725–758, 2013.

[114] Yimin Tan and Xiaojin Zhu. Dragging: Density-ratio bagging. Technical Report TR1795, Computer Science, University of Wisconsin-Madison, 2013.

[115] David M.J. Tax and Robert P.W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[116] George R. Terrell and David W. Scott. Variable kernel density estimation. *The Annals of Statistics*, 20(3):1236–1265, 1992.

[117] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[118] Alexey Tsymbal. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland, 2004.

[119] Vladimir N. Vapnik. *Statistical learning theory.* Wiley, 1998.

[120] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235. ACM, 2003.

[121] Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[122] Felix Weninger, Martin Wollmer, Jurgen Geiger, Bjorn Schuller, Jort F. Gemmeke, Antti Hurmalainen, Tuomas Virtanen, and Gerhard Rigoll. Non-negative matrix factorization for highly noise-robust ASR: to enhance or to recognize? In *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4681–4684. IEEE, 2012.

[123] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[124] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *Knowledge and Data Engineering, IEEE Transactions on*, 26(1):97–107, 2014.

[125] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

[126] Liu Yang, Steve Hanneke, and Jaime Carbonell. A theory of transfer learning with applications to active learning. *Machine Learning*, 90(2):161–189, 2013.

[127] Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604, 2006.

[128] Yaoliang Yu and Csaba Szepesvári. Analysis of kernel mean matching under covariate shift. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 607–614, 2012.

[129] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the IEEE 3rd International Conference on Data Mining (ICDM)*, 2003.

[130] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, pages 10–10, 2010.

[131] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1601–1608, 2004.

[132] Chunlin Zhang, Ju Jiang, and Mohamed Kamel. Intrusion detection using hierarchical neural networks. *Pattern Recognition Letters*, 26(6):779–791, 2005.

[133] Kun Zhang, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 819–827, 2013.

[134] Xiaojin Zhu and Andrew B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.

[135] Indrė Žliobaitė, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, pages 1–28, 2014.

[136] Arkaitz Zubiaga and Heng Ji. Harnessing web page directories for large-scale classification of tweets. In *Proceedings of the 22nd International Conference on World Wide Web Companion*, pages 225–226, 2013.