Design and Analysis of Techniques for Multiple-Instance Learning in the Presence of Balanced and Skewed Class Distributions

by

Xiaoguang Wang

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

With the continuous expansion of data availability in many large-scale, complex, and networked systems, such as surveillance, security, the Internet, and finance, it becomes critical to advance the fundamental understanding of knowledge discovery and analysis from raw data to support decision-making processes. Existing knowledge discovery and data analyzing techniques have shown great success in many real-world applications such as applying Automatic Target Recognition (ATR) methods to detect targets of interest in imagery, drug activity prediction, computer vision recognition, and so on. Among these techniques, Multiple-Instance (MI) learning is different from standard classification since it uses a set of bags containing many instances as input. The instances in each bag are not labeled — instead the bags themselves are labeled. In this area many researchers have accomplished a lot of work and made a lot of progress. However, there still exist some areas which are not covered. In this thesis, we focus on two topics of MI learning: (1) Investigating the relationship between MI learning and other multiple pattern learning methods, which include multi-view learning, data fusion method and multi-kernel SVM. (2) Dealing with the class imbalance problem of MI learning.

In the first topic, three different learning frameworks will be presented for general MI learning. The first uses multiple view approaches to deal with MI problem, the second is a data fusion framework, and the third framework, which is an extension of the first framework, uses multi-kernel SVM. Experimental results show that the approaches presented work well on solving MI problem.

The second topic is concerned with the imbalanced MI problem. Here we investigate the performance of learning algorithms in the presence of underrepresented data and severe class distribution skews. For this problem, we propose three solution frameworks: a data re-sampling framework, a cost-sensitive boosting framework and an adaptive instance-weighted boosting SVM (with the name IB_SVM) for MI learning. Experimental results — on both benchmark datasets and application datasets — show that the proposed frameworks are proved to be effective solutions for the imbalanced problem of MI learning.

Acknowledgements

I would like to extend my gratitude to the many people who helped to bring this research project to fruition.

First, I would like to thank my advisors — Professor Nathalie Japkowicz and Professor Stan Matwin for providing me the opportunity to do research on machine learning-my favourite research area. I am so deeply grateful for their professionalism, valuable guidance and financial support throughout my entire program of study. I do not have enough words to express my deep and sincere appreciation. I learned much from our meetings and casual discussions. Along the way, they gave me many suggestions on my thesis and helped me finish it.

I would like to thank Professor John Oommen, Dr. Chris Drummond, Professor Haibo He and Professor Herna Viktor, for reading my thesis and providing valuable feedback, suggestions and discussions.

I thank Jack Xu for his hard work of helping me edit this thesis.

Many thanks to the Text Analysis and Machine Learning Group (TAMALE) seminar in university of Ottawa. I have learned a lot and got many good ideas in this seminar.

Finally, I must thank my wife, my parents and my other family members. With their support, I can do my research with all my heart. I also thank all my friends who always support me and help me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the information age in which we live, accumulating data is easy and storing is inexpensive. People recognize that learning from data can gain new knowledge from human "experiences". Machine learning, the study of algorithms that automatically learns from data, improves our ability to perform real-world tasks.

In machine learning, supervised learning is concerned with algorithms that can learn concepts from labeled examples. If each instance is labeled, this kind of supervised learning is named as Single-Instance learning. In contrast, there is an alternative type of learning referred to as Multiple Instance (MI) learning [38]. In this learning, each individual pattern in a data set can be described by a set of individual instances. The instances in this set are not labeled but the set itself is. For MI data set, each set is called a "bag" and it can contain a number of instances. The number of instances in each bag can be different, and the same instance can even belong to different bags. Bags may group different instances sharing the same property. Bags may also group different instances of the same object. The MI problem first emerged in the work of Dietterich et al. [38] when addressing the problem of drug activity prediction. Since then MI learning has been used in many applications, including drug activity recognition [38], computer vision recognition [68][122] and text-categorization [4].

MI learning has several key advantages. Firstly, recent research [34][43][121] supports that it is possible to obtain more accurate classifications if the detection is based upon multiple instances of the object instead of a single instance of the object. Here each instance can be seen as an "aspect". Since MI algorithms can combine the information of multiple aspects of the object for recognition, the MI framework has been shown to be effective in such applications. Secondly, for supervised learning, data is labeled by experts and the labeling work is very time-consuming. For MI data, only the bags need to be labeled, which will highly reduce the labeling workload.

1.1 Motivation and Objectives

Side-scan sonar equipped vehicles, such as Autonomous Underwater Vehicles (AUVs), are frequently used by military and commercial organizations to acquire high-resolution sonar imagery for the detection of possible objects (e.g. mine like objects (MLOs)) or regions of interest. Automatic Target Recognition (ATR) methods have been successfully applied to detect MLOs in sonar imagery [43][77][83][84]. Since many sonar images of the same object are taken during different sonar passes, there are multiple images of the same object at different ranges and aspects. It is expected that the additional information obtained from more images of an object will improve the classification performance in the case of single image information. Experimental results from recent research prove this [43][77][83][84], and also find that misclassifications can be reduced if the detection is based on multiple images of an object, rather than a single image.

When using existing algorithms [43][77][83][84] for MLOs detection in our related project, performance (on both efficacy and efficiency) of these approaches is not always as good as we expect. We were motivated to develop new algorithms for this task. Based on the characters of the MLOs datasets in our project, MI learning, which has been successfully applied in many computer vision recognition applications, has been chosen as an Automatic Target Recognition (ATR) candidate method to detect MLOs. This was the original motivation of this research.

However, when using existing MI learning algorithms for some MLOs datasets, performance is still not always as good as we expect. We must look for a better performance algorithm for our tasks. For existing MI learning approaches, they can be categorized into "one-level" methods and "two-level" methods. The term "one-level" denotes the methods that extend the single-instance learners to deal with MI data. Different from "one-level" methods, "two-level" MI learning methods tackle the MI problem by introducing two levels of data space: the instance-level and the bag-level. Most existing MI methods are "one-level" methods. In our preliminary research, we have found that the "two-level" MI learning framework may provide better compatitive preformance and learning efficiency comparing to many "one-level" MI learning methods. To develop new MI algorithms which have good performance and efficiency, we plan to start our work under the "two-level" MI learning framework.

Moreover, the "MI assumption", a very important and specific concept class of MI learning, was made to regard the relationship between instances inside the bags and the label of the bag. The first MI assumption originally comes from the application of Dietterich et al. [38]'s work. Almost all early work in MI learning follows this "standard MI assumption" class known to be appropriate for a drug activity prediction domaim. However, this "standard MI assumption" is not guaranteed to hold in other domains and it is not suitable for many applications [46]. In the MLO detection task, we have to drop this "standard MI assumption" and consider alternative assumptions. We should also develop some new methods based on the alternative MI assumptions.

Furthermore, when applying ATR methods to detect possible objects such as MLOs, the number of naturally occurring clutter objects (e.g. rocks, shipwrecks, fish) that are

detected is typically far higher than the relatively rare event of detecting MLOs. Thus, the number of non-MLOs is always much greater than the number of MLOs. Regardless whether the classification is based on a single image or multiple images of an object, the training data sets are always class imbalanced. Our preliminary research shows that both single-instance classifiers and multi-instance classifiers are always impacted by the class imbalance problem.

For single-instance classification of MLO detection we can apply many existing approaches, including sampling methods [22] [59] and cost-sensitive classification methods [11] [40] [108]. For multi-instance classification of MLO detection, however, there are to our knowledge very few related discussions about multi-instance class imbalance problems. Since the existing methodologies dealing with single-instance class imbalance datasets work on instances, we cannot apply them directly to multi-instance datasets which are comprised of bags of instances.

From the description above we can find that although many works have been done in MI research, there exist many areas in MI learning which are not studied enough, in detail:

1. In recent years, numerous methods have been proposed to learn from multi-view data by considering the diversity of different views [115]. The basic idea of Multi-view learning is to use the consistency among different views to achieve better performance. In Multi-view learning, each particular view is learned by introducing one model, and then the redundant views of the same input data are exploited to improve the learning performance by jointly optimizing all the models [115]. The views can be obtained from multiple sources or different feature subsets. This is close to MI learning. However, the difference is that MI learning uses multiple instances to describe the object (bag), while multi-view learning uses multiple groups of features (views) in one instance. If we can find a way to transfer the MI problem into a multi-view problem in a 'two-level" MI framework, we can gain the benefits of both by applying existing single instance method for MI problem, and then take advantage of learning the second-level using different views.

2. Efficiency is a factor which always needs to be considered when developing new algorithms. We find that some "two-level" MI learning methods have high computational complexity [116] and others have not [47]. In many cases, keeping competitive accuracy with lower computational complexity is very important for developing new algorithms. In this research we are trying to develop such kinds of methodologies for combining MI learning with other existing technologies, such as data fusion. Data fusion is a technology which collates information from different sources. It has also been widely used in sonar imagery recognition [43][77][83][84]. The relationship between data fusion and MI learning has not been studied enough. The question that arises with these two frameworks is: can data fusion be used as a MI learning framework? If the answer is yes, we may find a way to develop a new methodology that has competitive performance with lower running time.

3. Some ensemble technologies have been used to combine classifiers to get better performance. Due to the scarcity of MI data, convincing tests and comparisons between

MI algorithms combined in ensemble methods are not researched enough. Thus, recognizing the connection between the various ensemble methods can sometimes inspire new solutions with well-founded theoretic justifications. For instance, the multi-kernel method [8] [90] has been shown to be a very strong ensemble algorithm which can combine different kernel functions into a final decision for classification. Can the multiple kernel frameworks be successfully used in MI learning? This question is worthy of further research.

4. While significant research efforts have been put into MI learning and many approaches have been proposed to tackle the MI problem, it is important to note that the performance of classification can be affected by the class imbalance problem when the training data distribution is skewed. Our research shows that similar to the single-instance imbalanced problem, classification of MI data with imbalanced class distributions significantly degrades the performance of most standard MI algorithms. Due to the inherent differences between MI learning and single-instance learning, the existing solutions for single-instance class imbalanced problems do not transfer directly to MI datasets. This is a drawback, as in our experience imbalanced MI problems often occur in the data mining problem.

Therefore we believe that the study of MI learning still requires a stronger and clearer theoretical interpretation, as well as use of more practical and artificial datasets for the purpose of evaluation. In order to capture the big picture of MI learning, we set up the following objectives:

- To create a new MI method which combines the multi-view learning method with MI approaches.
- To create a data fusion framework for MI learning.
- To create a new MI ensemble method using the multi-kernel method.
- To analyze the class imbalance MI problem and establish a general framework for this problem.
- To create new MI methods using algorithms extended from existed imbalanced single instance learning for imbalanced MI problem.
- To apply the proposed algorithm on a collection of datasets from real-world applications.

We aimed to achieve these objectives to the best of our abilities. Although some objectives may be too large-scaled to be fully accomplished, we hope that this thesis will improve the understanding of the MI problem.

1.2 Contributions

Although many algorithms were previously presented for MI learning, there is still a requirement to increase the efficiency of these algorithms. In this thesis we present several new approaches for the general MI problem. They are the Multi-view two-level concepts framework, the data fusion framework, and the Multi-kernel SVM framework. Due to the high efficiency of these learning frameworks, we expect better learning performance with respect to running time and accuracy.

In recent years, the imbalanced learning problem has drawn a significant amount of interest from academia, industry, and government funding agencies. The fundamental issue with the imbalanced learning problem is that imbalanced data can significantly compromise the performance of most standard learning algorithms. Although a lot of research has been done in the area of the class imbalance problem, in the context of MI learning, there is no discussion about this problem. In this thesis, we are the first to present and analyze the class imbalance problem of MI learning. We proposed several novel frameworks to solve this problem. They are the resampling framework, the cost sensitive boosting framework, and the instance-weighted boosting SVM framework. Part of this research has been published in some related papers [100][103][105][106][107][108].

1.3 Thesis Layout

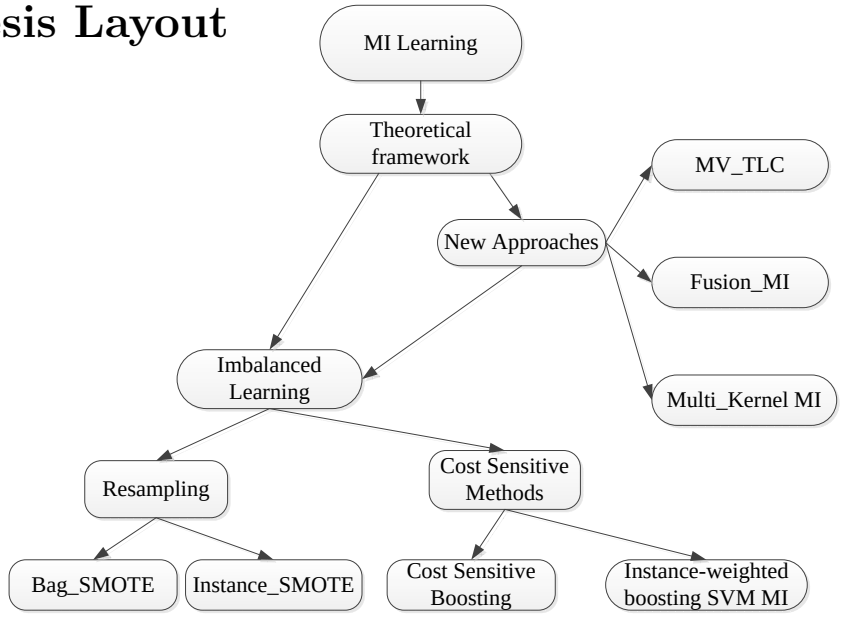

Figure 1.1: The framework of this research

The framework for this research is presented in Figure 1.1. This research could be divided into two parts. The first part is about solving the general MI problem using new methods to learn MI datasets. The second part is about the class imbalance problem of MI learning.

In the first part, we propose three solution frameworks for the MI problem. In the first framework, a multi-view two-level learning approach is presented. In the second framework,

we present a novel data fusion framework for learning MI datasets. In the third framework, a multi-kernel MI solution is presented.

In the second part, we also propose three solution frameworks for the MI class imbalance problem. In the first framework, we explore MI data sampling methods. In the second framework, we present a novel generalized version of MI cost-sensitive boosting technique. In the third framework, we present an instance-weighted boosting SVM technique for MI learning.

Experimental results — on both benchmark datasets and application datasets — show that the proposed frameworks are effective solutions for both of the general MI problem and imbalanced MI problem.

Chapter 2 gives an overview of the MI problem and its existing solutions. In chapter 3, a multi-view two-level concepts approach is applied for the MI problem. In chapter 4, a data fusion framework is given and discussed. In chapter 5, a multiple kernel SVM approach is applied for the MI problem. chapter 6 gives an overview of the class imbalance problem, the existing solutions, the discussion of the imbalanced MI problem and a brief overview of the options available for evaluation. Moreover, chapter 6 also discusses approaches for the imbalanced MI problem. The resampling approaches and a cost sensitive boosting approach is given and discussed. In chapter 7, an instance-weighted boosting SVM approach is presented for imbalanced MI problem. In chapter 8, a case study is given on the application of mine like objects using MI approaches. For each presented approaches, we discuss the experiments designed to evaluate these presented approaches. Subsequently, the implementation details, and experimental results obtained is also discussed. Finally, in chapter 9, we state our conclusions and our future work.

Chapter 2

Background and Literature Review for the MI Problem

2.1 Homogeneous learning and heterogeneous learning

The supervised learning is called "supervised" because the class labels are provided in the data. This is in contrast to the unsupervised learning, where the classes are not known. The task of the algorithm of the supervised learning is to build a model that will generate predictions of the labels of future examples.

A data set D used in the learning task consists of a set of data records, which are described by a set of attributes $A = \{A_1, A_2, \cdots, A_{|A|}\}$, where $\mid A \mid$ denotes the number of attributes or the size of the set A. The data set also has a class attribute/label C. A data set for learning is simply a relational table containing a set of data records. Each data record which is called an instance in machine learning describes a piece of "past experience". The objective of learning on D is to produce a classification/prediction function to relate values of attributes in A and classes in C. The function can be used to predict the class values/labels of the future data. The function is also called a classification model or simply a classifier.

In traditional supervised learning, each data record describes a piece of information. We call this kind of supervised learning "homogeneous learning". However, in real-world learning tasks, sometimes the structure of the information is very complex and we have to use different data structure to describe it. Then we have the chance to meet different data structures, in response there are different supervised learning models. We call these kinds of learning "Heterogeneous learning". Some commonly found heterogeneities include task heterogeneity (as in multi-task learning), view heterogeneity (as in multi-view learning), instance heterogeneity (as in multi-instance learning), and label heterogeneity (as in multi-label learning).

Figure 2.1 gives the structure of machine learning. Under the supervised learning, multi-view learning, data fusion, multi-task learning all belong to the Heterogeneous learning.

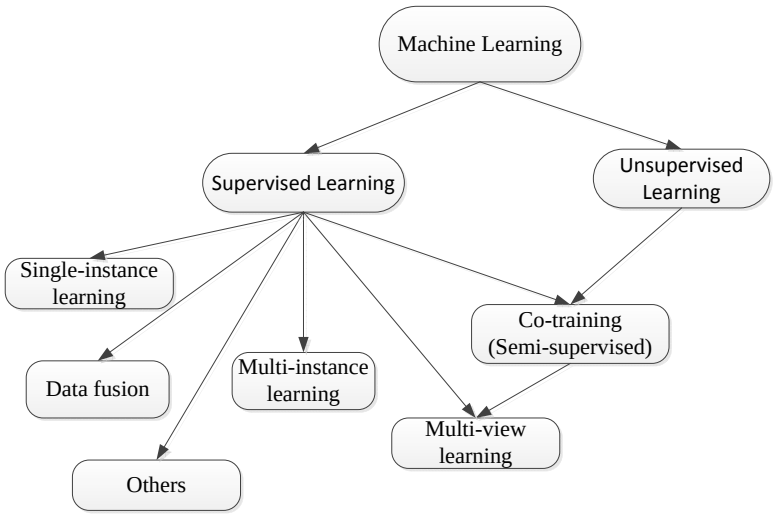

Figure 2.1: The structure of machine learning

2.2 The description of the MI problem

In single-instance learning, one piece of "past experience" is described by one instance. If the piece of "past experience" needs more than one instance to describe it, this learning task becomes a MI learning task. We call the group which contains instances describing one piece of "past experience" a "bag" and all of the instances in this data set have the same number of attributes.

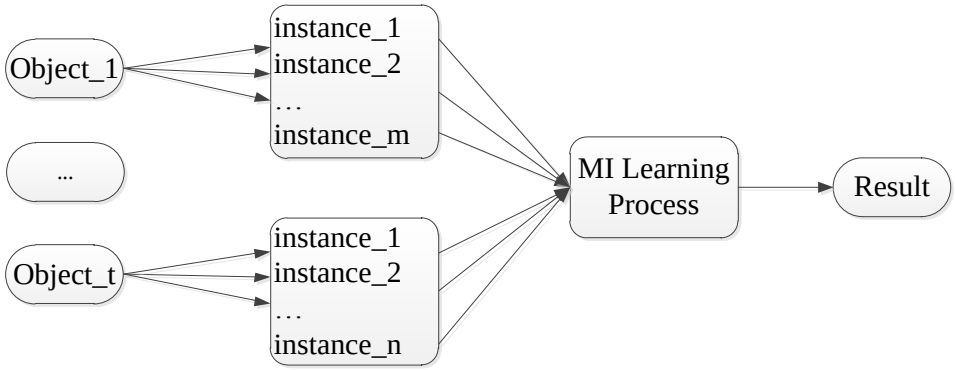

Figure 2.2: Data generation for MI learning

The difference between the single-instance learning and the MI learning can be depicted

graphically as shown in Figure 2.2 [38]. In this figure, the "object" is an example described by some attributes, the "result" is the class label and the MI learning process is the relationship. In single-instance learning, there is only one instance in each bag while in MI learning there is more than one instance although not all of the instances are useful for the learning process.

The first MI problem arises in drug activity prediction [38] about choices of representation for drug molecules. Drug molecules can adopt multiple shapes by rotating some of their internal bonds as shown in Figure 2.3. Every combination of angles of the rotatable bonds of a molecule defines a "conformation". Here, one molecule and its conformations can be treated as a group. If we use a set of features to represent this molecule, we will get one pattern, and each conformation of this molecule can also be represented by a pattern with same features. All of these patterns will build a bag which represents this molecule, and each pattern is an instance inside this bag. In the drug activity experiment, if there is one conformation of a molecule that is active, this molecule is defined as active. If a molecule is defined as inactive, that means there are no conformations of this molecule that are active. Since every molecule and its conformations have only one label (active or inactive), the bag of this molecule has only one label. Here we have the first MI learning example.

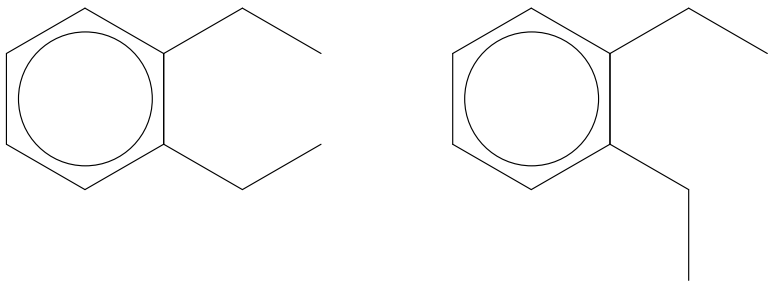

Figure 2.3: Different shapes of drug molecules in one conformation

To help clarify this MI learning concept, Chevaleyre & Zucker [30] refer to an example as the simple jailer problem. Imagine that there is a locked door, and there are N key chains with each containing a bunch of keys. If a keychain contains a key that can unlock the door, that key chain is considered to be useful. The learning problem is to build a model that can predict whether a given key chain is useful or not. Here each key is a pattern and the keychain may contain a different number of keys. If each key represents an instance, the keychain is a "bag" and this bag has only one pattern — useful or not. If the keychain is useful, that means there is at least one key that can unlock the door. Otherwise, there must be no key in this keychain that can unlock the door.

Despite the differences, MI learning basically adopts the same setting as the single-instance supervised learning. It still has examples with attributes and class labels, one example still has only one class label, and the task is still the inference of the relationship between attributes and class labels. The only difference is that every one of the examples is represented by more than one feature vector. If we regard one feature vector as an instance

of an example, normal supervised learning has only one instance per example while MI learning has multiple instances per example, hence named "MI learning" problem.

The MI learning problem described in [38] and [30] can be defined as:

Given: a set of bags $B_i, i = 1, \cdots, N$, their label $c(B_i) \in \{0, 1\}$, and the instances e_{ij} $(j = 1, \cdots, n_i)$ belonging to each bag.

Output: An unknown function f that classifies individual instances as 1 or 0, and for which it holds that $c(B_i) = 1$ if and only if there exists $e_{ij} \in B_i : f(e_{ij}) = 1$ (MI constraint, MIC).

Within a two class context, with class labels positive and negative, the original MI assumption states that an example is positive if at least one of its instances is positive and negative if all of its instances are negative. We refer to this MI assumption as the "standard MI assumption"[46] [110].

2.3 One-level MI learning and Two-level MI learning

In the area of MI learning, many researchers have accomplished a lot of work and made much progress. After Dietterich et al. [38] have introduced the Axis-Parallel hyper-Rectangle (APR) methods for MI learning, many MI learning approaches have been developed.

As we already mentioned in chapter 1, MI learning approaches can be categorized into "one-level" MI learning methods and "two-level" MI learning methods. "one-level" approaches denotes the methods that extend the single-instance learners to deal with MI data. Among these approaches, Diverse Density (DD) [73] and the Expectation-Maximization Diverse Density [123] were proposed as general frameworks for solving MI learning problem. For tree methods, Blockeel et al. [14] proposed a MI tree method (MITI), and Bjerring et al. [13] extended this in their work by adopting MITI to learn rules (MIRI). Andrews et al. [4] proposed two approaches to modify Support Vector Machines [33]: mi-SVM for instance level classification and MI-SVM for bag level classification.

Different from "one-level" MI learning methods, "two-level" MI learning methods tackle the MI problem by introducing two levels of data space: instance-level and bag-level. At the first level, the structure of the instance space is learned, and at the second level, the interaction that leads to a bags class label is discovered.

Weidmann et al. [110] introduced the idea of two-level-classification (TLC) to tackle generalized MI problems by constructing a single "meta-instance" from a bag. Xu [116] presented a group-conditional approach called "two-level distribution (TLD) approach". Zhou & Zhang [126] employs k-means with a different number of clustered groups to build the "meta-instance".

Chen et al. [26] introduced an approach with the name Multiple-Instance Learning via Embedded Instance Selection (MILES) to MI learning based on the diverse density framework. Moreover, Xu & Frank introduced two simple heuristic algorithms called MIWrapper [47] and MIboost [117] which are also "two-level" MI learning approaches.

Recent research [26] [47] [110] [117] [126] shows that "two-level" MI learning methods are very competitive with the more sophisticated and state-of-the-art "one-level" MI learning methods. In general, "two-level" MI learning methods have these following benefits:

1) Several methods [47] [117] are not difficult to implement and their performance is outstanding.

2) Almost all existing state-of-the-art single instance methods can be combined with these "two-level" MI learning methods.

2.4 MI Assumptions

After the first MI problem arises in [38], much research and progress has been accomplished by many researchers as we have already mentioned in chapter 1. Note that the standard MI assumption is based on instance-level class labels, thus the "standard MI Learning process" is commonly viewed as a two-step process with the first step consisting of a classification problem and the second step that is a "positive instances selecting" process based on the first step where the standard MI assumption is applied. This assumption played an influential role in the early stage of MI learning. However, as we shall see, there are methods that directly infer the output from the interactions between the input instances without the instance selection step. We noticed that several MI methods do not actually follow the standard MI assumption nor explicitly state which assumptions they use. As a matter of fact, we believe that this standard assumption may not be essential to make accurate predictions. What matters is the combination of the model from the first step and the assumption used in the second step. Given a certain type of model for classification, it may be appropriate to explicitly drop the standard MI assumption and establish other assumptions in the second step.

We agree with the statement mentioned in [116] that the two-step paradigm itself is one possibility for modeling the MI problem. For the instances within a bag for an example, there may be ambiguity, interactions and redundancy to exploit. The two-step model and the standard MI assumption may be suitable to exploit ambiguity [66] but if we are interested in other properties of the examples, other models and assumptions may be more appropriate. We may need more background knowledge to choose the right way to model the problem, but in most cases we lack such knowledge. Consequently it is necessary that we try a variety of models or assumptions in order to come up with an accurate representation of the data. Systematic reviews of MI assumptions have been given in [46], [116] and [118]. To explain the algorithms and models we will present in the following chapters, we shall also review those major MI assumptions and the algorithms following these assumptions.

In this section we review the MI assumptions classes that have been proposed and the algorithms under these assumptions. Figure 2.4 gives the relationships between the various major assumptions. Table 2.1 gives the list of these MI assumptions and algorithms under these assumptions. In the following contents of this chapter we will discuss some major assumptions as the representatives and the algorithms under these assumptions in detail.

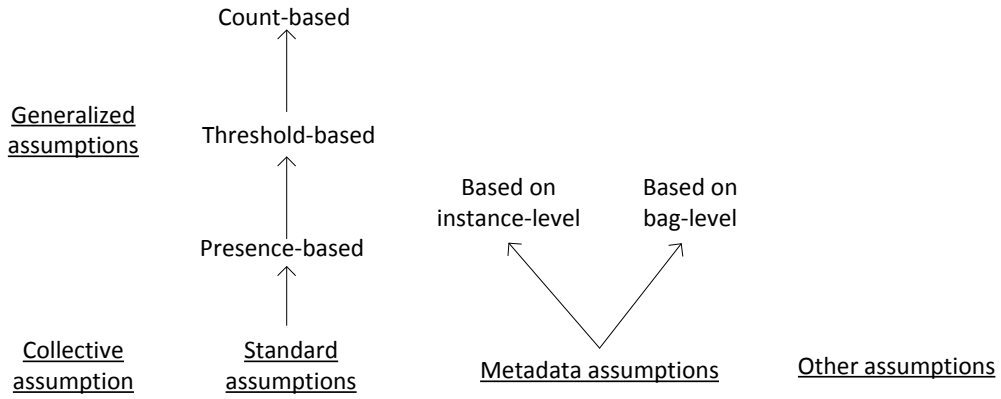

Figure 2.4: Relationships between MI assumptions [46]

Table 2.1: MI assumptions and algorithms

Name	Algorithms
the standard assumption	APR[38]; DD[73]; EM-DD[123]; MI-SVM[4]; NN[81]; mi-SVM[4]; MITI[14]; MITR[30]; Boosting[5]
the generalized assumption	TLC[110]; CCE[126];
the collective assumption	MIboost [117]; MIwrapper[47]
the metadata-based assumption	Simple MI[39]; Relic[85]; K_MI kernel[49]
the GMIL assumption	GMIL-1[87]; GMIL-2[94]
the count-based GMIL assumption	Count-Based GMIL[94]
the DD-SVM/MILES assumption	DD-SVM[27]; MILES[26]
the BARTMIP assumption	BARTMIP[122]
the nearest neighbor assumption	Bayesian-KNN[101]; Citation-KNN[101]
the MI graph assumption	MIGraph and miGraph[124]
the instance weights assumption	IFLIW[45]; YARDS[45];

2.4.1 The standard MI assumption

As we already introduced in section 2.2, the standard MI assumption is the original assumption for the MI problem. Weidmann et al. [110] indicated that by employing different assumptions of how the instances'classifications determine their bag's label, different kinds of MI problems can be defined.

Following the notation of Gärtner et al. [49] and Weidmann et al. [110], we need to assume a binary class attribute $\Omega = \{+, -\}$. Let χ be the instance space and $\mathbb{N}$ be the input space. Then an MI concept is a function $v_{MI} : \mathbb{N}^{\chi} \longrightarrow \Omega$. The task in MI learning is to learn this function based on a number of example elements of the function. Here, $\mathbb{N}^{\chi}$ refers to the set of all functions from χ to $\mathbb{N}$ (which is isomorphic to the set of all multi-subsets of χ). The output of $f(\chi) \in \mathbb{N}^{\chi}$ is viewed as the number of occurrences of χ in the multi-set.

Under the standard MI assumption, the MI learning problem can be defined as:

$$v_{MI}(X) \Leftrightarrow \exists x \in X : c_I(x) \tag{2.1}$$

where $c_I \in C$ is a concept from a concept space C (usually called the "underlying concept"), and $X \subseteq \chi$ is a set of instances.

The *standard MI assumption* states that each instance has a hidden class label $c \in \Omega = \{+, -\}$. Under this assumption, an example is positive only if one or more of its instances are positive. Thus, the bag-level class label is determined by the disjunction of the instance-level class labels.

A number of learning algorithms for MI classification under the standard MI assumption have been proposed in the literature. After first presenting the MI problem, Dietterich et al. [38] presented several algorithms for learning axis-parallel rectangles to identify the positive region of instance space. Learning axis-parallel concepts [38] is the first class of algorithms that were proposed to solve MI problem with the idea of finding an axis-parallel hyper-rectangle (APR) in the feature space to represent the target concept. Obviously, this APR should contain at least one instance from each positive bag and exclude all the instances from negative bags. Dietterich et al. [38] then suggested three algorithms to find such a hyper-rectangle: a "standard" algorithm finds the smallest APR that bounds all the instances from positive bags; an "outside-in" algorithm constructs the smallest APR that bounds all the instances in positive bags and then shrink the APR to exclude false positives; an "inside-out" algorithm starts from a seed point which grows into a rectangle with the goal of finding the smallest APR that covers at least one instance per positive bag and no instances from negative bags. The three algorithms are evaluated on one artificial and two real datasets, and the "inside-out" algorithm is shown to be the most effective one.

Blockeel et al. [14] proposed a tree method which is called MITI (multi-instance tree inducer) for MI learning. Similar to a single-instance decision tree (like C4.5 [79]), the MI tree is based on the information gain of a feature of the instance. The difference between the multi-decision tree and the single-decision tree is that the growing of a MI tree is based

on the information gain of a feature over a set of instances instead of the feature of one instance. The concept of information gain and entropy are extended to bags of instances in the MI learning framework. Suppose S is a collection of instances which belong to $p(S)$ positive bags and $n(S)$ negative bags, F is the feature being considered as the splitting criterion and S_n is the collection of instances whose value of feature F is n. The extended information gain and entropy are defined as:

$$Entropy_{multi}(S) = -\frac{p(S)}{p(S)+n(S)} \times \log_2(\frac{p(S)}{p(S)+n(S)}) - \frac{n(S)}{p(S)+n(S)} \times \log_2(\frac{n(S)}{p(S)+n(S)}) \tag{2.2}$$

$$InfoGain_{multi}(S,F) = Entropy_{multi}(S) - \Sigma_{n \in Values(F)} \frac{p(S_n)+n(S_n)}{p(S)+n(S)} \times Entropy_{multi}(S_n) \tag{2.3}$$

Moreover, compared to single-instance decision tree, MITI has two key modifications: (a) nodes are expanded in best-first order guided by a heuristic that aims to identify pure positive leaf nodes as quickly as possible, and (b) whenever a pure positive leaf node is created, all positive bags containing instances in this leaf node are deactivated.

Andrews et al. [4] proposed two approaches to modify Support Vector Machines, mi-SVM for instance-level classification and MI-SVM for bag-level classification. The goal of mi-SVM is to maximize the usual instance margin jointly over the unknown instance labels and a linear or kernelized discriminant function. In comparison, MI-SVM aims at maximizing the bag margin, which is defined as the margin of the "most positive" instance in case of positive bags, or the margin of the "least negative" instance in case of negative bags. Both methods were implemented using mixed integer quadratic programming. More discussion will be given in chapter 5 and chapter 7.

For other approaches, Maron & Lozano-Perez [73] defined diverse density, a measure of the likelihood that a point in instance space is a positive target concept, and used a gradient search to find the point that is most likely to define the target concept. A refinement of this algorithm, EM-DD, was proposed by Zhang & Goldman [123]. Several single-instance learning methods have been 'upgraded' to the MI scenario under the standard MI assumption, including neural networks [81] and weak learners for boosting [5]. Zhou & Xu [125] adapted semi-supervised support vector machines to the standard MI scenario by encoding a 'positive constraint' in the objective function of the SVM.

2.4.2 The generalized MI assumption

Although the standard MI assumption is widely believed to be appropriate for the musk drug activity prediction problem [38], the MI representation can be applied to a number of other problem domains where the standard MI assumption might not be directly applicable. Based on this, Weidmann et al. [110] formulated a hierarchy of generalized instance-based assumptions for MI learning. The hierarchy consists of the standard MI assumption and three types of generalized MI assumptions: presence-based MI, threshold-based MI and count-based MI, each more general than the previous.

If we use $v_{PB} : \mathbb{N}^{\chi} \longrightarrow \Omega$ to define a presence-based MI concept, $v_{TB} : \mathbb{N}^{\chi} \longrightarrow \Omega$ to define a threshold-based MI concept, and $v_{CB} : \mathbb{N}^{\chi} \longrightarrow \Omega$ to define a count-based MI concept, then the formal definitions of presence-based MI, threshold-based MI and count-based MI can be shown in Equations 2.4, 2.5 and 2.6 [110].

$$v_{PB}(X) \Leftrightarrow \forall c \in \hat{C} : \Delta(X, c) \geq 1 \tag{2.4}$$

$$v_{TB}(X) \Leftrightarrow \forall c_i \in \hat{C} : \Delta(X, c_i) \geq t_i \tag{2.5}$$

$$v_{CB}(X) \Leftrightarrow \forall c_i \in \hat{C} : t_i \leq \Delta(X, c_i) \leq z_i \tag{2.6}$$

Here we still follow the notation of Gärtner et al. [49] and Weidmann et al. [110]. In these equations, $\hat{C} \in C$ is the set of required concepts, $\Delta : \mathbb{N}^{\chi} \times C \rightarrow \Omega$ is the function that outputs the number of occurrences of a concept in the bag, $t_i \in \mathbb{N}$ is the lower threshold for concept I, and $z_i \in \mathbb{N}$ is the upper threshold for concept I. We have $\nu_{MI} \subset \nu_{PB} \subset \nu_{TB} \subset \nu_{CB}$.

The threshold-based MI assumption states that a bag is positive only if there are at least a certain number of instances in the bag that belong to each of the required concepts. Each concept can have a different threshold.

For approaches to be used in wider domains, algorithms that rely on the *generalized MI assumption* may be more appropriate. Under this assumption, Weidmann et al. [110] presented a Two-Level Classification (TLC) algorithm to learn the type of MI concepts that are described in their concept hierarchy [110]. Similarly, Zhou & Zhang [126] presented a constructive clustering ensemble (CCE) method. This algorithm uses a clustering method to group the instances in the training bags into d clusters, to build the concepts for the bag level classification.

2.4.3 The collective MI assumption

The collective assumption, designed as a general alternative to the standard MI assumption, was not precisely defined by Xu [116] but was defined by Foulds & Frank [46]. It is motivated by a view of the nature of MI bags that is based on probability theory. Under this view, a bag is not a finite collection of fixed elements as is generally assumed, but instead is a sample of an underlying population specific to that particular bag. Under this assumption, a bag can be modeled as a probability distribution $Pr(X \mid b)$ over the instance space, where the observed instances were generated by random sampling from that distribution.

Instances are assumed to be assigned class labels according to a probability function $g(x) = Pr(X \mid b)$. Under this assumption, the bag-level class probability function is determined by the expected class value of the population of that bag. Let c be a class label and let b be a bag. Then

$$Pr\left(c \mid b\right) = E_X\left[Pr\left(c \mid x\right) \mid b\right] = \int_X Pr\left(c \mid x\right) Pr\left(x \mid b\right) dx \tag{2.7}$$

To compute this exactly, we must know $Pr(X \mid b)$, the probability distribution for the bag. However, this is generally not known in practice so the sample provided by the instances in the bag is used instead:

$$Pr\left(c \mid b\right) = \frac{1}{n_b} \sum_{i=1}^{n_b} Pr(c \mid x_i) \tag{2.8}$$

where n_b is the number of instances in the bag. In the limit, as the sample size approaches infinity, the sample version of the equation will approach the population version.

Xu & Frank developed statistical algorithms for learning this kind of probabilistic concept, the most notable of which are versions of logistic regression and boosting, upgraded to solve MI learning problems under the collective MI assumption [117].

For single-instance classification, Logistic Regression assumes a parametric form for the distribution $Pr(Y \mid X)$, then directly estimates its parameters from the training data. The parametric model assumed by Logistic Regression in the case where Y is a boolean is:

$$Pr(Y = 1 \mid X) = \frac{1}{1 + \exp(\omega_0 + \sum_{i=1}^{n} \omega_i X_i)} \tag{2.9}$$

and

$$Pr(Y = 0 \mid X) = \frac{\exp(\omega_0 + \sum_{i=1}^{n} \omega_i X_i)}{1 + \exp(\omega_0 + \sum_{i=1}^{n} \omega_i X_i)} \tag{2.10}$$

However, the standard logistic regression model does not apply to MI data because the instances' class labels are masked by the "collective" class label of a bag. Xu & Frank [117] use a two-stage framework to upgrade linear logistic regression and boosting to MI data.

As usual, the maximization of the log-likelihood function is carried out via numeric optimization because there is no direct analytical solution. The optimization problem can be solved very efficiently because we are working with a linear model.

Moreover, Xu & Frank [117] introduced a simple heuristic algorithm called MIWrapper for applying single-instance learners under the collective assumption. The first step of the MIWrapper algorithm is to collect all of the instances from all of the bags, and label each of them with the label of the bag that they came from. This step effectively creates a propositional (i.e. single-instance) dataset. To set equal total weight for each bag, the algorithm weights all of the instances. Then a single-instance learner is applied to this propositional dataset. At classification time, the single-instance learner predicts class probabilities for all of the instances in the bag for which the classification is to be predicted. The output is merely the average of the predicted instance-level class probabilities. Using the arithmetic mean at prediction time, the method applies the sample version of the collective assumption formula when making predictions.

2.4.4 The Metadata-based MI assumptions

Since single-instance learning algorithms have been developed for a long time, a simple idea to MI learning is to transform the MI data into single-instance dataset and a single-instance

learning algorithm can then be applied directly to dataset. One approach is to replace each bag with a feature vector consisting of metadata features derived in some way from the instances in that bag [46]. At classification time, new bags are mapped into the metadata feature space, and predictions are made by outputting the prediction of the single-instance learner for the transformed version of the bag. Xu [116] refers to methods of this kind as metadata approaches. When this type of method is used, the implicit assumption is merely that the classification labels of the learning examples are directly related to the metadata. We will therefore refer to this type of MI assumption as a metadata assumption.

Lin [39] described three versions of a MI approach with the name "Simple MI" under the metadata assumption, each of which differs only in the type of summary statistics used for the single-instance feature space. The first two methods merely average the values of the instances in a bag for each dimension, using either the arithmetic or the geometric mean. The third option is called the 'minimax' method. Here, the minimum and maximum values of each variable are recorded for each bag. This method is equivalent to Gärtner et al.'s [49] minimax kernel, used as a kernel in a standard support vector machine algorithm.

The main advantage of Simple MI is that it is extremely fast. The computation of the feature space transformation is trivial, and the single-instance base learner only has to learn from as many instances as there are bags in the training set, regardless of how many instances are contained inside the bags.

Although this simple model is not able to represent all types of problems, Lin [39] found that Simple MI (with appropriate base learners) performs surprisingly well on many datasets, even outperforming all of the special-purpose MI algorithms that were investigated in some cases.

The MILES [26], YARDS [45], BARTMIP [122], TLC [110] and CCE [126] algorithms all use feature space transformations, where bags are mapped to single-instance feature vectors, and single instance algorithms are applied to the resulting datasets. These methods can therefore be viewed as metadata approaches.

2.4.5 Other MI assumptions

Scott et al. [87] introduced a MI assumption with the name GMIL based on theoretical results from geometric pattern recognition. Under this assumption, they proposed the GMIL-1 algorithm. Tao & Scott [95] proposed GMIL-2 to improve the computational and memory efficiency of the algorithm. Tao et al. [94] proposed another assumption with the name Count-Based GMIL which is more general than GMIL assumption and Count-based assumption, with the algorithm Count-Based GMIL under it.

Related to the GMIL assumption, the DD-SVM [27] and its successor MILES [26] use a generalized MI assumption with the name DD-SVM/MILES assumption.

The BARTMIP algorithm under the so-called BARTMIP assumption which is proposed by Zhang et al. [122] is also closely related to MILES. While MILES assumes that bag labels are related to the instance-level distance from a set of target points, the BARTMIP method assumes that bag labels are related to distances from target bags.

There are two MI assumptions using instance weights introduced by Foulds et al. [46]. The first one, the weighted collective MI assumption, is an extended version of the collective assumption that incorporates a weight function over the instance space as well as a probability function. The other one, the weighted linear threshold MI assumption, is based on linear classification models from single instance learning. In response, Foulds et al. [46] presented two algorithms, IFLIW [46] and YARDS [46], under these two assumptions respectively.

Zhou et al. [124] proposed algorithms that depend on the MIGraph assumption that the spatial relationships between instances in bags are important contributors to bag labels. Under this assumption, two algorithms (MIGraph and miGraph) were presented.

Nearest neighbor assumptions were proposed by Wang & Zucker [101] when proposing two variants of the standard k-nearest neighbor algorithm in their work: The first algorithm (Bayesian-KNN) uses a Bayesian method for predicting the most likely class given a set of neighbors, while the second algorithm(Citation-KNN) is based on the notions of references and citers from the field of library and information science: when making a classification decision, not only are nearest neighbors of an example considered, but also bags that consider the example to be a nearest neighbor.

2.5 Discussion about current solutions

From this definition of the MI learning problem in section 2.1 we find that the standard assumption of the MI learning problem [116] is based on the MI constraint. The first MI algorithm stems from the pioneering paper by Dietterich et al. [38], which also introduced the aforementioned Musk datasets. The "APR and the MI assumption" combination dominated the early stage of MI learning.

Some more practical MI algorithms were developed in this domain, such as MULTINST [5], which tries to calculate the expected number of instances per bag that fall into the hypothesis for the positive class (in this case an APR) and adheres closely to the MI assumption. Diverse Density (DD) [73] and EM-DD [123] model were also heavily influenced by the "APR and the MI assumption" combination.

However, this assumption originally comes from the application of Dietterich et al. [38]'s work and may not be suitable for all other applications. Some new methods broke the MI assumption and APR-like formulations. The "Citation-KNN" method [101] definitely violates the standard MI assumption because it decides a test example's class label by the majority class of its nearest examples. Thus, it does not classify an example based on whether at least one of its instances is positive or all of the instances are negative. The SVM with the MI kernel [49] also violates the standard MI assumption.

MI learning has been widely used to combine additional information for decision making and classification [105][110][117]. However, most of the application cases violate the standard MI assumption. We should develop some methods that are not based on the standard MI assumption. Following this idea, data fusion, a technology which collates information from different sources, can be used for MI learning problem.

On the other hand, when using existing MI learning algorithms for learning in some applications, performance is not always as good as we expect. We must look for a better performance algorithm for our tasks. As in the motivation of this research which was mentioned in chapter 1, one potential approach is the Multi-view learning framework [115] which has been used widely to improve performance of learning methods. The other potential approach is the Multiple Kernel Learning (MKL) method, an approach that has been reported to have good performance in many learning cases [8][50][90].

The last point about the drawback of existing MI learning methods is that there is no other research about the class imbalance problem in MI learning, as far as we know. Since the class imbalance problem exists in many MI learning application, it is necessary to have a comprehensive and systematic analysis on this problem.

All of the issues above are the motivations behind our work in this thesis. In the next chapters we will discuss them in detail.

Chapter 3

A Multi-View Two-level Learning Method for Generalized MI Problems

3.1 Motivation

In this chapter, we address the MI problem under the generalized MI assumption since it is less specific. Under this assumption, we introduce the idea of multi-view, two-level-classification to deal with the MI problem. The first step of this method constructs a single meta-instance from a bag, which represents regions in the instance space and has an attribute for each region. Every attribute indicates the number of instances in the bag that can be found in the corresponding region. By repeating this step using different construction methods, we get a multi-view attribute set containing attributes that represent regions in different instance spaces. Along with the bag's class label, the multi-view meta-instance can be passed to a standard propositional learner, to learn the influence of the regions on a bag's classification. In the second step, we introduce a multi-view approach to learn from the multiple independent sets of features generated in the first step. Multi-view learning describes the problem of learning from multiple independent sets of features (i.e. views) of the presented data. This framework has been successfully applied to many real-world applications [15][18][19][44]. Indeed, a multi-view learning problem with n views can be seen as n strongly uncorrelated feature sets that are distributed in the dataset.

This multi-view, two-level classification (MV-TLC) [107] strategy transforms a MI dataset into a multi-view, single meta-instance dataset, and learns from multiple views (the feature set) of this meta-dataset. The information acquired by view learners is then integrated to construct a final classification model. Our empirical studies show that our method compares well to other popular MI classifiers.

Unlike most current MI learning algorithms, which are derived from supervised learning algorithms by shifting the focus from the instances to the bags (i.e. adapting single-instance algorithms to the MI representation), our method demonstrates the feasibility of another approach to solving MI learning: adapting the MI representation to the single-instance algorithms. The main contribution of this chapter is that we transform the MI problem into

a multi-view single instance problem. This provides two potential benefits: any existing single instance algorithm can be applied, and the presented multi-view learning method uses the consistency among different views to achieve better performance.

The remainder of the chapter is structured as follows: Section 3.2 presents the related concepts; Section 3.3 describes and discusses the proposed algorithm, it also illustrates the efficiency of our algorithm as determined by experimentation and offers final remarks; Section 3.4 presents the conclusion and future work.

3.2 Multi-view learning

In recent years, numerous methods to learn from multi-view data by considering the diversity of different views have been proposed. The views can be obtained from multiple sources or different feature subsets. The basic idea of Multi-view learning is to make use of the consistency among different views to achieve better performance. As opposed to single view learning, multi-view introduces one function to model a particular view, and jointly optimizes all the functions to exploit the redundant views of the same input data to improve the learning performance. The Multi-view learning process has three stages: Multi-view construction, views validation and views combination.

View construction methods can be analyzed and categorized into three classes. The first class includes techniques that construct multiple views from so-called meta-data, using random approaches. Creating different views corresponds to feature set partitioning, which generalizes the task of feature selection. Instead of providing a single representative set of features, feature set partitioning decomposes the original set into multiple disjoint subsets to construct each view. For example, Brefeld et al. [18] presented a simple way to convert from a single view to multiple views, by splitting the original feature set into different views at random. Di & Crawford [36] conducted a thorough investigation of view generation for hyper-spectral image data. Three strategies: Clustering, Random selection and Uniform band slicing, have been proposed to construct multiple views by considering the key issues of diversity, compatibility and accuracy. Guo & Viktor [51] also presented a method using conventional data mining methods to classify relational objects without having to flatten the multiple relations into a universal one.

The second class consists of algorithms that reshape or decompose the original single-view feature into multiple views, such as the above matrix representations, or different kernel functions. For example, Wang et al. [109] developed a novel technique to reshape the original vector representation of a single view into multiple matrix representations. Matsubara et al. [74] proposed a pre-processing approach to simplify constructing the different views required by multi-view learning algorithms. Moreover, the literature shows that several kernel functions have been used successfully, and can be considered distinct views of the data. Thus, the problem of how to learn the kernel combination can be seen as multiple kernel learning [50].

The third class is comprised of methods that automatically perform feature set partitioning. Chen et al. [26] suggested a novel feature decomposition algorithm called Pseudo

Multi-view Co-training (PMC), which automatically divides the features of a single view dataset into two mutually exclusive subsets. This algorithm can find an optimal split of the features automatically, by solving an introduced optimization problem iteratively.

With respect to views validation, several approaches have been proposed to analyze the relationships between multiple views, or to cope with the problems resulting from the violation of view assumptions or the noise in the views. Muslea et al. [76] introduced a view validation algorithm that predicts whether the views are adequately compatible to solve multi-view learning tasks. This algorithm attempts to learn a decision tree in a supervised manner, in order to discriminate between learning tasks according to whether the views are sufficiently compatible for multi-view learning. Liu & Yuen [71] proposed two new confidence measures (inter-view confidence and intra-view confidence) to describe the view sufficiency and view dependency issues in multi-view learning. For multiple kernel learning, Lewis et al. [69] compared the performance of unweighted and weighted sums of kernels on a gene functional classification task. Christoudias et al. [31] presented a Bayesian localized approach for combining different feature representations with Gaussian processes that learns a local weight for each view.

Regarding views combination, Kumar & Daumé III [65] applied co-training to the unsupervised learning setting, and proposed a spectral clustering algorithm for multi-view data. In the Bayesian co-training proposed by Yu et al. [120], a Bayesian undirected graphical model for co-training through the gauss process is constructed. For multiple kernel learning, Lanckriet et al. [67] used a direct approach to optimize the unrestricted kernel combination weights. Gönen & Alpaydin [50] proposed an approach assigning different weights to kernel functions according to data distribution, and defined a locally combined kernel matrix.

3.3 Proposed Methods

The MI concepts described by Weidmann et al. [110] consists of a set of instance-level processes that are in some way related to the bag-level concepts. According to the definition of standard and generalized MI learning, the label of a bag is determined by the relationship between the feature vector set describing the bag and the target points in the instance space. There are two functions in the Weidmann et al. [110] concept hierarchy that determines a bag's class label: the mono-instance concept function that assigns an instance in a bag to a concept, and the MI concept function that computes a class label from the instances in a bag, given their concept membership by the first function. Thus, a two-level approach of learning is appropriate.

Most current MI learning algorithms use a strategy of adapting single-instance learning algorithms to meet the MI representation, and have been somewhat successful. And some MI learning algorithms using the opposite strategy, that is, adapting the MI representation to meet the requirements of existing single instance supervised learning algorithms, have also been effective [110][116][126].

The TLC approach proposed by Weidmann et al. [110] generates only one meta-instance

for each bag, using a standard decision tree. In contrast, CCE proposed by Zhou & Zhang [126] employs k-means with a different number of clustered groups to impose different structures on the instance space. In each structure, a meta-instance is generated for each bag. CCE also uses majority voting to combine the predictions of classifiers. It is clear that the role of the decision tree in TLC can be replaced by another supervised learning algorithm, and the role of k-means in CCE can be replaced by some other clustering method. This provides a starting point to develop an upgraded methodology based on TLC and CCE.

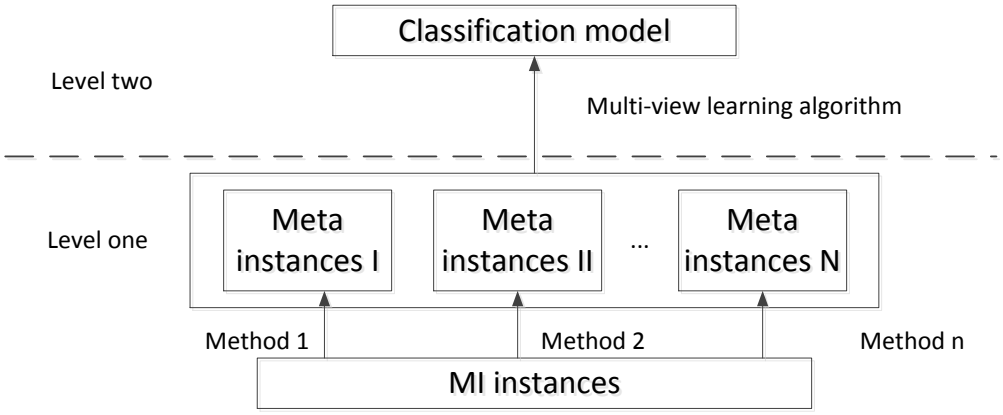

Figure 3.1: MV-TLC framework

In this thesis, we present a Multi-view two-level classification framework (MV-TLC) for generalized MI learning, as shown in Figure 3.1. In the first level, the framework imposes different structures on instance spaces with different dimensions, and in each structure a meta-instance is generated for each bag. Both the supervised and unsupervised learning algorithms are employed to generate the meta-instances. In the second level, a multi-view algorithm is applied to combine the predictions of the classifiers.

3.3.1 Construct the multi-view meta-instances dataset in level one

In level one, the instances in all the bags are collected. Supervised and unsupervised learning methods are then applied to construct a new concept for the second level. The same mapping is performed at classification time, and the bag-level predictions are made by the single-instance learner.

Using supervised learning methods to construct the meta-instances

Several algorithms can be used for the supervised learning methods, including decision tree (which is also used in TLC [110]) and rule induction. Here we chose a decision tree to impose a structure on the instance space. As shown in Figure 3.2, the decision tree is

built on the set of all instances contained in all bags, and labeled with their bag's class label. A unique identifier is assigned to each node or leaf of the tree, and information gain is used for test selection. A simple pre-pruning heuristic is applied, and nodes are not split further when the sum of the instance weights in the node is less than two. Each node in the tree represents a concept. Algorithm 1 illustrates the process.

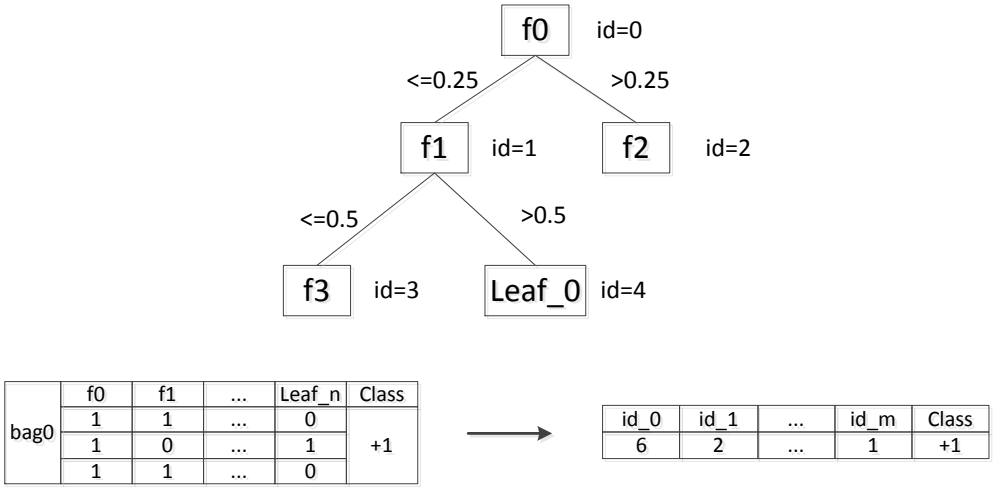

Figure 3.2: build the first level concepts using decision tree

In this algorithm, each bag is converted into a single-instance representation with an attribute for every node in the tree (i.e. each concept), the value of which is set to the number of instances that reach that node in the decision tree. The tree allows us to convert a bag into a single instance, with one numerical attribute for each node in the tree. Each attribute counts how many instances in the bag are assigned to the corresponding node in the tree. The TLC-decision tree algorithm proposed in Figure 3.3 is similar to the method used in TLC [110], but there are some important differences (e.g we do not initialize the weight of each instance). Unlike TLC, which only counts nodes as attributes, our method utilizes both the leaves and the nodes. Although only a TLC-decision tree method is proposed here, other supervised learning algorithm can also be used to construct new concepts in level one, such as rule induction algorithms and alternative tree methods.

Using unsupervised learning methods to construct the meta-instances

For the unsupervised learning methods, clustering algorithms are chosen to construct new concepts, and then applied to cluster the instances into d groups. Specifically, d features are generated so that if a bag has an instance in the i-th group, then the value of the i-th feature is set to 1; otherwise it is set to 0. Thus, each bag is represented by a d-dimensional binary feature vector, such that common single-instance supervised classifiers can be employed to distinguish the bags. Theoretically, any unsupervised learning algorithm can be used in the proposed TLC-clustering algorithm (e.g. k-means, EM). Figure 3.3 gives an

Algorithm 1 the TLC-decision tree algorithm

Given: D = the set of train bags; C = all instances in the bags in D
Set: L = decision tree classifier; F = a new single instance data set; $int\ i, j = 0$
for all $C_i \in C$ **do**
    $C_i.setClassValue(D_{(C_i)}.ClassValue)$
end for
$L.train(C)$
Output the final hypothesis:
set N = all nodes and leaves in L
while $j < D.size()$ **do**
    $F_j.setClassValue(D_j)$
    $F_j.setAttribute(N)$
    for $n \in \{1, N.size()\}$ **do**
        $F_j.attribute(n) = \sum count(N_{(C_i \in D_j)})$
    end for
end while
Return F

example of this process and Algorithm 2 illustrates the details of this method.

Algorithm 2 TLC-clustering

Given: D = the set of train bags $\{X^l\}$; C = all instances in the bags in D
Set: T = a set of m numbers $\{t_1, t_2, \cdots, t_m\}$; U = Clustering algorithm; S_i = a new single instance data set int $i, j = 0$
for $i \in \{1, \cdots, m\}$ **do**
    $U(C, t_i)$
    for $j \in \{1, \cdots, D.size()\}$ **do**
        for $k \in \{1, \cdots, t_i\}$ **do**
            $y_k^j \Leftarrow Overlap(X^j, U_k)$
        end for
        $S_i \Leftarrow S_i \bigcup \{< y_1^j, \cdots, y_{t_m}^j >\}$
    end for
    $S_i.setClassValue(D)$
end for
Return S_i

3.3.2 Learning the multi-view meta-instances dataset in the level two

Combining the generated concepts in level one gives us a multi-view single instance meta-dataset for level two. Here we can repeat the meta-data generation procession by using different supervised and unsupervised learners. For instance, for the decision tree algorithm

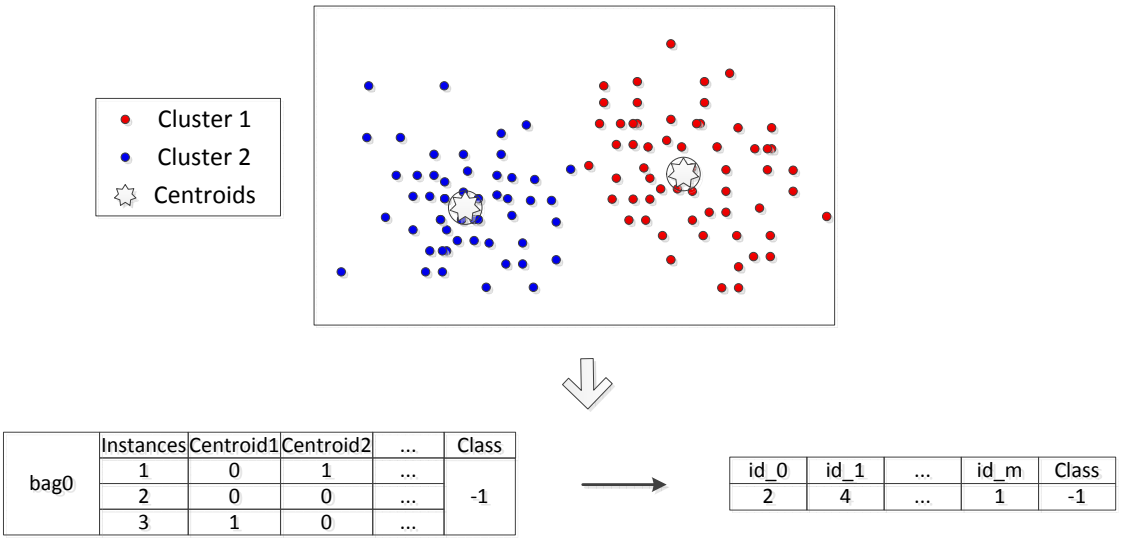

Figure 3.3: build the first level concepts using clustering method

in Algorithm 1, we have multiple choices such as J48 [79], Simple tree [92], Decision stump [58] and so on. We can also choose different parameters in these learners. Similarly, for the unsupervised algorithm in Algorithm 2, we also have multiple choices to get different learners by using different clustering methods. This is similar to the Multiple Kernel Learning (MKL) [49]. The benefit of generate more views is that we can collect less information when construct new concepts from the original multi-instance data. Simply merge these meta-dataset together, we will find that each bag in the original multi-instance data is transferred into a single instance. Applying more meta-data generation will increase the number of features in this meta-data.

To learn this meta-dataset, we can simply use any single-instance learning method, or we can use any feature selection method to process this data. Here we introduce an ensemble learning method combining with multi-view feature selection algorithm in level two of MV-TLC.

In [104], we presented a framework which uses multiple kernel SVM classifiers as the base learners for stacked generalization [113], a general method of using a high-level model to combine lower-level models, to achieve greater computational efficiency.

Stacked generalization is [113] a heterogeneous ensemble method for combining multiple classiers (base models) by learning a meta-level classier based on the output of the base-level classiers, estimated via cross-validation. We have found that since stacked generalization is an ideal ensemble method for heterogeneous base models, if we use feature selection method to generate multiple views on the meta-data constructed in level 1 in Figure 3.1, we can build a heterogeneous multi-view ensemble method in level 2 in Figure 3.1.

Algorithm 3 provides the details of the MV-TLC algorithm.

In the second level of the MV-TLC framework, the multiple views construction stage builds various hypotheses on the target concept, based on the multiple training data sets given by level one of the MV-TLC. Single view learners are used to learn the target concept from each view of the database separately.

Algorithm 3 MV-TLC learning algorithm

1: Given: dataset Φ which is generated from level one of MV-TLC
2: Set: L = view learner; M = meta learner
3: Output: the final hypothesis: $\pounds$
4: Let View Set $V = \emptyset$; Hypothesis set $H = \emptyset$
5: Generate view set $V = \{V^1, \cdots, V^m\}$ using the chosen correlation-based feature selectors and measurements from Φ
6: Train L with V, forming hypothesis set H
7: Form final model F by combining H, using M
8: Return $\pounds$

In the views validation stage, a number of different view learners are trained. All the learners from the views construction stage are evaluated in this stage, as they must be validated before being used by the meta-learner of the stacked generalization. This process is required to ensure they are capable of learning the target concept on their respective training sets. The algorithm uses a heuristic measure to evaluate the correlation between views; a similar heuristic principle was applied in the feature selection approach by Hall [53].

After completing construction of the view feature set, the MV-TLC algorithm ranks view feature subsets according to the correlation-based heuristic evaluation measure and selects the best ranked subset. Various heuristic approaches are employed to search the view feature space. The MV-TLC algorithm uses the best first and ranker [53] as the search method. A similar strategy has been applied by Guo & Viktor [51] .

In the last step of the MV-TLC, the multi-view learners from the views validation stage are incorporated into a meta-learner to construct the final classification model. The meta-learner is used to create a function to control the way that the view learners work together to achieve maximum classification accuracy. This function and the hypotheses constructed by each of the view learners constitute the final model.

3.3.3 Analysis

Blum & Mitchell [15] proved that when two adequate views are conditionally independent given the class label, co-training can be successful. Based on the same conditional independence assumption, Dasgupta *et al.* [35] provided the PAC style bounds for co-training. Let S be an $i.i.d$ sample consisting of individual samples $s_1, \cdots, s_m$. A partial rule h on a dataset X is a mapping from X to the label set $\{1, \cdots, k, \perp\}$, where k is the number of class labels and $\perp$ denotes the partial rule h that gives no opinion. We have the following for all pairs of rules h_1 and h_2:

If $\gamma_i(h_1, h_2, \delta/2) > 0$ for $1 \leq i \leq k$ then f is a permutation, and for all $1 \leq i \leq k$, we have:

$$P(h_1 = i | f(y) = i, h_1 \neq \perp) \leq \frac{1}{\gamma_i(h_1, h_2, \delta)} (\epsilon_i(h_1, h_2, \delta) + \hat{P}(h_1 \neq i | h_2 = i, h_1 \neq \perp)) \quad (3.1)$$

where $\epsilon_i(h_1, h_2, \delta) = \sqrt{\frac{\ln 2(|h_1|+|h_2|)+\ln 2/\delta}{2|S(h_2=i, h_1 \neq \perp)|}}$ and $\gamma_i(h_1, h_2, \delta) = \hat{P}(h_1 = i | h_2 = i, h_1 \neq \perp) - \hat{P}(h_1 \neq i | h_2 = i, h_1 \neq \perp) - 2\epsilon_i(h_1, h_2, \delta)$. Abney [1] relaxed this assumption, and found that weak dependence alone can lead to successful co-training. Given the mapping function $Y = y$, the conditional dependence of opposing-view rules h_1 and h_2 is defined as:

$$d_y = \frac{1}{2} \sum_{u,v} |Pr[h_1 = v | Y = y, h_2 = u] - Pr[h_1 = v | Y = y]| \tag{3.2}$$

And the weak dependence rule is :

$$d_y \leq p_2 \frac{q_1 - p_1}{2 p_1 q_1} \tag{3.3}$$

where $p_1 = \min_u Pr[h_2 = u | Y = y]$, $p_2 = \min_u Pr[h_1 = u | Y = y]$, and $q_1 = 1 - p_1$. As the proposed MV-TLC framework deploys multi-views constructed using supervised or unsupervised learning methods, h_1 and h_2 can be learned from view concepts constructed using different methods. In this case, X_1 and X_2 are constructed in different feature spaces, and we can consider that the weak dependence rule in equation 3.3, or the conditional independence rule in equation 3.2, are satisfied. Under these assumptions, the MV-TLC algorithm will be successful.

However, X_1 and X_2 can be constructed using the same method with different or identical parameter settings, and in these situations additional weak assumptions are required to guarantee the success of the MV-TLC algorithm. We can use the ϵ-expansion assumption, first mentioned by Balcan *et al.* [44], to analyze the success condition. Here, we provide the definition of ϵ-expansion. We assume that the examples in Algorithm 3 are drawn from a distribution D over an instance space X, and let X^+ and X^- denote the positive and negative regions of X respectively. For $S_1 \subseteq X_1$ and $S_2 \subseteq X_2$ let $S_i (i = 1, 2)$ denote the event of an example where $\langle x_1, x_2 \rangle$ has $x_i \in S_i$. If we set S_1 and S_2 as confident sets in each view, then $Pr(S_1 \wedge S_2)$ denotes the probability mass of examples for which we are confident about both views, and $Pr(S_1 \oplus S_2)$ denotes the probability mass of examples for which we are confident about only one view. D^+ is ϵ-expanding if, for any $S_1 \subseteq X_1^+$ and $S_2 \subseteq X_2^+$, $Pr(S_1 \oplus S_2) \geq \epsilon \min[Pr(S_1 \wedge S_2), Pr(\overline{S_1} \wedge \overline{S_2})]$.

Another slightly more powerful type of expansion, known as 'left-right expansion', can be defined as: D^+ is ϵ-right-expanding if, for any $S_1 \subseteq X_1^+$ and $S_2 \subseteq X_2^+$, $if\ Pr(S_1) \leq 0.5$ *and* $Pr(S_2|S_1) \geq 1 - \epsilon$ *then* $Pr(S_2) \geq (1 + \epsilon) Pr(S_1)$.

If the MI problem is under the *standard MI assumption*, and the learning algorithm used in each view is naturally confident about being positive and is able to learn from positive examples only, then the assumption of ϵ-expanding will be satisfied, and we can say that the distribution D^+ over positive examples is expanding.

If the MI problem is under the generalized MI assumption, many concept classes cannot be learned from positive examples only. In this situation, Wang & Zhou [102] demonstrated that when the diversity between two learners is larger than their errors, the performance of the learners can be improved by multi-view style algorithms. The difference $d(h_i, h_j)$

Table 3.1: Details of Datasets ('#'denotes 'number of').

Dataset	#bags	# attribute	# positive	# negative	instances
Elephant	200	230	100	100	1391
Fox	200	230	100	100	1320
Tiger	200	230	100	100	1220
Mutagenesis_atoms	188	10	125	63	1618
Mutagenesis_bonds	188	16	125	63	3995
MutagenesisM_chains	188	24	125	63	5349
Musk1	92	166	47	45	476
Musk2	102	166	39	63	6598

between the two classifiers h_i and h_j implies their different biases. In the meta-level of Algorithm 3, if the examples labeled by classifier h_i are useful for classifier h_j, h_i should have information that h_j does not have; in other words, h_i and h_j should have significant differences. In the MV-TLC algorithm, we chose different meta-learners to guarantee the differences between h_i and h_j.

3.3.4 Experiments

In this section, we explain our experiments to investigate and compare the proposed MV-TLC method to other popular MI learning algorithms.

Details of Datasets

The datasets used in our experiments are those employed in [14] and [45], and can be retrieved from http://www.eecs.uottawa.ca/~bwang009/. Table 3.1 shows the details of the datasets.

Experimental results

We compare MV-TLC with other MI algorithms, including MITI [14], MILR [28], DD [66], EMDD [123], MISMO (RBF kernel) [4], SimpleMI [39], MIboost [117], MIwrapper [47] and TLC with attribute selection (TLC_AS) [110]. Random Forest [56] was chosen as the propositional learner for SimpleMI, MIboost and MIwrapper.

For MV_TLC, in level one we chose a standard decision tree method as the supervised learning method, with k-means Euclidean distance and k-means Manhattan distance as unsupervised learning methods to generate the concepts. Empirically, each clustering method groups the instances into 20 groups.

In level two, LibSVM (RBF kernel) [21], LibSVM (linear kernel) and Random Forest [56] were chosen as the views learners, and LibSVM (RBF kernel) was chosen as the propositional learner for meta-instances.

We choose both Accuracy and AUC as the measures for our algorithms and experiments. Tables 3.2 and 3.3 present the experimental results using accuracy and AUC separately. "M" in "M_atoms", "M_bonds" and "M_chains" denotes "Mutagenesis".

Table 3.2: Experiment results (Accuracy by percent)

	Classifier	Elephant	Fox	Tiger	M_atoms	M_bonds	M_chains	Musk1	Musk2
1	MITI	77.7±1.0	61.4±3.7	75.9±2.0	80.4±2.8	79.6±2.6	82.0±1.0	70.7±2.8	70.0±1.1
2	MILR	75.4±1.4	60.2±5.0	75.3±2.9	72.0±1.0	74.6±2.2	76.2±0.8	70.9±1.6	76.5±2.1
3	DD	75.7±1.4	65.4±1.6	66.5±2.7	71.6±0.5	71.6±1.5	75.2±1.9	77.2±4.1	76.9±2.5
4	EMDD	74.3±3.4	62.2±0.4	72.5±1.5	72.2±1.2	72.1±2.6	69.7±4.4	82.2±2.7	84.7±2.0
5	MISMO(RBF)	82.3±2.0	55.1±3.3	81.7±0.8	68.3±0.7	81.5±1.5	83.5±2.0	86.3±2.0	82.5±1.8
6	Simple-MI(RF)	79.4±1.9	59.5±4.5	77.3±2.0	79.3±2.4	85.2±2.8	82.4±1.1	77.6±3.6	78.0±3.4
7	Simple-MI(SVM)	84.0±1.1	57.6±1.1	79.7±1.4	68.0±0.2	69.0±1.7	75.1±1.2	51.1±0.0	61.8±0.0
8	MI-wrapper(RF)	85.7±1.0	62.8±2.4	81.7±1.2	82.6±1.8	80.2±2.0	82.9±2.0	85.0±3.0	78.2±1.6
9	MI-wrapper(SVM)	82.5±0.9	60.4±1.3	78.0±1.1	66.5±0.0	66.8±0.3	67.0±0.0	49.1±0.5	61.8±0.0
10	MIboost(RF)	84.5±1.1	60.6±1.8	79.1±2.1	80.6±1.4	80.0±0.9	81.7±1.1	85.0±4.4	78.4±1.6
11	MIboost(SVM)	83.1±1.6	60.5±1.7	78.4±1.1	66.5±0.0	66.9±0.2	67.0±0.0	48.9±0.0	61.8±0.0
12	TLC_AS	81.5±3.1	63.5±3.4	75.0±2.2	78.2±1.6	82.4±0.7	83.8±0.6	84.8±1.9	81.4±1.7
13	MV-TLC	82.5±2.7	68.0±1.5	79.0±1.9	79.7±2.1	86.2±1.2	85.2±0.8	84.8±2.9	83.3±1.1

Table 3.3: Experiment results (AUC)

	Classifier	Elephant	Fox	Tiger	M_atoms	M_bonds	M_chains	Musk1	Musk2
1	MITI	77.5±1.3	61.3±2.7	75.7±1.8	73.5±3.7	72.3±3.4	74.4±1.7	70.4±2.7	71.7±1.4
2	MILR	81.8±3.5	56.1±5.4	79.9±2.7	77.9±0.5	80.6±3.7	80.8±2.7	74.3±1.5	83.2±1.5
3	DD	83.8±1.0	69.3±0.7	72.2±4.6	75.7±0.4	76.0±0.6	81.5±0.8	83.0±7.0	76.9±2.5
4	EMDD	80.4±4.5	63.2±2.0	76.3±1.6	64.6±3.7	73.3±6.6	72.8±5.7	85.7±6.3	90.6±3.2
5	MISMO(RBF)	82.3±2.0	55.1±3.3	81.7±0.8	60.0±1.3	79.9±2.2	81.3±2.7	86.2±2.0	82.4±2.3
6	Simple-MI(RF)	88.1±2.2	63.2±3.6	85.0±2.0	83.6±1.8	90.5±1.8	85.1±0.9	86.1±2.5	85.5±3.2
7	Simple-MI(SVM)	84.0±1.1	57.6±1.1	79.7±1.4	53.9±0.4	56.9±1.5	65.1±1.3	50.0±0.0	50.0±0.0
8	MI-wrapper(RF)	93.6±1.1	67.9±1.2	88.7±1.0	84.7±1.5	82.9±2.4	85.9±0.8	92.9±2.0	82.9±3.8
9	MI-wrapper(SVM)	88.4±0.9	66.1±2.7	85.5±1.0	74.9±0.0	79.9±2.1	75.8±2.6	51.9±0.6	53.9±0.0
10	MIboost(RF)	92.3±1.1	66.2±1.2	87.9±1.3	84.4±0.9	83.3±0.4	85.9±1.2	90.8±2.2	81.7±2.2
11	MIboost(SVM)	88.3±1.5	64.4±1.3	85.8±1.2	73.5±0.0	78.7±3.4	77.6±0.8	53.4±0.0	55.0±0.0
12	TLC_AS	81.4±2.3	62.0±2.1	77.0±1.4	76.7±1.2	83.9±2.0	84.9±1.8	83.8±2.3	82.3±1.4
13	MV-TLC	91.4±1.7	69.0±1.5	81.9±1.9	86.4±1.3	91.2±0.6	89.8±0.9	90.9±2.4	87.2±2.3

We applied a statistical test method–Nemenyi's post-hoc test [60] to determine which classifier had the best performance. First, we ranked the evaluation values for each dataset with different classifiers, and the sum of the ranks for all datasets is represented by $R_{.i}$, where i represents a classifier. Then we used the following formula to calculate the q value between different classifiers:

$$q_{ij} = (\overline{R_i} - \overline{R_j}) / \sqrt{\frac{k(k+1)}{6n}} \tag{3.4}$$

where k is the number of classifiers and n is the number of datasets. We then determined if one algorithm is better than another by comparing their q values with the critical value q_α. The result of 5-5-2 for MITI shows that the algorithm wins five times, loses five and equals two times. If we set the scores as win=1, equal=0 and lose=-1, the total score of each algorithm in the tables can be calculated. The results of this are shown in Table 3.4. The numbers in the first row denote the classifier numbers listed in Table 3.2 and 3.3.

Table 3.4: Experiment result using the statistical test method

	1	2	3	4	5	6	7	8	9	10	11	12	13
Acc	5-5-2	1-8-3	2-7-3	4-6-2	8-3-1	6-5-1	2-8-2	11-0-1	0-11-1	9-2-1	0-10-2	8-2-2	11-0-1
AUC	0-11-1	3-4-4	3-4-5	2-9-1	2-5-5	9-3-0	0-11-1	11-0-1	3-4-5	10-2-0	3-4-5	4-4-4	11-0-1
Score	-11	-8	-6	-9	2	7	-17	22	-12	15	-11	6	22

Tables 3.2 to 3.4 show that the performance of MV-TLC is as good as the performance of MI-wrapper with random forest and both of these are better than the other algorithms. Although MV-TLC is not designed for standard MI problems, its performance is comparable to other algorithms that achieve the best results.

3.3.5 Discussion

Data with complex structures, such as MI data, is usually difficult to learn with traditional machine learning paradigms. Constructive induction is a general approach to address inadequate features found in original data. Using this strategy, TLC [110] and CCE [126] demonstrate good performance compared to other popular MI algorithms. However, the fact that TLC only generates the concepts in one instance space can make the constructed features not adequate enough to be learned. Although CCE employs clustering to impose different structures, and uses the power of ensemble learning rather than single classifier, its performance on count-based MI is as good as that of TLC with AS [110] or MI kernel [126]. Zhou & Zhang [126] found that this could be because the binary feature vectors used by CCE are not sufficient to represent the exact number of instances in a cluster. To use the power of ensemble learning, Dietterich [37] indicated that an effective paradigm for generating diverse classifiers is required. The feature vector diversity generated by clustering the instances into different number of groups may not be strong enough for the ensemble learning method.

The empirical study shows that by using a multi-view style algorithm, and generating multi-view concepts with different structures in different instance spaces, MV-TLC can achieve higher and more stable performance compared to TLC [110] and CCE [126].

Under the collective assumption that instances contribute equally and independently to bag-level class labels, MI-wrapper with Random Forest showed great performance. However, for some applications, this assumption might not always be suitable. In data with more complex construction, instances could be dependent on bag-level class labels, and in such cases the generalized MI assumption utilized by MV-TLC may be more appropriate. On the other hand, we found that for some data, both MI-wrapper and MIboost are sensitive to propositional learners. In our empirical study, when choosing SVM with RBF kernel as the propositional learner, MI-wrapper and MIboost's performance was low. In contrast, MV-TLC can employ the power of multi-view learning and is not sensitive to propositional learners.

3.4 Conclusions and future research

In this chapter, we proposed a framework that demonstrates a solution for MI learning by adapting MI representation to single-instance algorithms. The proposed algorithm, known as MV-TLC, employs processes that include supervised and unsupervised learning methods to help construct new multi-view meta-features that can be exploited by common supervised learning algorithms. MV-TLC also utilizes the power of multi-view learning paradigms to achieve strong generalization ability. Experiments show that MV-TLC works well with both standard and generalized MI problems, without requiring any modifications.

There are many potential ways to modify the MI representation. Exploring other schemes for adapting to single-instance algorithms would be an interesting future direction. In addition, the success of MV-TLC reveals that other multi-view methods can be considered when learning data with complex structures. Applying these techniques would be another promising area for further investigation.

Chapter 4

The Data fusion framework for the MI Problem

In this chapter, we will introduce the fusion assumption. This is a new MI assumption that differs from other assumptions we discussed in chapter 2.

4.1 Introduction

Data fusion is a technology that collates information from different sources in order to improve accuracy and to make better inferences than which could be achieved by the use of a single classifier or data set alone. While the coverage of methodological areas of data fusion systems includes artificial intelligence, pattern recognition, and statistical inference, application areas of data fusion are widespread.

In data fusion, modeling procedures consist of association, estimation, and identity declaration. Association determines which pairs of observations belong together, by representing observations of the same entity. Commonly used association measures include correlation coefficients, distance measures, association coefficients, and probabilistic similarity measures. In terms of estimation, parameters of the fusion model are estimated using maximum likelihood estimator, least square estimator, or Kalman filter estimator. For identity declaration, typically the information extracted from the multiple sources can be fused at different fusion levels and with different methods [52]. The fusion levels are:

- The data fusion (low-level fusion) is the fusion of information directly at the output
- The fusion of characteristics (mid-level fusion) is the fusion of information extracted from data obtained directly from the output.
- The decision fusion (high-level fusion) is the merging of information corresponding to the formulation of hypotheses derived from an expert or a system (e.g. a classifier).

In the data-level fusion, different data sources are fused directly. For instance, in imagery pattern recognition, data-level fusion is often referred to as pixel level fusion [7].

Feature-level fusion, or state-level fusion, involves the extraction of representative features from data. In this level, features are extracted from multiple data sources and are combined into a single feature vector which is used as an input for pattern recognition approaches based on base learners.

Decision-level fusion involves fusion of classifiers. Examples of decision level fusion methods include classical approaches such as Dempster-Shafer (DS) algorithm [88] and logistic fusion, along with weighted decision methods (ensemble or voting techniques).

Before we get into the discussion of the proposed approach for MI learning, we review the application literature on data fusion in following section.

4.2 Literature review

The most common numerical fusion techniques used are Bayesian probability theory, Fuzzy systems and Dempster-Shafer (DS) theory [88]. Fuzzy systems contain a wealth of possible fusion operators. However, many of the operators are non-associative and the choice of operators is case dependent, which means that the order in which the information is fused has an impact on the final result. Bayesian and DS models have both been successfully applied . But DS theory provides some features that Bayesian theory does not. One of the most significant features is that DS theory can consider the union of classes. This feature is used to improve the separability of different classes. As a result, the DS method is a popular data fusion method that has been widely used [7] [121].

Buede & Girardi [20] demonstrated how Bayesian and DS algorithms can address the same target identification problem involving multiple levels of abstraction, such as identification based on type, class, and nature. In the process of demonstrating target identification with these two reasoning methods, the authors compared their convergence time to a long run asymptote for a broad range of aircraft identification scenarios that include missing reports and mis-associated reports.

J. Fawcett et al. [43] investigated two approaches for fusing multiple views: fuse-feature and fuse-classification. In the first approach, the two feature sets taken at different aspects were combined to form a large feature vector. Then a kernel based classifier was trained with this feature vector. In the second approach, two individual-aspect classifications of two feature vectors were fused using the DS theory [88].

S. Reed et al. [83][84] have also investigated the classication of a target by fusing several views using DS theory [88]. They present a model to extend the standard mine/not-mine classication procedure to provide information on the shape and size of the object. The difference between their work and the work of others is that they generated the mass functions using a fuzzy functions membership algorithm based on fuzzy logic.

Decision level fusion methods include weighted decision methods such as ensemble or voting techniques. Data ensemble combines various results obtained from a single classifier

fitted repeatedly based on bootstrap resamples. Such algorithms can be divided into two types: those that adaptively change the distribution of the bootstrap training set based on the performance of previous classifiers, as in Boosting methods, and those that do not, as in Bagging. Military application includes automated target recognition, guidance for autonomous vehicles, remote sensing, battlefield surveillance, and automated threat recognition systems [43] [83] [84]. Nonmilitary conventional applications include monitoring of manufacturing processes, condition-based maintenance of complex machinery, robotics, and medical pattern recognition [20] [52].

4.3 Fusion assumption and the MI fusion framework for MI learning

The fusion assumption is motivated by a view of the nature of MI bags that is based on the Bayesian theory of subjective probability. Under this view, a bag is not a finite collection of fixed elements, but instead is a sample of an underlying population specific to that particular bag. A bag can be treated as a group of different information sources with different degrees of reliability. To account for this, we associate "instance reliability" with each instance so that a less reliable instance has a smaller effect on the global fusion of local decisions made by each instance. These local decisions are fused using the Dempster's rule [88] to get the final decision. The diagram in Figure 4.1 shows this high level fusion processing flow for classification/fusion using MI data.

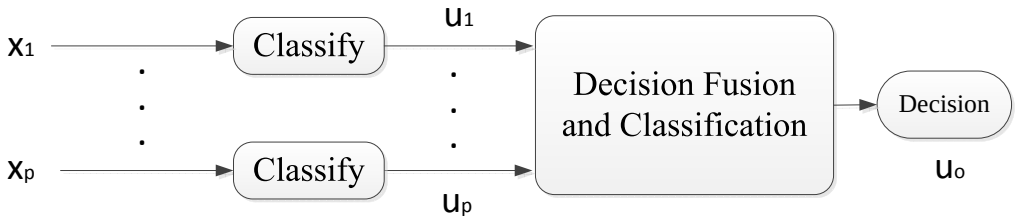

Figure 4.1: the processing flow for classification/fusion using multiple views

The DS method [88] is based on two ideas: obtaining degrees of belief for one question from subjective probabilities for a related question, and Dempster's rule for combining such degrees of belief when they are based on independent items of evidence.

The Dempster's rule [88] of combination is a purely conjunctive operation (AND). The combination rule results in a belief function based on conjunctive pooled evidence and it can also be used for multi aspect classification.

In DS theory, each unique class makes up a set called the frame of discernment $\Theta = \{\omega_1, \omega_2, \cdots, \omega_M\}$. Belief is attributed to hypotheses within the power set through a basic probability assignment called the mass function $m(A)$ for any collection $A_1, A_2, \cdots, A_n$, of

subset of Θ. A function $Bel : 2^{\Theta} \longrightarrow [0, 1]$, is a belief function if it satisfies $Bel(\Theta) = 0$, and for any collection $A_1, A_2, \cdots, A_n$, of subset of Θ,

$$Bel(A_1 \cup \cdots \cup A_n) \geq \sum_{I \subseteq 1, \cdots, n, I \neq \emptyset} (-1)^{|I|+1} Bel(\bigcap_{i \in I} A_i) \tag{4.1}$$

A belief function assigns to each subset of Θ a measure of the total belief in the proposition represented by the subset. The notation, $\mid I \mid$ is the cardinality of the set I. Suppose that we have two views of target S_1 and S_2 and the mass functions $m_1(S_1)$ and $m_2(S_2)$. Based on the Dempster's rule, the mass after fusion for the set A is:

$$m_{12} = K \frac{\sum_{S_1 \cup S_2 = A} m_1(S_1) m_2(S_2)}{1 - \sum_{S_1 \cup S_2 = \emptyset} m_1(S_1) m_2(S_2)} \tag{4.2}$$

Here K is the constant of proportionality which normalized the result to compensate for the measure of belief committed to $\emptyset$.

The classification rule for this case is

$$g(S_1, S_2) = \text{argmax}_{1 \leq i \leq n} m_{12}(\omega_i) \tag{4.3}$$

Using the graphical interpretation, it is straightforward to write down the formula for the orthogonal sum of more than two belief functions. Let $m = m_1 \oplus \cdots \oplus m_n$, then $m(\emptyset) = 0$ and

$$m(A) = K \sum_{\cap A_i = A} \prod_{1 \leq i \leq n} m_i(A_i) \tag{4.4}$$

$$K^{-1} = 1 - \sum_{\cap A_i = \emptyset} \prod_{1 \leq i \leq n} m_i(A_i) = \sum_{\cap A_i \neq \emptyset} \prod_{1 \leq i \leq n} m_i(A_i) \tag{4.5}$$

Similar to many authors, we use DS theory as a choice for MI classification in our research. In our algorithm, we use a training dataset for the single-aspect classifier and then save the predicted class labels from the testing data. Using T cross validation we can get a $T \times M$ output matrix. Let $\beta_i(k), k = 1, 2, \cdots, T$ which corresponds to the ith column of the prediction vector for the kth testing feature vector.

For n output vectors $\beta_i(k), i = 1, 2, \cdots, n$ obtained from n single-aspect classifications, the n sets of masses are finally fused using Dempster's rule and the final decision is given by the classification rule $g(x_1, x_2, \cdots, x_n)$ [10].

Algorithm 4 gives the details of the MI_fusion algorithm.

4.4 Experiments

In this section, we explain our experiments to investigate and compare the proposed MI-DS method to other popular MI learning algorithms.

Algorithm 4 the MI_fusion algorithm

1: Given: A multi-instance training dataset with a set of bags $\chi_i, i = 1, \cdots, N$, where each bag can consist of an arbitrary number of instances and a given label:$\chi_i = \{x_i^1, x_i^2, \cdots, x_i^{n_i}; Y_i\}, i = 1, \cdots, N, Y_i \in \{-1, +1\}$, and each instance $x_i^{n_i}$ is an M-tuple of attribute values belonging to a certain domain or instance space $\mathbb{R}$.
2: C = all instances $x_i^{n_i}, i = 1, \cdots, N$ in the bags of χ
3: L = single instance classifier
4: int j=0
5: for all $C_j \in \chi$ do C_j.setClassValue(χ_i)
6: end for
7: L.train(C)
8: Save the predicted class labels from the testing data. Using T cross validation to get a $T \times M$ output matrix.
9: The mass after fusion for the set A is: $m(A) = K \sum_{\cap A_i = A} \prod_{1 \le i \le n} m_i(A_i)$
10: Get the final decision given by the classification rule $g(x_1, x_2, \cdots, x_n)$ using Dempster's rule $g(x_1, x_2, \cdots, x_n) = \text{argmax}_{1 \le i \le n} m_{1,2,\cdots,n}(\omega_i)$

4.4.1 Experimental results on Accuracy and AUC

The datasets used in our experiments are those employed in chapter 3 where Table 3.1 shows the details of the datasets. We compare MI_fusion with other MI algorithms, including MITI [14], MILR [117], DD [73], EMDD [123], MISMO (RBF kernel) [4], MIoptimalBall [6].

For MI_fusion, Random Forest [56] was chosen as the propositional learner. We choose both Accuracy and AUC as the measures for our algorithms and experiments. Tables 4.1 and 4.2 present the experimental results using accuracy and AUC separately. "M" in "M_atoms", "M_bonds" and "M_chains" denotes "Mutagenesis".

Table 4.1: Experimental results (Accuracy by percent)

	Classifier	Elephant	Fox	Tiger	M_atoms	M_bonds	M_chains	Musk1	Musk2
1	MITI	77.7±1.0	61.4±3.7	75.9±2.0	80.4±2.8	79.6±2.6	82.0±1.0	70.7±2.8	70.0±1.1
2	MILR	75.4±1.4	60.2±5.0	75.3±2.9	72.0±1.0	74.6±2.2	76.2±0.8	70.9±1.6	76.5±2.1
3	DD	75.7±1.4	65.4±1.6	66.5±2.7	71.6±0.5	71.6±1.5	75.2±1.9	77.2±4.1	76.9±2.5
4	EMDD	74.3±3.4	62.2±0.4	72.5±1.5	72.2±1.2	72.1±2.6	69.7±4.4	82.2±2.7	84.7±2.0
5	MISMO(RBF)	82.3±2.0	55.1±3.3	81.7±0.8	68.3±0.7	81.5±1.5	83.5±2.0	86.3±2.0	82.5±1.8
6	MIoptimalBall	73.5±2.3	50.5±1.8	66.0±2.5	72.3±1.9	73.4±1.4	72.3±2.2	70.7±1.7	79.4±1.2
7	MI_fusion(RF)	81.5±0.9	64.2±2.6	79.0±2.4	79.5±1.4	80.7±2.2	80.0±1.3	83.9±1.4	78.2±1.8

Here we applied Nemenyi's post-hoc test [60] to determine which classifier had the best performance. If we set the scores as win=1, equal=0 and lose=-1, the total score of each algorithm in the tables can be calculated. The results of this are shown in Table 4.3.

Tables 4.1, 4.2 and 4.3 show that the performance of MI_fusion is competitive. Although MI_fusion is not designed for standard MI problems, its performance is comparable to other algorithms which achieve the best results.

Table 4.2: Experiment results (Experiment results (AUC))

	Classifier	Elephant	Fox	Tiger	M_atoms	M_bonds	M_chains	Musk1	Musk2
1	MITI	77.5±1.3	61.3±2.7	75.7±1.8	73.5±3.7	72.3±3.4	74.4±1.7	70.4±2.7	71.7±1.4
2	MILR	81.8±3.5	56.1±5.4	79.9±2.7	77.9±0.5	80.6±3.7	80.8±2.7	74.3±1.5	83.2±1.5
3	DD	83.8±1.0	69.3±0.7	72.2±4.6	75.7±0.4	76.0±0.6	81.5±0.8	83.0±7.0	76.9±2.5
4	EMDD	80.4±4.5	63.2±2.0	76.3±1.6	64.6±3.7	73.3±6.6	72.8±5.7	85.7±6.3	90.6±3.2
5	MISMO(RBF)	82.3±2.0	55.1±3.3	81.7±0.8	60.0±1.3	79.9±2.2	81.3±2.7	86.2±2.0	82.4±2.3
6	MIoptimalBall	73.5±4.2	50.5±1.7	66.0±3.6	69.0±1.8	70.9±4.8	67.8±3.9	70.6±4.5	75.5±2.5
7	MI_fusion (RF)	89.9±1.6	66.5±4.1	85.7±1.8	77.5±1.4	80.5±1.8	78.4±1.8	92.4±2.5	82.9±2.4

Table 4.3: Experiment result using the statistical test method

	MITI	MILR	DD	EMDD	MISMO(RBF)	MIoptimalBall	MI_fusion(RF)
Accuracy	4-2-0	1-3-2	1-3-2	1-3-2	5-0-1	0-6-0	5-0-1
AUC	1-5-0	4-1-0	3-1-2	2-4-0	3-2-0	0-6-0	6-0-0
Score	-2	1	0	-4	6	-12	11

4.4.2 Experimental results on running time

In this experiment, we compare the running time of MI_fusion with other classifiers on dataset Musk2. We report running time results (Intel Core i3-4130 @ 3.40GHz processor, 3.75G RAM) in Table 4.4.

Table 4.4: Experiment results (Running Time)

	Classifier	Time(ms)
1	MITI	3765
2	MILR	45528
3	DD	3898139
4	EMDD	307873
5	MISMO(RBF)	93213
6	CitationKNN	159018
7	MIoptimalBall	47513
8	MINND	15467
9	MI_fusion(RF)	3015

From the experimental results on running time we can find that MI_fusion using random forest is faster than other algorithms while its accuracy is competitive.

4.5 Analysis and Conclusion

The MI Fusion assumption is designed as a general alternative to the standard MI assumption. Similar to the collective assumption, the MI fusion assumption is based on probability theory. However, unlike the collective assumption, The MI Fusion assumption does not assume that the population of a bag is independent of each other. Experimental result on

performance shows that the MI_fusion approach under this assumption is competitive in contrast with other MI approaches. Moreover, Experimental results on the running time of training show that its computational complexity is lower than that of many popular MI approaches.

Chapter 5

The Multi-kernel framework for MI Problem

5.1 Introduction

In chapter 3, we introduced a multi-view, two-level classification (MV-TLC) strategy to transform a MI dataset into a multi-view, single meta-instance dataset, and we learned from multiple views (the feature set) of this meta-dataset. In this chapter we will extend this topic using another multi-view strategy — the Multiple Kernel learning approach on the MI learning problem.

As mentioned in chapter 2, kernel methods have been used in MI learning widely such as mi-SVM or MI-SVM [4] and have been proved to be successful MI learning approaches. However, the main drawback of kernel methods still exists in MI learning, which is that they require the user to specify a single suitable kernel at the beginning. This is usually a hard task even when the user is familiar with the problem domain and is often critical to the kernel method's success.

To ease this burden, significant attention has been paid to the problem of automatically learning the kernel. Methods such as the grid search [57] and the random search [12] have been provided recently. Moreover, the majority of the previous work in this area has focused on the Multiple Kernel Learning (MKL) [49] setting, where the user only needs to specify a set of base kernels, and the learning algorithm is responsible to find a combination of these base kernels that is appropriate for the learning problem.

In this chapter, we adapt the strategy of MKL to the MI learning problem which can eliminate the drawback of single kernel methods mentioned above. We present a Multiple-Kernel MI-SVM framework based on this idea. Section 5.2 gives a literature review of the background and related work; Section 5.3 gives the detail of the Multiple-Kernel MI-SVM framework; section 5.4 is the experimental part followed by the conclusion in section 5.5.

5.2 Background and related work

Recent developments on SVMs and other kernel methods have shown the need to consider multiple kernels. This provides flexibility and reflects the fact that typical learning problems often involve multiple, heterogeneous data sources.

The reasoning is similar to combining different classifiers. Instead of choosing a single kernel function, it is better to have a set and allow the algorithm to do the picking or combination step. Multiple Kernel Learning (MKL) was originally developed to control the search space capacity of the possible kernel matrices to achieve good generalization. It has also been widely applied to problems involving multi-view data. This is because kernels in MKL naturally correspond to different views and combining kernels either linearly or non-linearly improves learning performance.

MKL can be useful in two aspects: (1) Since a kernel plays the role of defining the similarity between instances, different kernels correspond to different notions of similarity, and using a specific kernel may be a source of bias. To avoid this, we can apply a learning method to pick the best kernel for us or use a combination of a kernel set. In allowing a learner to choose among a set of kernels, a better solution can be found. (2) Different kernels may be using inputs coming from different representations. Since there may be different measures of similarity corresponding to different kernels, combining kernels is a possible way to combine multiple information sources.

The key idea of MKL is to learn a linear combination of a given set of base kernels by maximizing the margin between the two classes or by maximizing kernel alignment. We can think of a kernel combination as a weighted average of kernels and consider the weight $\beta \in \mathbb{R}_+^P$ and $\sum_{m=1}^P \beta_m = 1$, where P denotes the number of weights. Suppose one is given n $m \times m$ symmetric kernel matrices $K_j, j = 1, \cdots, n$, and m class labels $y_i \in \{1, -1\}, i = 1, \cdots, m$. A linear combination of the n kernels under an l_1 norm constraint is considered:

$$K = \sum_{j=1}^{n} \beta_j K_j, \beta \geq \emptyset, \|\beta\|_1 = 1 \tag{5.1}$$

where $\beta = (\beta_1, \cdots, \beta_n)^T \in \mathbb{R}^n$, and $\emptyset$ is the n dimensional vector of zeros. Geometrically, different scaling of the feature spaces lead to different embeddings of the data in the composite feature space. Then the goal of MKL is to learn the optimal scaling of the feature spaces, with the so called "separability" of the two classes in the composite feature space maximized.

In kernel methods, the choice of a kernel function is critical, since it completely determines the embedding of the data in the feature space. Ideally, this embedding should be learnt from the training data. In practice, a relaxed version of this very challenging problem is often considered: given multiple kernels capturing different "views" of the problem, how does one learn an "optimal" combination of them?

Lanckriet et al. [67] have proposed to use the soft margin of SVM as a measure of separability, that is, to learn the weight β by maximising the soft margin between the two classes. Bach et al. [8] have reformulated the problem and then proposed a SMO algorithm

for medium-scale problem. Cortes et al. [32] discuss the suitability of the 2-norm for MKL. In their paper they conclude that using the 1-norm improves the performance for a small number of kernels, but not for a large number of kernels. Meanwhile, the 2-norm increases the performance significantly for larger sets of candidate kernels and never decreases it. Sonnenburg et al. [90] developed an efficient semi-infinite linear program and made MKL applicable to large scale problems. Rakotomamonjy et al. [80] proposed simple MKL by exploring an adaptive 2-norm regularization formulation. Gönen & Alpaydn [50] have reviewed the literature on MKL.

The performance improvement of MKL comes at a price. Learning the entire set of models and then combining their predictions is computationally more expensive than learning just one simple model. The computational complexity of MKL is very high for two major reasons: (1) Similar to normal kernel based methods, MKL needs to compute kernel functions for each sample-pair over the training set; (2) MKL needs to optimize the classifier parameters and kernel weights in an alternative manner, thus learning global optimal parameters would incur intensive computation. More specifically, MKL that use optimization approaches to learn combination parameters have high computational complexity, since they are generally modeled as a semi definite programming (SDP) problem, a quadratically constrained quadratic programming (QCQP) problem, or a second-order cone programming (SOCP) problem. MKL can also be modeled as a semi-infinite linear programming (SILP) [8] problem, which uses a generic linear programming (LP) solver and a canonical SVM solver in the inner loop. This method is more efficient than previous methods but the computational complexity is still very high.

In recent years, a lot of efforts have been made on reducing the computational complexity of the SVM algorithm. For MKL, Chen et al. [28] have proposed a method by dividing the global problem with multiple kernels into multiple local problems, each of which optimize in a local processor with a single kernel.

5.3 Proposed approach

Multiple kernel SVM is used for some situations where a machine learning practitioner may be interested in more flexible models. In this thesis, we present a multiple kernel framework for the MI problem. We can expect MKL to have a better performance in MI learning problem for two reasons. (1) Obviously, because the highly complicated description of real objects in MI learning cause a special problem, a flexible model is necessary for the learning task. (2) Due to the enhanced interpretability of the decision function, more effectible computation and higher predication accuracy are expected in MI learning.

Therefore, it is significant to add multiple kernel methods to the MI learning problem. In this section, the model and algorithm of Multiple-Kernel MI-SVM will be given. In Multiple-Kernel MI-SVM method, we also defined the functional margin of a bag with respect to a hyperplane by $y_i = Y_I \max_{i \in I}(\langle W, X_i \rangle + b)$. Based on this rule, the inequality constraints in multiple kernel SVM can be changed for solving the MI learning problem. Therefore the Multiple-Kernel MI-SVM model can be expressed as a new optimization

problem shown in the following:

$$\min_{\{f_m\},b,\xi,d} \frac{1}{2}\sum_m \frac{1}{d_m}\|f_m\|^2_{H_m} + C\sum_I \xi_I \tag{5.2}$$

Subject to:

$$Y_I \max_{i\in I}(\sum_m f_m(x_i) + b) \geq 1 - \xi_I \quad \forall I$$

$$\xi_I \geq 0 \ \sum_m d_m = 1, d_m \geq 0 \quad \forall m$$

Since the first constraint in our MI formulation contains the max operations, we unfolded this max operation as [3]. For negative bags, the inequality constraint can be read as $-\langle w, x_i\rangle - b \geq 1 - \xi_I, \forall i \in I$, where $Y_I = -1$. For positive bags, a selector variable $s(I) \in I$ which denotes the instance selected as the positive instance in per positive bag B_I will be given. For d_m, $s(I) \in I$ and α, b, alternately compute one set of variables and hold other sets. This leads to the following Multiple-Kernel MI-SVM (MKMI) algorithm:

Algorithm 5 Multiple Kernel MI-SVM algorithm

1: Given: A multi-instance training dataset with a set of bags $\chi_i, i = 1, \cdots, N$, where each bag can consist of an arbitrary number of instances and a given label:$\chi_i = \{x_i^1, x_i^2, \cdots, x_i^{n_i}; Y_i\}, i = 1, \cdots, N, Y_i \in \{-1, +1\}$, and each instance $x_i^{n_i}$ is an M-tuple of attribute values belonging to a certain domain or instance space $\mathbb{R}$.
2: Initialize $y_i = Y_i \ for \ i \in I$
3: Compute $K = \sum_{m=1}^{M} d_m K_m(x_n, x_i), \alpha, b$ for data set with imputed labels
4: Compute $f_i = \sum_{n=1}^{l} \alpha_i K + b$ for all x_i in positive bags
5: For every positive bag B_I do
6: If $(\sum_{i\in I}(1 + y_i)/2) = 0)$
7: Compute $i' = \arg\max_{i\in I} f_i$
8: Set $f_{i'} = 1$
9: End if
10: End for
11: While (imputed labels have changed)
12: Output (d_m, α, b)

Many research works have reported that when using the Multi-kernel SVM classifier, higher accuracy of classification can be expected. However, most of these research works are about single instance learning. For MI learning, we need experimental results to support our hypothesis that Multi-kernel SVM can also be used for successfully solving MI problems.

5.4 Experiments

In this section, we explain our experiments to investigate and compare the proposed Multiple Kernel MI-SVM (MKMI) method to other popular MI learning algorithms.

5.4.1 Experimental setup and results

In our experiments we use the same datasets employed in chapter 3. We compare MKMI with benchmark algorithms: SVM with different kernels. Two popular kernels (RBF and linear) are chosen in this experiment and the hyper-parameters of SVMs are set using the Cross-Validation and Grid-Search methods [57]. For MKMI, MI_fusion is chosen for MI part and 30 kernels are chosen for MKL. Table 5.1 gives the detail of these kernels. The format of these kernels is kernel_name (gamma, cost). For all kernels the epsilon is 1e-5, and the coef0 is 0.

Table 5.1: Kernels details

1)Gaussian(2.1,1)	2)Polynomial(1)	3)Sigmoid(0.1)
4)Exponential(10,10)	5)Spherical(10)	6)Gaussian(20,1)
7)Circular(1)	8)Gaussian(100,10)	9)InverseMultiQuadric(1)
10)Gaussian(10,10)	11)T-Student(1)	12)Gaussian(1,1)
13)Linear	14)Spline	15)Chi-square(1)
16)RationalQuadratic(10)	17)Polynomial(10)	18)HistogramIntersection
19)ANOVA(1)	20)Distance(1)	21)Spherical(1)
22)Wavelet(1,1)	23)Sigmoid(0.01)	24)Polynomial(0.1)
25)Polynomial(2)	26)Gaussian(10,1)	27)Cauchy(1)
28)RationalQuadratic(1)	29)T-Student(0.1)	30)InverseMultiQuadric(10)

We choose both the Accuracy and the AUC as the measures for our algorithms and experiments. Tables 5.2 presents the experimental results using accuracy and AUC separately.

We applied Nemenyi's post-hoc test [60] to determine which classifier had the best performance. The results of this are shown in Table 5.3.

Table 5.2 and 5.3 show that the performance of MKMI is better than the algorithms using single kernels.

From chapter 3 to chapter 5, we have presented 3 different frameworks for MI learning. In table A.3 and A.4 we listed the results of all experiments applying these algorithms on the benchmark datasets listed in Table 3.1. Combining the results listed in Table A.3 and A.4, we can find that MV_TLC gives the best performance comparing to other algorithms on the benchmark datasets.

5.5 Conclusion

In this chapter, we formulated a novel method to solve the classification problem within the MI learning context by multiple kernel learning. To improve the classification precision of the SVM method regarding the MI problem, this chapter introduced the multiple kernel learning method to the process of multiple instance learning. A new SVM model (MKMI) is also proposed, which is based on the two-level MI model. The solution for this model was

Table 5.2: The Experiment Results

Dataset	Kernel Type	Accuracy		AUC	
		SVM	MKMI	SVM	MKMI
Elephant	RBF	85.4±0.7	84.7±0.4	85.4±0.3	84.5±0.7
	Linear	82.6±1.4		83.6±1.3	
	Sigmoid	84.1±1.0		84.2±0.6	
	Polynomial	83.1±0.6		83.5±0.4	
Tiger	RBF	79.3±1.7	82.9±0.4	79.3±1.7	82.8±0.6
	Linear	80.6±1.6		80.3±1.3	
	Sigmoid	80.2±1.2		79.8±0.6	
	Polynomial	81.0±1.1		81.2±0.8	
Fox	RBF	61.7±1.3	57.8±2.1	61.7±1.3	57.8±2.3
	Linear	54.6±1.5		53.8±1.2	
	Sigmoid	54.5±1.1		54.6±1.3	
	Polynomial	61.5±0.7		61.4±0.6	
M_atoms	RBF	78.7±0.7	79.2±0.9	72.6±0.4	74.6±1.0
	Linear	73.4±0.9		66.9±0.7	
	Sigmoid	53.4±0.3		49.8±0.2	
	Polynomial	62.6±2.1		63.0±1.8	
M_bonds	RBF	78.2±0.5	85.2±1.9	75.3±1.3	84.3±0.5
	Linear	79.5±0.8		76.1±1.0	
	Sigmoid	63.3±1.2		64.7±1.1	
	Polynomial	79.3±0.7		80.1±1.2	
M_chains	RBF	83.0±0.6	83.0±0.8	79.7±0.5	80.1±0.8
	Linear	83.0±1.1		82.1±0.9	
	Sigmoid	63.3±1.2		65.7±1.4	
	Polynomial	80.3±1.5		78.9±1.6	
Musk1	RBF	51.1±0.0	82.4±2.4	50.0±0.0	82.3±2.5
	Linear	79.3±2.7		79.2±2.7	
	Sigmoid	51.1±0.0		51.3±0.0	
	Polynomial	81.3±1.9		81.2±2.0	
Musk2	RBF	61.8±0.6	81.4±2.6	50.5±0.5	82.3±2.1
	Linear	80.0±0.5		80.9±1.1	
	Sigmoid	49.0±0.6		51.4±1.0	
	Polynomial	80.4±2.3		81.2±2.5	

Table 5.3: Experiment result using the statistical test method

	RBF	Linear	Sigmoid	Polynomial	MKMI
Acc	1-1-2	1-1-2	0-4-0	1-1-2	4-0-0
Acc	1-1-2	1-1-2	0-4-0	2-1-1	4-0-0
Score	-1	0	-8	-1	8

presented and some numerical experiments on benchmark data were performed. Computational results on a number of datasets indicate that the proposed algorithm is competitive with other SVM methods.

Chapter 6

Imbalanced MI Problem

6.1 Background and Literature Review for the Class imbalance problem

6.1.1 The description of the class Imbalance problem

A single-instance dataset is imbalanced if its classes are not represented approximately equally. In imbalanced datasets, the number of examples in one class is often much higher than the number in the other class. In this situation, a default classifier always predicts a new example as the "the majority class". The significant difficulties of the class imbalance problem, and its frequent occurrence in practical applications of machine learning and data mining, have attracted a lot of interest in the field. A number of research papers dedicated to this problem can be found in [55] and other publications [11][61][93].

Generally the imbalance problem in machine learning is referred to as a between-class imbalance. A within-class imbalance, which concerns itself with the distribution of representative data for sub-concepts within a class, also exists in many datasets. Standard classifiers generally perform poorly on imbalanced datasets because they are designed to generalize from training data and output a hypothesis that best fits the data to get the smallest error rate [93]. Consequently, this design pays less attention to rare cases in an imbalanced dataset when achieving the minimum error rate.

Although the degree of imbalance varies from one application to another, in most situations, the correct classification of minority samples often has a greater value than the other cases. For example, in a cancer diagnostic classification application where the cancer cases are usually quite rare as compared with normal populations, the skewed class distribution will train the classifier to identify every sample to the normal class since in this case the error rate of classification is close to zero. However, a favorable classification model is one that provides a higher identification rate on the cancer category. In this case, the imbalanced or skewed class distribution problem makes the classifier fail on this task.

Research on the class imbalance problem is critical in data mining and machine learning. Two observations account for this point: (1) the class imbalance problem is pervasive in a

large number of domains in the data mining and machine learning community. (2) Most popular classification modeling systems are reported to be inadequate when dealing with the class imbalance problem [2][9][11][55].

To provide a concrete understanding of the direct effects of the imbalanced learning problem on standard learning algorithms, He et al. [55] provides a case study of the popular decision tree learning algorithm. In this case, imbalanced data sets exploit inadequacies in the splitting criterion at each node [79]. There may be two problems with this procedure: (1) Fewer observations in the minority examples results fewer leaves. This correlates with the problem of relative and absolute imbalances. (2) Concepts that have dependencies on different feature space conjunctions correlates with the between-class imbalance. In both cases, the effects of imbalanced data on performance are detrimental. Wu & Chang [114] also provides a case study of another popular learning algorithm-SVM. In this case, a checkboard example is provided. Comparing the class boundaries with a low and a high ratio of the number of majority instances to the number of minority instances, it is obviously to be observed that the class boundary with a high ratio is much more skewed towards the minority quadrant, and hence causes a higher incidence of false majority class.

6.1.2 Solutions for the single-instance class imbalance problem

Most of the current studies are based on the topics covered in section 6.1.1. Particularly, the majority of the current solutions are designed to reduce the impact that these problems induced on standard learning algorithms. He et al. [55] made a systematic literature review on the solutions for the class imbalance problem. Based on his work, we list the solutions proposed to overcome the effects of imbalanced data.

The re-sampling methods have been used widely in order to provide a balanced distribution. Related studies have shown improved overall classification performance on a balanced data set [41][55][59][100]. For single-instance learning, re-sampling methods can be divided into under-sampling and over-sampling. For under-sampling method, Kubat and Matwin [64] proposed a technique called one-sided selection (OSS). One-sided selection attempts to intelligently under-sample the majority class by removing majority class examples that are considered either redundant or 'noisy'. For over-sampling method, the Synthetic Minority Over-sampling Technique (SMOTE) [22] was introduced as a directed over-sampling method. With SMOTE, the neighborhood of a minority instance is assumed to be the same class as the instances between two minority instances. Barandela et al. [9] proposed a modified distance calculation, which causes an example to be biased more towards being identified with positive examples than negative ones. Han et al. presented a modification of SMOTE [22] technique which they call borderline-SMOTE (BSM) [54] for single-instance learning.

Different from re-sampling methods which attempt to balance distributions by considering the representative proportions of class examples in the distribution, cost-sensitive learning methods consider the costs associated with misclassifying examples [40][97]. It targets the imbalanced learning problem by using different cost matrices that describe the costs for misclassifying any particular data example.

Except re-sampling methods and cost-sensitive learning methods, numerous other approaches have also been pursued in the community. Since kernel-based learning methods provide state-of-the-art techniques for many of today's data engineering applications, the use of kernel-based methods to deal with imbalanced problem has naturally attracted growing attention recently. In terms of SVMs, several attempts have been made to improve their class prediction accuracy [2][75][89][99][100]. We will make a systematic discussion about this topic in chapter 7.

Ensembles have been used to combine the results of several classifiers, each induced with different costs ratio. The SMOTEBoost approach [25] creates synthetic examples from the minority class, thus indirectly changes the updating weights and compensates for skewed distributions. Moreover, several cost-sensitive boosting methods for imbalanced learning have been proposed in recent years. Three cost-sensitive boosting methods, AdaC1, AdaC2 and AdaC3, were proposed in [93], which introduced cost items into the weight updating strategy of AdaBoost. AdaCost [42] is another cost-sensitive boosting algorithm that follows a similar methodology.

Besides the methods mentioned above, there are other methods that have also attracted attention in the community, such as the one-class learning or novelty detection method [24]. Raskutti & Kowalczyk [82] show that one-class learning is particularly useful on extremely imbalanced data sets with a high dimensional noisy feature space. Juszczak & Duin [62] combine one-class classifiers with a re-sampling method with the aim of adding information in to the training set, both from the target (minority) class and the outlier (majority) class. Su & Hsiao presented an evaluation of a model with the name MTS [91] for imbalanced learning with comparisons to several classification algorithms. This work showed the effectiveness of the MTS in the presence of imbalanced data.

6.1.3 Evaluation methods for the class imbalance problem

Confusion matrix: the confusion matrix, as shown in the following table, represents the typical metrics for evaluating the performance of machine learning algorithms. In the Table 6.1, TN is the number of True Negatives, FP is the number of False Positives, FN is the number of False Negatives, and TP is the number of True Positives. The FP Rate is calculated as $FP/(FP+TN)$.

Table 6.1: Confusion matrix for a two-class problem

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	TP (True Positive)	FN (False Negative)
Actual Negative Class	FP (False Positive)	TN (True Negative)

Gmean and F-measure: When learning imbalanced data, the overall classification accuracy is often not an effective measure of performance [11][55]. We chose Gmean [64] as the measure for our algorithms and experiments. The definition of Gmean is expressed in (6.1).

$$Gmean = (TN/(TN+FP) \times TP/(TP+FN))^{1/2} \tag{6.1}$$

Here $TP/(TP+FN)$ is defined as true positive rate (TPR) (or Sensitivity [64]), and $TN/(TN+FP)$ is defined as true negative rate (TNR) (or Specificity [64]). We use TPR_{min} to denote the true positive rate of the minority class, and TPR_{maj} to denote the true positive rate of the majority class. It should be noted that $TPR_{min} = TNR_{maj}$.

In order to evaluate the experimental results with different methods, we also used F-measure [55] as a measurement tool for our algorithm and experiment. F-measure is defined as:

$$F - measure = (1+\beta^2)/\beta^2 \times TP/(2TP+FP+FN) \tag{6.2}$$

where β is a coefficient to adjust the value, and usually $\beta = 1$. The area under the ROC curve (AUC) which represents the expected performance as a single scalar is also used as an evaluation method in this work.

6.2 The imbalanced MI problem

Since MI learning is being employed in increasingly more practical applications as an alternative to supervised learning, the occurrences of class imbalance problems in MI learning are most often related to these applications. The problem with the application of MLO detection, as mentioned in chapter 1, is one of the related examples. In the area of MLO detection, Williams et al. [111] used innitely imbalanced logistic regression to solve the class imbalance problem. Wang et al. [108] also provided a solution to this problem, using an instance-weighted support vector machines algorithm. However, all these works are about single-instance learning. There are very few discussions related to MI learning. Thus, finding solutions for this problem is becoming an important issue in the knowledge discovery and data mining field.

For a MI dataset, it is possible that the bag number of one class is much higher than the bag number of the other classes. In this case, since the final margin of MI classification is at bag level, and the training process of MI learning always favors the bag level accuracy, the default classifier would always tend to predict "majority class". Figure 6.1 shows the class imbalance problem of MI learning where there are far more majority bags than minority bags, and the margin learned by the default classifier is "pushed" closer to the minority bags from the ideal margin. In this situation, whether a MI dataset is imbalanced or not depends on the distribution of the bags of each class. When a MI dataset is imbalanced at the bag level, it will have one of the following situations at the instance level: (1) the number of majority instances is much higher than the number of minority instances; (2) the instances are "balanced"; or (3) the number of majority instances is less than the number of minority instances.

- For the first situation

Given a bag b, its class label property $C(Y \mid b)$ can be calculated by: [116]

$$C(Y \mid b) = E_{X|b}[C(Y \mid X)] = \int_x C(Y \mid x) Pr(x \mid b) dx \tag{6.3}$$

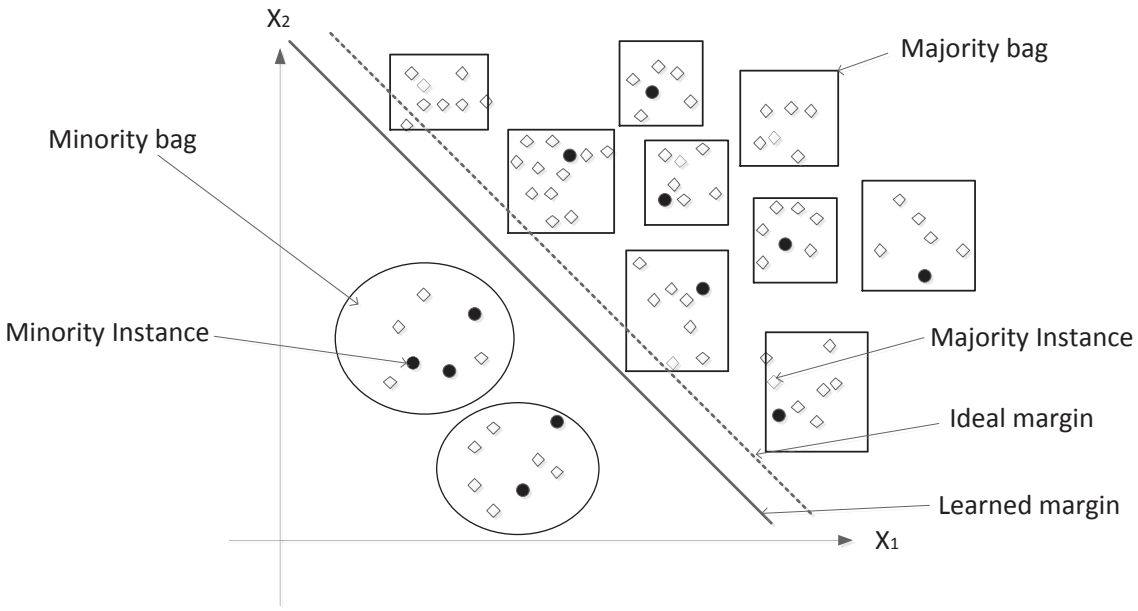

Figure 6.1: This figure illustrates the class imbalance problem of MIL.

where $x_i (i = 1, \cdots, n)$ are instances in bag b. In this situation $Pr(x_{maj} \mid b) \gg Pr(x_{min} \mid b)$, we can easily derive the following from (2):

$$C(Y \mid b) \cong \int_x C(Y \mid x_{maj}) Pr(x_{maj} \mid b) dx_{maj} \tag{6.4}$$

Here the "maj" denotes majority class and "min" denotes minority class.

- For the second and third situation

It is presented that the class imbalance problem is not only directly caused by class imbalances, but also related to the degree of overlapping among the classes [78]. Although single-instance learning is discussed in [78], we can apply similar explanations to MI learning.

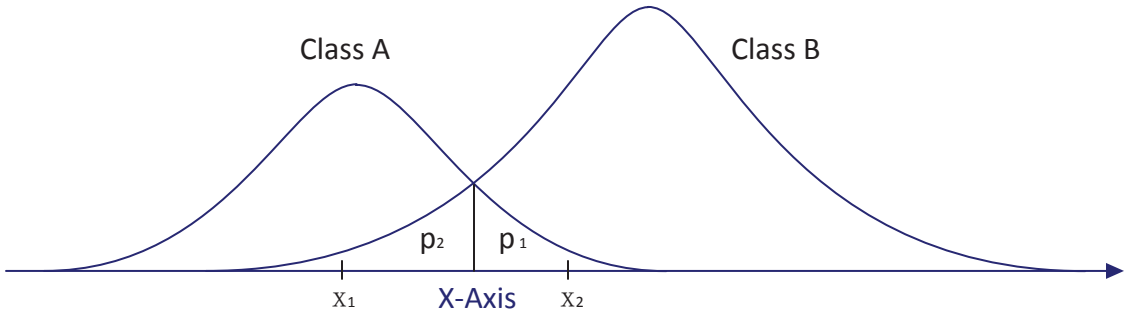

Figure 6.2: Overlap of two classes with a single feature in multi-instance dataset

Figure 6.2 illustrates a situation where a single feature x is used to classify bags as Class A or Class B. Here we assume perfect knowledge, as the probability distributions

of x conditional on class membership $f_A(x)$ and $f_B(x)$ are known. In Figure 6.2, p_1 and p_2 represents the portion of the probability density functions of Class A and Class B respectively. We can measure the overlap of the probability density functions by $p_1 + p_2$, which depends on the distance $x_2 - x_1$ separating the means of the conditional distributions relative to the standard deviations.

Suppose a proportion τ of bags in the population are from Class A. The proportion of cases misclassified with perfect knowledge will be $p_1\tau$ from Class A and $p_2(1-\tau)$ from Class B. Overall, we have $p_1\tau + p_2(1-\tau)$ proportion of bags misclassified, which depends on both the overlap and the degree of class imbalance. If the number of bags in each class is highly imbalanced, either τ or $1-\tau$ will be close to 0. For a multi-instance data set, it is common for one instance to appear in different bags. This means that the bags can be highly overlapped, even when they are from different classes. In this situation, the majority decisions will always tend to be made in favor of the bag level majority class, even if the instance level is balanced.

From the analysis, we conclude that for a MI dataset the class imbalance problem happens at the bag level. To address this problem, we must look for solutions at the bag level.

6.3 Re-sampling Techniques for Imbalanced MI Problem

For single-instance learning, SMOTE was proposed to counter the effect of having only a few instances of the minority class in a dataset [22]. Related research shows that SMOTE can create synthetic instances of the minority class, by operating in the "feature space" rather than the "data space". Generally, SMOTE has more effective performance than over-sampling with replacement [98].

Although it is based on the analysis in section 6.2, we know that the class imbalance problem for MI learning happens at the bag level. However, we are not sure if creating synthetic instances of the minority class at the instance level can improve the bag level classification. This is because creating synthetic instances in the minority bags can also increase the decision region for each bag, although this increase may not be as big as synthetic bag creation. Here we present two MI SMOTE algorithms, both at the bag level and instance level.

The original SMOTE algorithm was created for single-instance classification. To apply it to a MI dataset, we must modify the MI dataset to a single-instance dataset, and use the bag label as its instance labels. After using the SMOTE algorithm to create synthetic instances of the minority class, the new single-instance dataset must be changed back to a MI dataset. In this case, the generated synthetic instance has the same bag ID as its father instance's bag ID. As a result, the total number of minority instances increases, but the number of bags does not change. We name this MI algorithm "Instance_SMOTE".

The Instance_SMOTE algorithm creates synthetic minority instances in each bag, without creating new bags. We also introduce a Bag_SMOTE algorithm to create new synthetic minority bags with new instances.

Algorithm 6 The Bag_SMOTE algorithm

1. For each bag of minority class $X_i \in \mathbb{X}_{min}$ $i = 1, \cdots, M$, where X is the set of training bags and M is the number of bags in the minority class:
1) Create a new bag X^i_{new} with the same label as X_i, and a new bag ID; Set S_{min} as a set of all instances in $\mathbb{X}_{min}$.
2) For each feature vector (instance) x_j in X_i :
• For the continuous features, create a synthetic sample:

$$x^j_{new} = x_j + (\hat{x}_j - x_j) \times \delta \tag{6.5}$$

• For the nominal features, take a majority vote between x_j and $\hat{x}_j$ for the nominal feature value ε. In the case of a tie, choose at random. Assign ε to x^j_{new}. Here $\hat{x}_j$ is one of the K-nearest neighbors for $x_j : \hat{x}_j \in S_{min}$, and $\delta \in [0, 1]$ is a random number.
3) Insert x^j_{new} into X^i_{new}.
2. Insert X^i_{new} into $\mathbb{X}_{min}$.

Algorithm 6 shows the details of the Bag_SMOTE algorithm. The algorithm is a bag level over-sampling approach in which the minority class is over-sampled by creating "synthetic" bags, rather than by over-sampling with replacement.

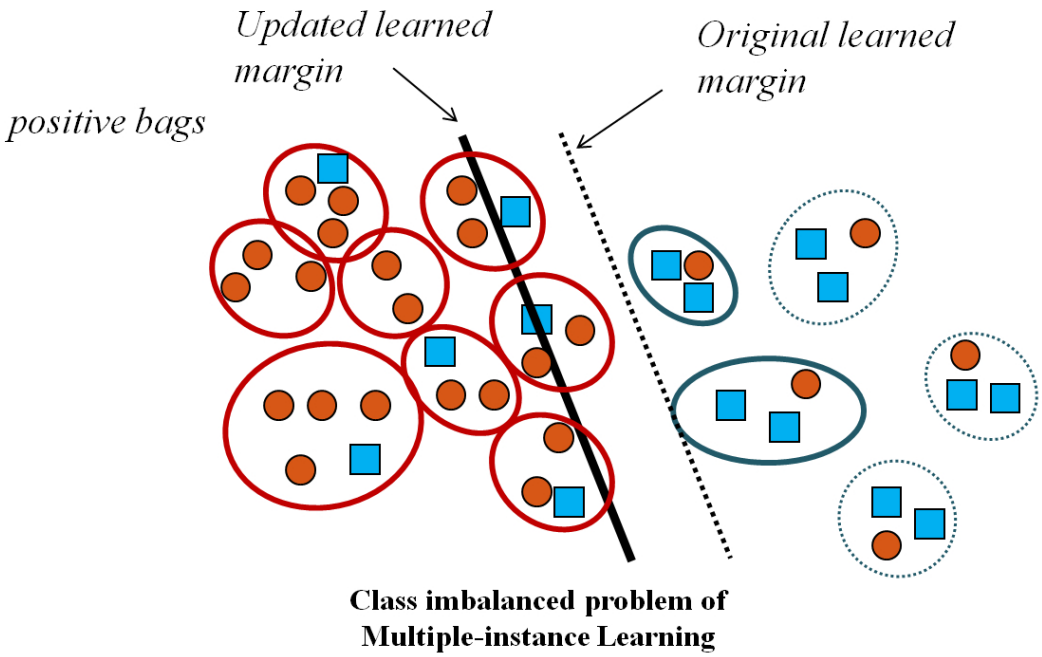

Figure 6.3: The process of generating synthetic bags on minority class

Figure 6.3 gives the process of generating synthetic bags on minority class using the

Bag_SMOTE algorithm. New bags are generated to "balance" the bag numbers of majority and minority classes.

6.4 Cost-sensitive Boosting algorithm for Imbalanced Multiple-instance Problem

Here we propose a generalized version of the Cost-sensitive Boosting-MI algorithm. Our goal is to reduce the bias inherent in the learning procedure due to class imbalance, and to increase the weights for the minority class in each round of boosting.

Take as input the training set $(x_1, y_1), \cdots, (x_m, y_m)$; $x_i \in X, y_i \in \{-1, +1\}$, where x_i is an n-tuple of attribute values belonging to a certain domain or instance space X, and y_i is a label in a label set Y. The key process of the AdaBoost.M1 method [48] is to iteratively update the distribution function over the training data. This means that for every iteration $t = 1, \cdots, T$, where T is a given number of the total number of iterations, the distribution function D_t is updated sequentially, and used to train a new hypothesis:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{6.6}$$

where $\alpha_t = 0.5 \ln((1 - \varepsilon_t)/\varepsilon_t$ is the weight updating parameter, $h_t(x_i)$ is the prediction output of hypothesis h_t on the instance x_i, ε_t is the error of hypothesis h_t over the training data, and Z_t is a normalization factor.

Schapire and Singer [86] used a generalized version of Adaboost, as shown in [86], where the training error of the final classifier is bounded as:

$$\frac{1}{m} \left|\{i : H(x_i) \neq y_i\}\right| \leq \prod_t Z_t \tag{6.7}$$

where

$$\begin{aligned} Z_t &= \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \\ &\leq \sum_i D_t(i) \left(\frac{1+y_i h_t(x_i)}{2} e^{-\alpha} + \frac{1-y_i h_t(x_i)}{2} e^{\alpha}\right) \end{aligned} \tag{6.8}$$

Minimizing Z_t on each round, α_t is induced as:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{\sum_{i, y_i = h_t(x_i)} D_t(i)}{\sum_{i, y_i \neq h_t(x_i)} D_t(i)} \right) \tag{6.9}$$

The weighting strategy of AdaBoost identifies samples by their classification outputs, as correctly classified or misclassified. However, it handles samples of different classes equally. The weights of misclassified samples from different classes are increased by an

identical ratio, while the weights of correctly classified samples from different classes are decreased by an identical ratio.

Similar to the methods for managing the single-instance class imbalance problem in [40], [42], [93] and [96], the learning objective when dealing with the multi-instance class imbalance problem is to improve the identification performance on the minority class. Our strategy is to apply cost-minimizing techniques to the combination schemes of ensemble methods. This objective expects that the weighting strategy of a boosting algorithm will preserve a significant weighted sample size of the minority class. A preferred boosting strategy can distinguish different types of samples, and boost more weights on the samples associated with higher identification importance.

To indicate the different identification importance among bags, each bag is associated with a cost item; the higher the value, the higher the importance of correctly identifying the sample. For an imbalanced multi-instance dataset, there are many more bags with the majority class label than with the minority class label. Using the same learning framework as AdaBoost, the cost items can be fed into the weight update formula of AdaBoost to bias the weighting strategy. Algorithm 7 shows the proposed algorithms.

Algorithm 7 Cost-sensitive boosting algorithm for MI learning

Given: A multi-instance training dataset with a set of bags $X_i, i = 1, \cdots, N$, where each bag can consist of an arbitrary number of instances and a given label:$X_i = \{x_i^1, x_i^2, \cdots, x_i^{n_i}; Y_i\}$, $i = 1, \cdots, N, Y_i \in \{-1, +1\}$, and each instance $x_i^{n_i}$ is an M-tuple of attribute values belonging to a certain domain or instance space $\mathbb{R}$.
Initialize $D_1(i) = 1/m$.
For $t = 1, \cdots, T$ and the constraint condition η is satisfied
1. Train a weak learner using distribution D_t.
2. Get a weak hypothesis $h_t : X \longrightarrow \mathbb{R}$.
3. Choose $\alpha_t \in R$.
4. Update:

$$D_{t+1}(i) = \frac{D_t(i) K_t(X_i, Y_i)}{Z_t} \tag{6.10}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).
Output the final hypothesis:

$$H(X) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(X)\right) \tag{6.11}$$

For the original Adaboost, $K_t(X_i, Y_i)$ in (6.10) is set as $\exp(-\alpha_t Y_i h_t(X_i))$. Our proposed algorithms introduced cost items into the weight update formula of AdaBoost, either inside the exponent, outside the exponent, or both. Each modification can be a new boosting algorithm, denoted as $Ab(p, q)$. The modifications of $K_t(X_i, Y_i)$ are then given by:

$$K_t(X_i, Y_i) = C_i^p \exp(-C_i^q \alpha_t Y_i h_t(X_i)) \tag{6.12}$$

Now we induce the weight update parameter α_t and constraint condition η in Algo-

rithm 7 for $Ab(p,q)$. From (6.12) we get:

$$D_{t+1} = \frac{C_i^p D_t(i) \exp\left(-C_i^q \alpha_t Y_i h_t(X_i)\right)}{Z_t} = \frac{C_i^{pt} \exp\left(-C_i^q Y_i f(X_i)\right)}{m \prod_t Z_t} \tag{6.13}$$

where

$$f(X) = \sum_t \alpha_t h_t(X) \tag{6.14}$$

and

$$Z_t = \sum_i C_i^p D_t\left(i\right) \exp\ \left(-C_i^q \alpha_t Y_i h_t\left(X_i\right)\right) \tag{6.15}$$

The overall training error is bounded as:

$$\begin{aligned} \frac{1}{m} \left|\{i : H(X_i) \neq Y_i\}\right| &\leq \frac{1}{m} \sum_i C_i^p \exp\left(-C_i^q Y_i f(X_i)\right) \\ &= \prod_t Z_t \sum_i \left(\frac{C_i^p}{C_i^{pt}}\right) D_{t+1}\left(i\right) \end{aligned} \tag{6.16}$$

According to [86], for weak hypotheses $C_i^2 \alpha_t Y_i h_t(X_i) \in [-1, +1]$ with range [-1,+1], α can be obtained by approximating Z as follows:

$$\begin{aligned} Z_t = &\sum_i C_i^p D_t(i) \exp\left(-C_i^q \alpha_t Y_i h_t(X_i)\right) \leq \\ &\sum_i C_i^p D_t(i) \left(\frac{1+C_i^q Y_i h_t(X_i)}{2} e^{-\alpha_t} + \frac{1-C_i^q Y_i h_t(X_i)}{2} e^{\alpha_t}\right) \end{aligned} \tag{6.17}$$

Let

$$G(\alpha_t) = \sum_i C_i^p D_t(i) \left(\frac{1+C_i^q Y_i h_t(X_i)}{2} e^{-\alpha_t} + \frac{1-C_i^q Y_i h_t(X_i)}{2} e^{\alpha_t}\right) \tag{6.18}$$

Our purpose is for α_t to minimize $G(\alpha_t)$, so we can obtain:

$$G'\left(\alpha_t\right) = \frac{dG}{d\alpha_t} = 0. \tag{6.19}$$

Next, we can analytically obtain α_t from (6.19), giving:

$$\alpha_t = \frac{1}{2} \times \ln\left(\frac{\sum_i C_i^p D_t(i) + \sum_{i, Y_i = h_t(X_i)} C_i^{p+q} D_t(i) - \sum_{i, Y_i \neq h_t(X_i)} C_i^{p+q} D_t(i)}{\sum_i C_i^p D_t(i) - \sum_{i, Y_i = h_t(X_i)} C_i^{p+q} D_t(i) + \sum_{i, Y_i \neq h_t(X_i)} C_i^{p+q} D_t(i)}\right) \tag{6.20}$$

The sample weight updating goal of AdaBoost is to decrease the weight of the training samples that are correctly classified, and increase the weight of the opposite samples [48] [86]. Therefore, α_t should be a positive value, and the training error should be less than random guessing, based on the current data distribution. To ensure that α_t is positive, we get

$$\sum_{i, Y_i = h_t(X_i)} C_i^{p+q} D_t(i) \ > \sum_{i, Y_i \neq h_t(X_i)} C_i^{p+q} D_t(i) \tag{6.21}$$

This is the constraint condition η in Algorithm 7.

6.5 Experimental Results

6.5.1 Experimental results on benchmark datasets

Experimental setup

The techniques presented in this chapter were implemented in the Java in the framework of the WEKA machine learning tool [112]. The 9 datasets we used in our empirical study listed in Table 6.2 are from popular benchmark datasets employed in [14] and [45]. The percentage of minority bags varies from 8.27% to 25.15%, while the percentage of minority instances (instances in minority bags) varies from 8.34% to 25.38%. The datasets come from a wide range of application domains.

Table 6.2: Details of datasets

Datasets	# bags	# attribute	# min bags	% min bags	# min instances	% min instances
Elephant	125	230	25	20.00	150	16.45
Fox	121	230	21	17.36	134	17.16
Tiger	126	230	26	20.63	164	23.16
M_atoms	167	10	42	25.15	365	25.38
M_bonds	160	16	35	21.88	603	16.95
M_chains	152	24	27	17.76	514	11.10
Component	281	200	48	17.08	186	17.05
Function	233	200	41	17.60	254	12.99
Process	220	200	31	14.09	179	14.72

All the datasets used in this study have a binary class. In this section, we explain the experiments we conducted to investigate and compare the proposed algorithms. The Tree method was chosen as the weak learner/base learner because it is, 1) stable in MI learning [13], [14], and 2) suitable to be the weak learner in many related works [93], [97]. For sampling methods, the degree of minority over-sampling is determined by the imbalance ratio, which means that in all datasets new minority class bags were created or replicated until the number of minority class bags was equal to the number of majority class bags.

For the cost-sensitive methods, the original costs were usually chosen according to the number of bags of each class (higher weights on minority classes and lower weights on majority classes), except for the experiments which investigated the effects of cost ratio setups. The iteration rounds of boosting could be terminated by, a) the prefixed number T, or b) the constraint condition η in algorithm 7. The parameter T, which governs the number of classifiers generated, was set to ten in each boosting algorithm. The Ten-fold Cross-Validation method was used in all experiments.

Experimental results of re-sampling methods

- **Degree of minority over-sampling**

Figure 6.4 compares the minority bag oversampling with replacement (Bag_over-sampling), Instance_SMOTE and Bag_SMOTE on different degrees of minority over-sampling. The experiments were conducted on the Mutagenesis3_atoms dataset, and MITI [12] was chosen as the classifier. The minority class was over-sampled from 100% to 800% of its original size. As a result, the minority bag number has increased from 100% to 800% for Bag_over-sampling and Bag_SMOTE; the bag number does not increase for Instance_SMOTE, but the number of minority instances increases from 100% to 800%.

Figure 6.4 shows that with an increase of the minority over-sampling degree, the TPR_min of Bag_over-sampling and Bag_SMOTE increases, while that of Instance_SMOTE maintains almost the same value. The Gmean of these three method follows the trend of TPR_min, since the change of TNR_min is not as great as the change of TPR_min. After 200% over-sampling, the Gmean of Bag_SMOTE is always bigger than the Gmean of the other two methods, since Bag_SMOTE gets more preference on TPR_min, but loses less on TNR_min.

- **Comparison of different re-sampling technologies**

Table A.1 presents the experimental results of the base learners (MITI [14]) and all the sampling methods, using the base learners on all datasets. This comparison shows that compared with the base learner, almost all of these re-sampling methods have increases of TPR_min and decreases of TNR_min. Since the increase of TPR_min is always greater than the decrease of TNR_min, the Gmean of all re-sampling methods is improved, with Bag_SMOTE being the best.

Experimental results of Cost-sensitive boosting methods

- **Cost ratio setups**

For cost-sensitive techniques for single-instance class imbalanced learning, the cost was experimentally chosen according to the instance number in each class [40][42][59][96]. In this experiment, we investigated the effects of cost ratio setups. The misclassification cost C_M denotes the misclassification cost of the majority class, and C_m denotes that of the minority class. The ratio between C_M and C_m shows the deviation of the learning importance between the two classes. Here, we fix the cost item of the minority class to 1, while changing the cost item of the majority class in the range of 0.1 to 1.0. When the cost ratio is 1.0, the proposed algorithm reduces to the MI AdaBoost algorithm.

Figure 6.5 shows the experimental result of this classification when using Ab (1,1) with the base learner MITI [14] on Mutagenesis3_atoms dataset. Using a Ten-fold Cross-Validation method, the real imbalance ratio between the number of minority and majority

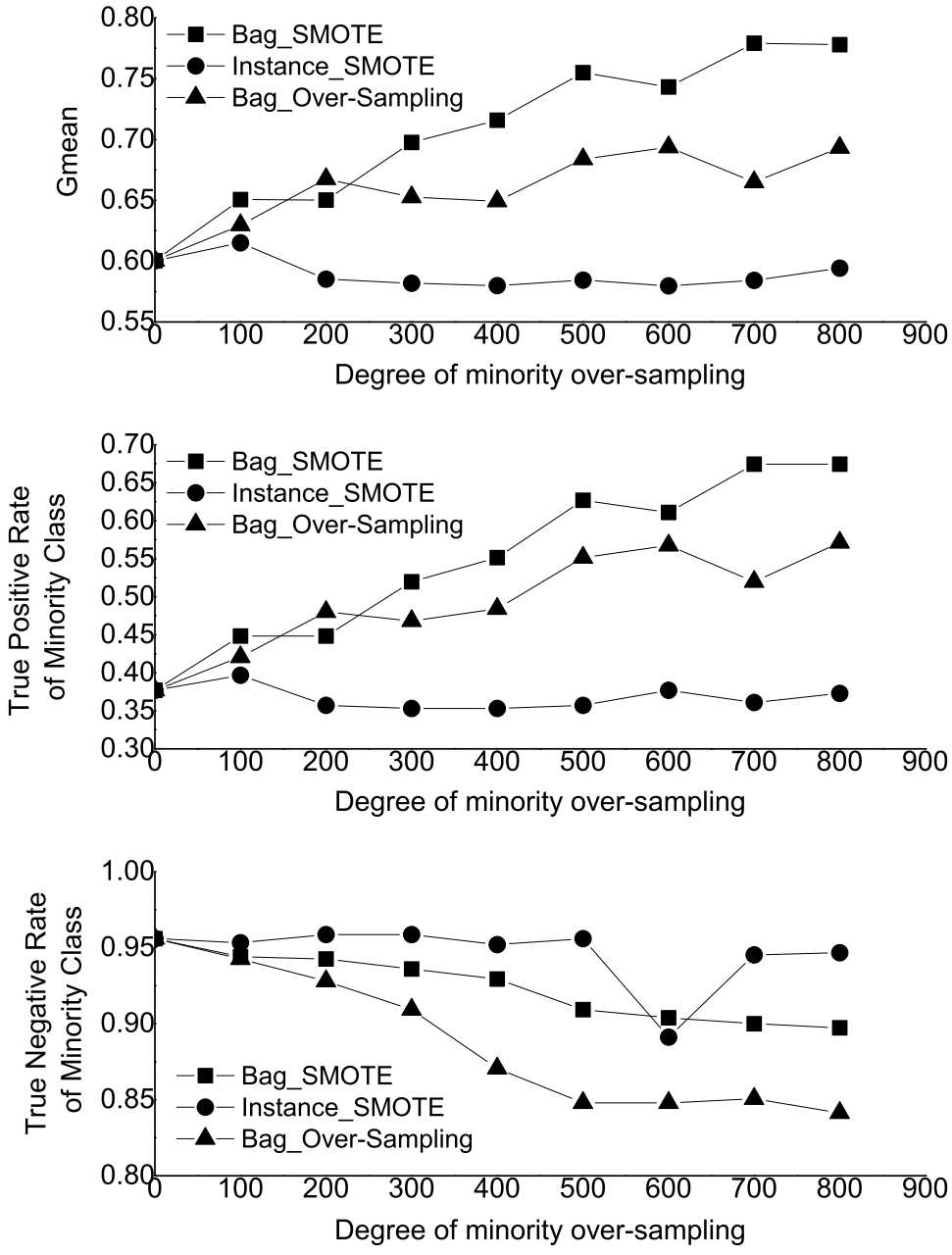

Figure 6.4: Comparison of % Minority correct for replicated Bag_over-sampling, Instance_SMOTE and Bag_SMOTE for the M_atoms dataset

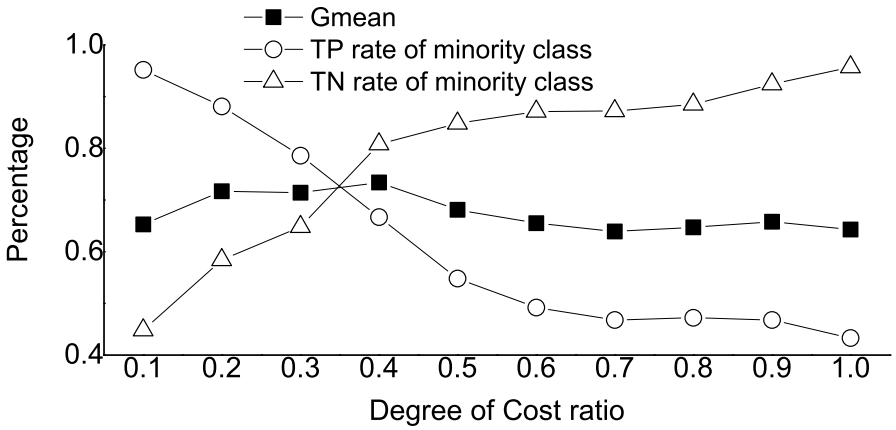

Figure 6.5: Gmean, TP rate and TN rate with Cost ratio setups for $Ab(1,1)$ on M_atoms dataset using MITI[14]

bags for each experiment is about 0.33. From Figure 6.5, we find that when the cost ratio is set from 0.1 to 1.0, the true negative rate increases and the true positive rate decreases. The best Gmean is obtained when the cost ratio is 0.4, which is close to the bags imbalance ratio.

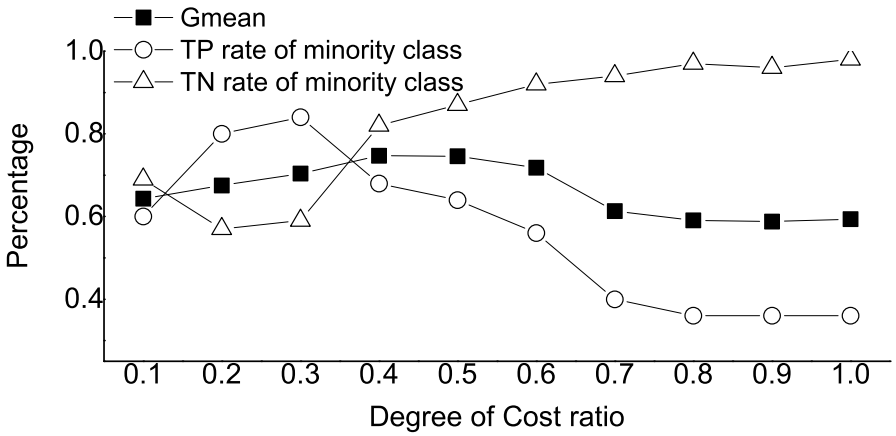

Figure 6.6: Gmean, TP rate and TN rate with Cost ratio setups for $Ab(2,1)$ on Elephant dataset using MIRI [13]

We then we chose the same strategy with a different classifier on a different dataset. MIRI was chosen as the base learner and Ab (2,1) as the boosting algorithm, and we repeated the experiment on the elephant dataset. Figure 6.6 shows the experimental result. The real imbalance ratio is about 0.25. We find that after the point where cost ratio is 0.3, TPR_min decreases and TNR_min increases with an increase in the cost ratio degree. But before cost ratio 0.3 the graph does not follow this rule, and the best Gmean is when the cost ratio equal 0.4.

- **Parameters P and Q setups**

In (6.13), the parameters P and Q are the coefficient parameters of the cost items inside and outside the exponent respectively. In this experiment we investigated the impact of P and Q setups. Here we set P and Q in a range from 0 to 2.5, in steps of 0.5. When P=Q=0, the proposed algorithm reduces to the MI AdaBoost algorithm.

Table 6.3: Comparison of coefficient parameters P and Q setups using Gmean as the evaluation on M_atoms dataset

	Q=0	0.5	1.0	1.5	2.0	2.5
P=0	0.643	0.632	0.668	0.695	0.715	0.727
0.5	0.648	0.677	0.694	0.750	0.719	0.716
1.0	0.666	0.694	0.737	0.720	0.689	0.666
1.5	0.647	0.731	0.702	0.705	0.691	0.666
2.0	0.712	0.775	0.714	0.680	0.609	0.648
2.5	0.702	0.720	0.642	0.648	0.632	0.636

Table 6.3 shows the experimental result of this classification on dataset Mutagenesis3_atoms. Here we still used MITI [14] as the base learner, and chose Gmean as the evaluation measurement.

From Table 6.3, we can see that increasing P and Q independently consistently increases the Gmean value of the classification. However, changing these two cost items concurrently gives the classification a chance for better performance. In this experiment, the best performance is at (P=2.0, Q=0.5). When the sum of P and Q is over 2.5, the performance of the classifier begins to decrease.

- **Comparison of different cost-sensitive boosting technologies**

Table A.1 also shows the experimental results of the base learner (MITI [14]) and the base learner with the presented cost-sensitive boosting algorithms using different coefficient parameters P and Q on all datasets. A set of (P,Q) settings of [(0,0); (0,1); (1,0); (1,1); (1,2); (2,1); (2,2)] are tested. Here Ab (0,0) is actually the original Adaboost algorithm, as mentioned previously.

Friedman's test [60] was performed on the multiplication results of Gmean and F-measure, and the result is shown in Table 6.4.

Table 6.4: Friedman's test result

Friedman X^2	df	p-value	Critical X^2
28.028	12	0.00548	18.31
28.028 > 18.31, hypothesis rejected			

Since Friedman's test shows that the classifiers performed differently, we applied Nemenyi's post-hoc test [60] to determine which pairs of classifiers had different performance. The scores of all the classifiers are shown in Table 6.5. For example, the result of 10-0-0 for Ab (1,1) means that, compared to the other 10 classifiers, this classifier wins 10 times, is

equal zero times, and loses zero times. If we set the scores as win=1, equal=0 and lose=-1, the total score of each classifier can be calculated. This result is also shown in Table 6.5.

Table 6.5: Experiment result using the statistical test method

	Ab(1,1)	Ab(1,2)	Ab(2,1)	Ab(0,1)	Ab(2,2)	Bag_ SMOTE	Ab(1,0)	Bag_Over-Sampling	Instance_ SMOTE	Adaboost	Base_ Learner
Gmean×(F-measure)	10-0-0	9-0-0	8-0-2	6-1-3	6-1-3	5-0-5	4-0-6	3-0-7	1-1-8	1-1-8	0-0-10
Score	10	9	6	3	3	0	-2	-4	-7	-7	-10

6.5.2 Analysis and discussion

Based on the experiments conducted in this work, a number of conclusions can be drawn.

The experimental results show that the three proposed solutions (i.e. the classifier independent method re-sampling technique, the classifier dependent method cost-sensitive boosting technique, and the Adaweight_SVM algorithm) can improve the performance of the base learner for class imbalanced multi-instance learning.

For the re-sampling technique, experimental results show that the classification performance using Instance_SMOTE is not as high as when using Bag_over-sampling and Bag_SMOTE. This proves our expectation that the class imbalance problem occurs at bag-level, so the solution should also be found at the bag-level. The fact that Bag_SMOTE is better than the bag replication method shows that with bag replication, the decision region that results in a classification decision for the minority class can be smaller and more specific. Bag_SMOTE can cause the classifier to create larger decision regions that contain nearby minority class bags.

The cost-sensitive boosting technique assumes that a cost matrix is known for different types of errors or samples. However, for a real world dataset the cost matrix is often unavailable. Experimenting with cost ratio setups for cost-sensitive learning shows that the best cost ratio setup is dependent on the dataset and the chosen classifier. When the cost matrix is unavailable, choosing the natural bags imbalance ratio as the original cost ratio for cost-sensitive learning proved to be a practical solution for the algorithms presented in our research. Moreover, the coefficient parameters of the cost items were found to be related to the classification performance of class imbalanced MI learning. The combination of these two coefficient parameters gives us opportunities to achieve better classification performance.

In our experiment, the performance of Ab (1,1) is the best overall (Table A.1), followed by Ab (1,2), Ab (2,1), Ab (0,1) and Ab (2,2). These experimental results show that the weighted updating strategy of cost-sensitive boosting algorithms increases the weights on the misclassified bags from the minority class more than it does on those from the majority class. Similarly, it decreases the weights on correctly classified bags from the minority class less than on those from the majority class.

In both Table A.1 and Table 6.5, we also compared the re-sampling framework and the Cost-sensitive boosting framework on all datasets. Most Cost-sensitive boosting algorithms show more competitive performance than the Bag_SMOTE technique. A possible reason for this is that when Bag_SMOTE creates new synthetic bags, majority instances could also

be included in the original minority bags, which would increase the chance of introducing noise for created bags.

6.6 Conclusion and Future work

In this chapter, we have presented a comprehensive and systematic analysis of a challenging and critical issue in the knowledge discovery and data mining field: the multi-instance class imbalance problem. We have also provided two novel solution frameworks for this problem: re-sampling technology and cost-sensitive boosting technology. Based on these methods, we conducted experimental analysis using different learning algorithms with nine real-world benchmark datasets from a variety of application domains, and four datasets from a real-world project. Experimental evidence derived from standard datasets was presented to support the improved performance of the proposed algorithms. We found that the presented algorithms that included both data re-sampling methods and cost-sensitive boosting methods deal with the class imbalance problem in MI learning effectively.

The objective of this research is to provide practical guidance to machine learning practitioners when they build classifiers from multi-instance class imbalanced datasets, and to provide researchers with possible directions for future study. It would be worthwhile to investigate if the methods presented here are suitable for datasets with extremely high imbalance ratios. For re-sampling algorithms and cost-sensitive boosting algorithms, Tree methods are also used as base learners for techniques presented in this chapter. Although the presented re-sampling algorithms and cost-sensitive boosting algorithms are learner independent, the setup of the parameters could be influenced by choosing base learners with different technologies. Future work will include investigating the relationship between these two presented techniques and different base learners. We also intend to research other applications of the presented algorithms for multi-instance classification, and investigate other related algorithms.

Chapter 7

Applying Instance-weighted Boosting SVM to Imbalanced MI Learning

7.1 Introduction

Support vector machines, a classification method based on the principle of Structural Risk Minimization from statistical learning theory, is widely used in the machine learning area. For MI learning, Andrews et al. [4] presented an approach to multiple-instance learning based on two alternative generalizations of the maximum margin idea used in SVM classification. Cheung & Kwok [29] presented a support vector regression approach for MI regression under the standard MI regression assumption. To make computation feasible, their method relies on the simplifying assumption that the primary instance is the one with the highest output value according to the SVM.

In this chapter we apply an instance-weighted variant of SVM combining with MI_fusion which is presented in chapter 4 to deal with the MI learning problem. We develop an asymmetric boosting method on the weights of the trade-off parameters to optimize the instance-weighted SVM with both 1-norm and 2-norm format. The rest of this chapter is organized as follows: Section 7.2 reviews the related algorithms. In section 7.3, we introduce and discuss our new algorithm in detail. After these developments, we also present an experimental part (subsection 7.3.4) that illustrates the efficiency of our algorithm and we make some observations about its performance in this section. Section 7.4 is the conclusion and future work.

7.2 Background and Related Work

SVM [16][33] is a learning algorithm seeking a linear decision boundary

$$f(x) = \omega^T \Phi(x) + b \tag{7.1}$$

in a feature space $\mathcal{F}$, where $\Phi : \chi \to \mathcal{F}$ is a map from the input space χ to the feature space $\mathcal{F}$, $\omega \in \mathcal{F}$ is a coefficient vector, $b \in \mathbb{R}$ is a bias term.

The parameters ω and b are learned as

$$\min_{\xi,\omega,b} \|\omega\|^2 + C \sum_{i=1}^{l} \xi_i^n \tag{7.2}$$

Subject to:

$$y_i(\omega^T \Phi(x_i) + b) \geq 1 - \xi_i, i = 1 \dots l,$$
$$\xi_i \geq 0, i = 1 \dots l,$$

where l is the number of training instance, C is the trade-off parameter, ξ is the so-called slack variable.

The corresponding Lagrangian for the 1-norm (n=1) soft margin optimization problem is:

$$L = \frac{1}{2} \min_{\xi,\omega,b} \|\omega\|^2 + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i [y_i(\omega^T \Phi(x_i) + b) - 1 + \xi_i] - \sum_{i:y_i=1}^{l} \mu_i \xi_i \tag{7.3}$$

Where $\alpha_i \geq 0$ and $\mu_i \geq 0$.

Although SVM has been very successful in many application areas, when faced with imbalanced datasets, the performance of SVM drops significantly [75][114]. The causes for this performance loss are: (a) Minority points lie further from the ideal boundary and the imbalanced support vector ratio. Wu & Chang [114] explain this phenomenon as one source of boundary skew. They point out that the imbalance in the training data ratio means that the minority instances may lie further away from the "ideal" boundary than the majority instances. Moreover, they mention that the ratio between the majority and the minority support vectors also becomes imbalanced when the data gets imbalanced. As a result, the neighborhood of a test instance close to the boundary is more likely to be dominated by majority support vectors and the decision function is more likely to classify a boundary point into majority class. (b) Akbani et al. [2] pointed that the constant C specifies the tradeoff between maximizing the margin and minimizing the error. If C is not very large, SVM simply learns to classify everything as majority class because that makes the "margin" the largest with zero cumulative error on the minority examples, which does not count for much. This explains why SVM fails completely in situations with a high degree of imbalance.

To deal with the class imbalanced problem on single-instance data using SVM, many works have been recently done. Shawe-Taylor et al. [89] showed that the distance of a test point from the boundary is related to its probability of misclassification. This observation resulted in the generation of a related technique which was used in their paper. The technique provides a more severe penalty if an error is made on a minority example than if it is made on a majority example.

Morik et al. [75] proposed an algorithm that uses the 1-norm (n=1). Two cost-factors are chosen so that the potential total cost of the false positives equals the potential total cost of the false negatives. This means that the parameters of the SVM are selected so that they obey the ratio: $C^+/C^- = N^-/N^+$ where C^+ and C^- are cost factors that adjust the cost of false positive vs. false negatives and N^+ and N^- are the numbers of positive instances and negative instances respectively. By increasing the margin on the side of the minority class, this method provides a way to induce a decision boundary which is much more distant from the "critical" class than it is from the other. Finding this hyper-plane can be translated into the following optimization problem:

$$\min_{\xi,\omega,b} \|\omega\|^2 + C^+ \sum_{i:y_i=1}^{l} \xi_i + C^- \sum_{i:y_i=-1}^{l} \xi_i \tag{7.4}$$

Subject to:

$$y_i(\omega^T \Phi(x_i) + b) \geq 1 - \xi_i, i = 1 \dots l,$$
$$\xi_i \geq 0, i = 1 \dots l,$$

The corresponding Lagrangian for the 1-norm soft margin optimization problem is:

$$L = \frac{1}{2} \min_{\xi,\omega,b} \|\omega\|^2 + C^+ \sum_{i:y_i=1}^{l} \xi_i + C^- \sum_{i:y_i=-1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i [y_i(\omega^T \Phi(x_i) + b) - 1 + \xi_i] - \sum_{i:y_i=1}^{l} \mu_i \xi_i \tag{7.5}$$

Where $\alpha_i \geq 0$ and $\mu_i \geq 0$.

The dual problem of problem (7.4) is

$$\max_{\alpha} L = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j k(x^i, x^j) \tag{7.6}$$

Subject to

$$0 \leq \alpha_i \leq C^+$$
$$0 \leq \alpha_i \leq C^-$$
$$\sum_{i=1}^{l} y_i \alpha_i = 0$$

Instead of using the 1-norm for the loss measure, Veropoulos et al. [99] use the 2-norm (n=2 in equation (7.2)). This method enables the algorithm to control the balance between sensitivity and specificity without adding any additional information. Experimental results [99] show that this method is also an effective method to deal with the class imbalanced problem.

There have been many works in the community that apply general sampling and ensemble techniques to the SVM framework. For instance, Akbani et al. [2] presented a so-called SDC algorithm which combined the SMOTE [22] technique with the weighted_SVM

methods [75]. This algorithm makes the minority instances more densely distributed in an attempt to guarantee a more well-defined boundary. In Akbani et al.'s work [2], they compared five different techniques: SVM, SVM+under_ sampling, SMOTE+SVM, weighted_SVM and their presented algorithm: SMOTE+weighted_SVM. From their experimental results, we find that the weighted_SVM performs the best comparing with the first three techniques which use the statistical testing method. Comparing the weighted_SVM with the SMOTE+weighted_SVM, the weighted_SVM is still competitive (wins 3 times, ties 2 times and loses 5 times). In their experiments all SVMs are 1_norm.

Meanwhile, the methods proposed in [63][72] develop ensemble systems by modifying the data distributions. Moreover, Wang & Japkowicz [100] proposed to modify the Adaboost method by combining it with the weighted_SVM. With this method, a modified version of the training data based on a sequential learning procedure can be built to improve classification performance. Wu & Chang [114] proposed a class boundary-alignment algorithm, which modified the kernel matrix K based on the distribution of the training data. Instead of using an input space, they conducted the kernel transformation based on the spatial distribution of the support vectors in feature space.

Although there are so many works related with SVM on class imbalanced problem, as we mentioned in chapter 6, all of them are about single-instance learning. It is very necessary for us to research on SVM about imbalanced MI problem.

7.3 Proposed Approach

The motivation of our approach comes from Morik et al. [75] and Veropoulos et al.[99]'s algorithms. Unlike the penalty term in Morik et al. [75] and Veropoulos et al.[99]'s algorithm, where the values of C^+ and C^- are fixed and all training data points with the same label are equally treated during the training process, the penalty term in our algorithm is weighted in order to reduce the effect of less important data points.

To adjust the cost of false training data points, we choose a more general setting which assigns each instance a regularization parameter W_i [70][119]. This parameter vector gives us more flexibility for adjusting the cost of false positives vs. false negatives. Although we still call this algorithm "instance-weighted", here the "weight" is a regularization parameter on slack variables in the optimization function of SVM, not the weight of the instance used in Adaboost [86].

The function of n-Norm Soft Margin instance-weighted SVM can be formulated as the following optimization problem: The parameters ω and b are learned as

$$\min_{\xi,\omega,b} \|\omega\|^2 + C\sum_{i=1}^{l} W_i \xi_i^n \tag{7.7}$$

Subject to:

$$y_i(\omega^T \Phi(x_i) + b) \geq 1 - \xi_i, i = 1 \ldots l,$$
$$\xi_i \geq 0, i = 1 \ldots l,$$

7.3.1 1-Norm Soft Margin instance-weighted SVM

The corresponding Lagrangian for the 1-norm soft margin optimization problem is:

$$L = \frac{1}{2}\min_{\xi,\omega,b} \|\omega\|^2 + C\sum_{i:y_i=1}^{l} W_i\xi_i - \sum_{i=1}^{l}\alpha_i[y_i(\omega^T\Phi(x_i)+b) - 1 + \xi_i] - \sum_{i:y_i=1}^{l}\mu_i\xi_i \tag{7.8}$$

Where $\alpha_i \geq 0$ and $\mu_i \geq 0$

Setting the derivatives of the above Lagrangian w.r.t the primal variables ω, b and ξ_i to zero, we obtain

$$\frac{\partial L}{\partial \omega} = 0 \Leftrightarrow \omega = \sum_{i=1}^{l}\alpha_i y_i \Phi(x_i) \tag{7.9}$$

$$\frac{\partial L}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^{l}\alpha_i y_i = 0 \tag{7.10}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Leftrightarrow \alpha_i = CW_i - \mu_i \tag{7.11}$$

where 0 denotes the vector with all zeros. $i = 1\ldots l$,. Since $\mu_i \geq 0$ we can get $0 \leq \alpha_i \leq 1, i = 1\ldots l$.

Substituting these equations into (7.8), we arrive at the following dual problem :

$$\max_{\alpha} L = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{7.12}$$

Subject to

$$0 \leq \alpha_i \leq CW_i$$
$$\sum_{i=1}^{l}\alpha_i y_i = 0$$

and the KKT [17][33] conditions of this formulation become

$$\begin{aligned} \alpha_i[y_i(\omega^T\Phi(x_i)+b) - 1 + \xi_i] &= 0 \\ (CW_i - \alpha_i)\xi_i = 0, i &= 1\ldots l. \end{aligned} \tag{7.13}$$

Figure 7.1 shows the separating hyper plane and the margin of this classification problem where each instance is assigned a regularization parameter W_i. The point labeled ξ_i are said to be on the wrong side of their margin with the cost factor CW_i, and each parameter of the point labeled ξ_i can affect the hyper plane and margin.

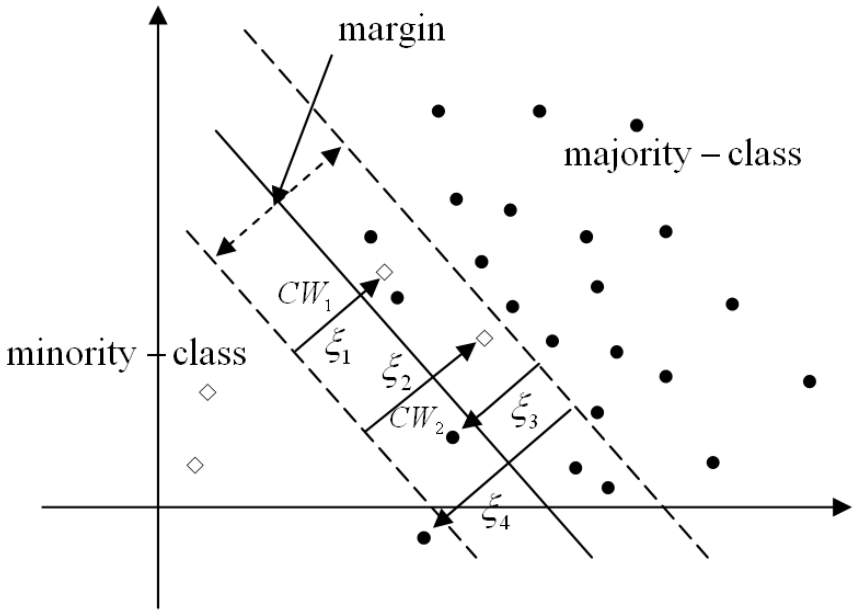

Figure 7.1: The non-separable classification problem together with the separating hyper plane and the margin.

7.3.2 2-Norm Soft Margin instance-weighted SVM

The corresponding Lagrangian for the 2-norm soft margin optimization problem is:

$$L = \frac{1}{2}\min_{\xi,\omega,b} \|\omega\|^2 + C\sum_{i:y_i=1}^{l} W_i\xi_i^2 - \sum_{i=1}^{l}\alpha_i[y_i(\omega^T\Phi(x_i)+b) - 1 + \xi_i] \tag{7.14}$$

Where $\alpha_i \geq 0$ and $\mu_i \geq 0$

Setting the derivatives of the above Lagrangian w.r.t the primal variables ω, b and ξ_i to zero, we obtain

$$\frac{\partial L}{\partial \omega} = 0 \Leftrightarrow \omega = \sum_{i=1}^{l}\alpha_i y_i \Phi(x_i) \tag{7.15}$$

$$\frac{\partial L}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^{l}\alpha_i y_i = 0 \tag{7.16}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Leftrightarrow \alpha_i = CW_i\xi_i \tag{7.17}$$

where 0 denotes the vector with all zeros and $i = 1\ldots l$,. Since $\mu_i \geq 0$ we can get that $0 \leq \alpha_i \leq, i = 1\ldots l$.
Substituting these equations into 7.14, we arrive at the following dual problem :

$$\max_{\alpha} L = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \frac{1}{2C}\|\alpha\|^2 \tag{7.18}$$

Subject to

$$\alpha_i \geq 0$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

and the KKT [17][33] conditions of this formulation become

$$\alpha_i[y_i(\omega^T \Phi(x_i) + b) - 1 + \xi_i] = 0 \tag{7.19}$$

7.3.3 Instance-weighted boosting SVM algorithm

In equation 7.7, each instance is assigned a regularization parameter W_i. Our target is to choose the best set of parameters W_i so that the classifier can accurately predict unknown data (i.e. testing data). It is not known beforehand the optimal value of W_i for a given problem.

In [57], Hsu et al. provided grid search through a subset of the hyper-parameter space of a learning algorithm to solve the problem of hyper-parameter optimization. In [12], Bergstra et al. compared different strategies and proved that random search is generally better than grid search for hyper-parameter optimization. However, when faced with too many parameters, neither grid search nor random search is suitable for the problem of hyper-parameter optimization. Our idea is to update the hyper-parameters W_i iteratively over the training examples by calling the instance-weighted SVM with the initial hyper parameters as the base learner. The process is similar to the Adaboost [86] algorithm.

Here we present an instance-weighted boosting SVM (IB_SVM) algorithm for the hyper-parameter optimization of W_i.

Different from the traditional Adaboost [86] algorithm and its variants [42][93], IB_SVM does not optimize the actual weights of instances. Instead, it tries to optimize the regularization parameter on the cost of slack variables in function (7.7). Algorithm 8 gives the details for this algorithm.

At each iteration t, IB_SVM makes a call to the base learner with input set and hyper-parameter distribution W_t and returns a hypothesis h_t. The distribution $W_t(i)$ is updated so that the values of $W_t(i)$ that h_t misclassifies are reduced and the values of $W_t(i)$ that h_t classifies correctly are increased. This way, in the next iteration the base learner is forced to focus its attention on the examples that have been misclassified more often in previous rounds. However, IB_SVM focuses more attention on the misclassified examples not by increasing the instance weight which is adopted by Adaboost [86] but by a different method.

From equation (7.7), we find that minimizing the second term $C\sum_{(i=1)}^{l} W_i\xi_i^n$ minimizes the associated error. The weight factors $C\sum_{i=1}^{l} W_i$ specify what tradeoff we are willing to tolerate between maximizing the margin and minimizing the error. Increasing the tradeoff parameters $\sum_{i=1}^{l} W_i$ of one class will increase the distance between the margin and the

Algorithm 8 IB_SVM algorithm

1: Given: $(x_1, y_1), \cdots, (x_m, y_m)$ where $x_i \in X, i = 1, \cdots, M, y_i \in Y = \{-1, +1\}$.
2: Choose an instance_weighted SVM classifier as the base learner.
3: initialize the hyper-parameter distribution $W_1(i)$ for it.
4: **for** $t = 1, \ldots, T$ **do**
5: Train the base learner using the hyper-parameter distribution W_t.
6: Get a hypothesis $h_t : \chi \to \mathbb{R}$.
7: Choose $\alpha_t \in \mathbb{R}$.
8: Update:

$$W_{t+1}(i) = \frac{W_t exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{7.20}$$

where Z_t is a weight normalization factor (chosen so that W_{t+1} will be a distribution).
9: **end for**
10: Output the final hypothesis:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{7.21}$$

slack variables of this class and will reduce the misclassified instances of this class. However, this can also increase the misclassified instances of the other class. The IB_SVM's goal is to obtain the optimal tradeoff by tuning the hyper-parameter distribution $W_t(i)$.

- Choosing α_t

In the original Adaboost algorithm, the weight updating parameter is expressed as

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \tag{7.22}$$

where ϵ_t is the error of hypothesis h_t on the instance x_i in the original boosting algorithm. Here we choose the balanced error

$$\epsilon'_t = \sum_{i:h_t(x_i) \neq y_i} |W_i / R_i| \tag{7.23}$$

instead of ϵ_t in this algorithm where R_i denotes the imbalance ratio ($R_i \in (0, 1], i = 1, \cdots, l$). The reason we choose ϵ'_t instead of ϵ_t is that for the class imbalanced problem, the balanced error of hypothesis h_t is a more unbiased evaluating method than the error of hypothesis h_t. Which means the evaluation on the majority class and minority class testing instances is asymmetric. For the initialized value of $W_1(i)$, we set

$$W_1(i) = R_i = \begin{cases} N^{min}/N^{maj}, \text{ if } y_i = Label^{min} \\ 1, \text{ if } y_i = Label^{maj} \end{cases} \tag{7.24}$$

where N denotes the number of instances, "min" denotes "minority" and "maj" denotes "majority".
The initial values of $W_1(i)$ are set to be different for majority and minority class. When $t = 1$, the IB_SVM is the same as the Weighted_SVM.

- Computational complexity analysis

If m is the size of the training set, training an SVM takes $O(m^2)$ with a general purposed QP solver [21]. Suppose the IB_SVM algorithm takes n iterations, the total time complexity of IB_SVM algorithm is $O(nm^2)$). Since n is always set as a constant (generally it is 10) in our experiments, the time complexity of IB_SVM algorithm is always acceptable.

- Benefits from diverse multiple learners

Ensemble approaches like Boosting [86] reduce the generalization error of machine learning systems by building and aggregating diverse multiple classifiers. However they suffer from the limitation of their basic theoretical property—they are only suitable for improving the predictive performance of unstable learners [66]. The class imbalanced problem exploits the "instability"of almost all of the learners [66], which gives us a chance to build and aggregate some diverse multiple stable classifiers such as SVM. By assigning each instance a regularization parameter for the hyper-parameter C, we can successfully build diverse sources of training dataset. By successive reweighting of the training set where current weight depends on the previous classifier's performance, IB_SVM grows an ensemble of classifiers without decreasing prediction accuracy of base learners.

- Analysis

Theorem 1 *Suppose the base learning algorithm Instance-weighted SVM, when called by IB_SVM, generates hypotheses with balanced errors* $\epsilon_1, \cdots, \epsilon_t$*.(as defined in Eq.(7.23)) Then the balanced error* $\epsilon = Pr_i[h_f(x_i) \neq y_i]$ *of the final hypothesis* h_f *output by IB_SVM is bounded above by*

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}.$$

Define the weight vector: $w_i = W(i)$ for $i = 1, \cdots, M$. $\beta_t = \epsilon_t/(1-\epsilon_t)$. The update

rule given in step 8 in algorithm 8 implies that

$$\sum_{i=1}^{M} w_i^{t+1} = \sum_{i=1}^{N} w_i^t \beta_i^{1-|h_t(x_i)-y_i|}$$

$$\leq \sum_{i=1}^{M} w_i^t (1-(1-\beta_t)(1-|h_t(x_i)-y_i|))$$

$$= (\sum_{i=1}^{M} w_i^t)(1-(1-\epsilon_t)(1-\beta_t))$$

The final hypothesis h_f, makes a mistake on instance i only if

$$\prod_{t=1}^{T} \beta_t - |h_t(x_i) - y_i| \geq (\prod_{t=1}^{T} \beta_t)^{-1/2}$$

The final hyper weight on any instance i is

$$w_i^{T+1} = W(i) \prod_{t=1}^{T} \beta_t^{1-|h_t(x_i)-y_i|}$$

Combining the two Eqs above we can get the lower bound of the sum of the final hyper weights by getting the sum of the final weights of the examples on which h_f is incorrect:

$$\sum_{i=1}^{M} w_i^{T+1} \geq \sum_{i:h_f(x_i)\neq y_i} w_i^{T+1}$$

$$\geq (\sum_{i:h_f(x_i)\neq y_i} W(i))(\prod_{t=1}^{T} \beta_t)^{-1/2}$$

$$= \epsilon \cdot (\prod_{t=1}^{T} \beta_t)^{-1/2}$$

Then from above we can get:

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}.$$

This gives the proof of Theorem 1.

7.3.4 Experimental setup and results

In our experiments we use the same datasets employed in chapter 3. In our experiments, we choose a 5×5 fold cross validation strategy. In this strategy, each data set is divided

equally into five folds. One of the folds is taken as testing dataset while the other folds are regarded as training dataset.

We chose the LibSVM [21][57] as the algorithm for the support vector machine classification. Modifications were made to the original source code based on the LibSVM license [21]. MI_fusion which is developed in chapter 4 is combined with IB_SVM here for imbalanced MI classification.

For single-instance learning, the IB_SVM algorithm and related experiments have already been presented in our related work [106]. With the help of MI_fusion, now we can apply the IB_SVM on imbalanced MI datasets and compare our method with Morik et al. [75] and Veropoulos et al. [99]'s algorithms on different norms. Morik et al.[75] and Veropoulos et al.'s algorithms [99] are denoted as weighted_SVM (1-norm and 2-norm). The experimental results are presented in Table A.5. Four kernels (linear, RBF, sigmoid and polynomial) are chosen in this experiment. The hyper-parameters of all SVMs are set using the Cross-Validation and Grid-Search methods [57]. Here we list the kernel with the best result for all SVMs.

Table 7.1: Experiment result using the statistical test method

	Weighted_SVM		IB_SVM	
	1-norm	2-norm	1-norm	2-norm
Accuracy	0-2-1	0-2-1	2-1-0	3-0-0
AUC	0-2-1	0-2-1	2-0-1	2-0-1
Total Wins	-4	-4	3	5

We listed the experimental results in Table A.5 and summarized the best results for each algorithm in Table 7.1. For each data set, we pick up the winners on Gmean and AUC respectively for all kernel methods and norms. From Table 7.1 we can get the conclusion that on most datasets we tested in our experiment, IB_SVM has better performance than weighted_SVM with the same kernel function.

7.4 Conclusion and Future work

In this chapter, an adaptive instance weighted boosting SVM combining with MI_fusion method is proposed for the imbalanced MI problem. We have extended Morik et al. [75] and Veropoulos et al.'s algorithm [99] by choosing a more general parameter setting: each instance is assigned a regularization weight hyper-parameter W_i. This method gives us more flexibility than only adjusting the weights of the false positives vs. the false negatives. Experimental results show that this method can improve the prediction performance of the classification on both measurements of Accuracy and AUC [23]. Our future work will include hyper-parameters optimization including the original cost factor C and the kernel parameters. Moreover, the relationship of the process of hyper-parameter optimization and the performance of classifiers will also be investigated.

Chapter 8

Case study on detecting mine like objects

As introduced in chapter 1, MI algorithms can be chosen as the Automatic Target Recognition (ATR) methods to detect possible objects (such as MLOs) or regions of interest in sonar imagery. Since many of the sonar images of the same object are taken from different sonar passes, there are multiple images of the same object at different ranges and aspects of the sonar.

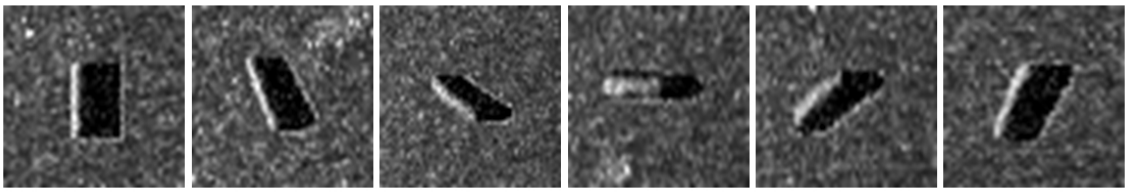

Figure 8.1: Multiple images of one target in sonar imagery for the detection of mine like objects (MLO)

Figure 8.1 gives an example of multiple images of one target. It is anticipated that the additional information obtained from additional images of an object should improve the classification performance over single-aspect classification. In this case, the MLOs classification on multiple images of an object is defined as a binary MI learning problem:

Input: Given a set of bags $X_i, i = 1, \cdots, N$, where each bag represents an object which might or might not be an MLO. A bag consists of an arbitrary number of instances which have information about the multiple sonar images of the object:

$$X_i = \left\{x_i^1, x_i^2, \cdots, x_i^{n_i}; Y_i\right\}, i = 1, \cdots, N, Y_i \in \{-1, +1\} \tag{8.1}$$

where each instance $x_i^{n_i}$ is an M-tuple of attribute values belonging to a certain domain or instance space $\mathbb{R}$. There is a function c that classifies individual bags as +1(MLO) or -1(non-MLO).

Output: a model that predicts whether a new example, which consists of an arbitrary number of multiple sonar images of the object, falls into MLO category or not.

8.1 Experimental results on MLO datasets

The MLO datasets are from the MLO detection project, a real world application. The sonar data used in this study was provided by the Ocean Systems Laboratory, Heriot-Watt University, Edinburgh, UK. The sonar images were collected by an AUV fitted with side-scan sonar on Loch Earn (Scotland) in November, 2010. In MLO classification, sonar images collected by AUVs convey important information about the underwater conditions. Raw sonar images are pre-processed before machine learning algorithms are applied.

8.1.1 Data preprocessing

In mine countermeasure missions (MCM), sonar images collected by AUVs will convey important information about the underwater conditions. Raw sonar images have to be properly pre-processed before machine learning algorithms are applied. How to properly process the sonar images will have a significant impact on the subsequent MLOs detection and classification stages.

In the MCMs, a large part of sonar images collected by AUVs represent the backgroundseabed. In MLOs detection and classification, we are more interested in the object that lies on the seabed rather than the background. Therefore, the areas from the images with only background information can be simply discarded.

The first step of this classification task is the segmentation of the sonar images. Image segmentation is a widely used image processing technique to detect target objects and segment the original images into small pieces that contain the target objects.

In this step, our goal is to delete image data that contain only background information and reduce the amount of data to be processed. Therefore whether the size, shape and location of the target object are accurately found is not a main concern in this step. The foreground objects are assumed to have a more complex texture than the seabed. Thus, the foreground object areas are obtained by using local range and standard deviation filters.

In sonar image segmentation, edges are significant local changes of intensity in an image. Edge detection is often used in order to extract the pieces of MLO objects from the background of seabed. Discrete differencing is a local edge enhancement technique. It is used to sharpen edge elements in an image by discrete differencing in either the vertical or horizontal direction, or in a combined fashion. In 2D images, edges are expressed using partial derivatives. Points which lie on an edge can be detected by detecting the maximum of gradient.

The gradient is a vector which has certain magnitude and direction:

$$\nabla I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix} \tag{8.2}$$

$$magn(\nabla I) = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2} = \sqrt{u^2 + v^2} \tag{8.3}$$

$$dir(\nabla I) = \tan^{-1}(v/u) \tag{8.4}$$

The magnitude of gradient is usually approximated by:

$$magn(\nabla I) \approx \mid u \mid + \mid v \mid \tag{8.5}$$

The gradient can be approximated by finite differences:

$$\partial I/\partial x = (I(x + h_x, y) - I(x, y))/h_x = I(x + 1, y) - I(x, y), (h_x = 1) \tag{8.6}$$

$$\partial I/\partial y = (I(x, y + h_y) - I(x, y))/h_y = I(x, y + 1) - I(x, y), (h_y = 1) \tag{8.7}$$

Here we choose the Sobel method as the edge detection method.

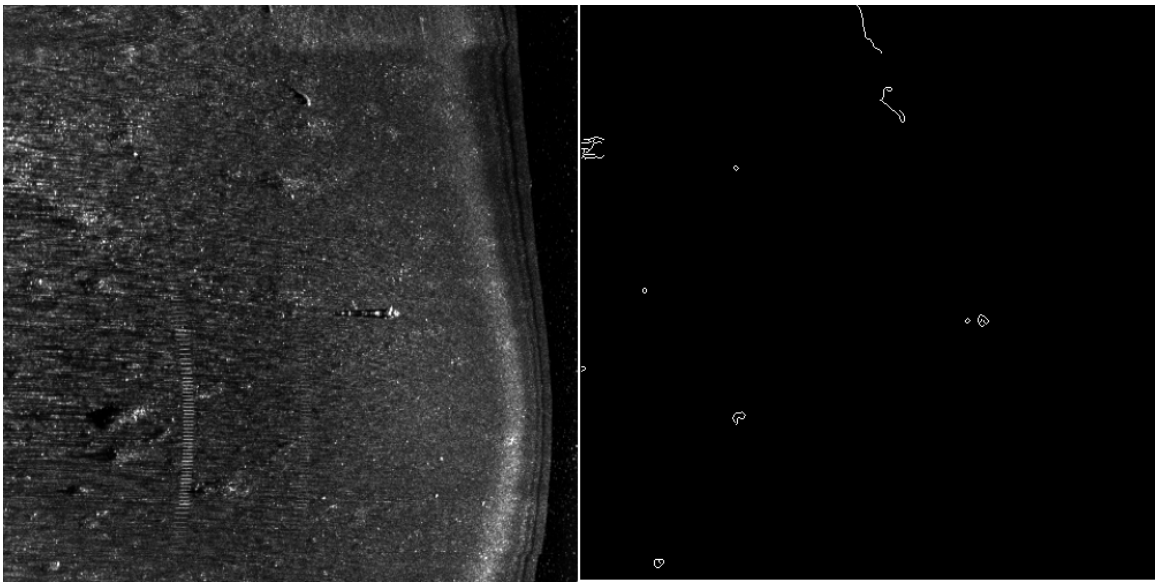

Figure 8.2: Example of an image processing result on an image provided by the Ocean System Lab, Heriot-Watt University

Figure 8.2 illustrates the extraction of foreground objects from a sonar image which was provided by the Ocean Systems Lab, Heriot-Watt University. Areas that do not have a reasonable size will be ignored.

The objective of the image processing procedures at this point is data reduction rather than MLOs detection. Thus, a relatively high false alarm rate is acceptable.

The next step of this image processing task is the segmentation of the sonar images into three distinct regions: highlight or target echo (sound scattered by the target by active sonar), shadow (regions of low acoustic energy created by an object or seabed feature blocking the sound propagation) and background or seabed.

For object detection tasks, an object should be detected through a single image, no matter where and how it lies on the seabed. Therefore, the features used should be robust to the location and orientation of the object. The grayscale histogram, a simple but informative statistical feature, is considered. In many image recognition systems, many

complex features are used, but such features will inevitably increase the computational complexity, impeding the real time detection. The histogram is easy to calculate and robust to rotation. The distribution of the grayscale value can be well described by this feature.

The size of each sonar image used in our study is 500×1024, so there are $512,000$ pixel values in each raw image. In order to translate this image information into data (which can be processed by machine learning algorithms), we draw the histogram for the final image. As in the unit 8 image, the brightness of a pixel falls into the region $0 - 255$. This region is divided into 16 bins. The number of the pixels falling into each bin are counted and normalized by the total number of the pixels. These 16 numbers are taken as the features of the data. The advantage of this method is that the features are robust to the variation of the location and orientations of the objects. The grayscale histograms are normalized to the frequency that a pixel value falls into each bin. The MLOs are labeled as the positive examples.

8.1.2 Comparison of different technologies on class imbalanced MLO datasets

Table 8.1: Details of datasets

Datasets	# bags	# attribute	# min bags	% min bags	# min instances	% min instances
MLO_1	561	16	58	10.34	116	10.34
MLO_2	555	16	64	11.53	144	12.18
MLO_3	425	16	65	15.29	158	17.67
MLO_4	762	16	63	8.27	132	8.34

The 4 MLO datasets we used in this experiment are listed in Table 8.1. The percentage of minority bags varies from 8.27% to 15.29%. In this experiment we compare different technologies presented in chapter 6 on MLO datasets and the experimental results are listed in table A.2. The result shows that both of Ab(1,1) and Ab(1,2) have the best performance on MLO datasets. We also compare the methods presented in chapter 7 on these data and the experimental results are listed in table A.6. The result shows that with MI_fusion, IB_SVM can improve the performance slightly comparing with SVM_weight.

8.1.3 Classification on multiple images of object

In the experiments, we study the classification performances as a function of the number of aspects and compare the experimental result using different MI classifiers.

The binary datasets used in these empirical studies are described in Table 8.2. The negative examples denote the non_MLOs and the positive examples denote the MLOs. In Table 8.2, dataset MLO_5 has 360 objects while the number of MLOs (positive) and

non_MLOs (negative) are both 180. Each object has three images so we can study the classification performances as a function of the number of images.

Table 8.2: Details of dataset MLO_5

Datasets	#objects	#attribute	#positive examples	#negative examples
MLO_5	360	16	180	180

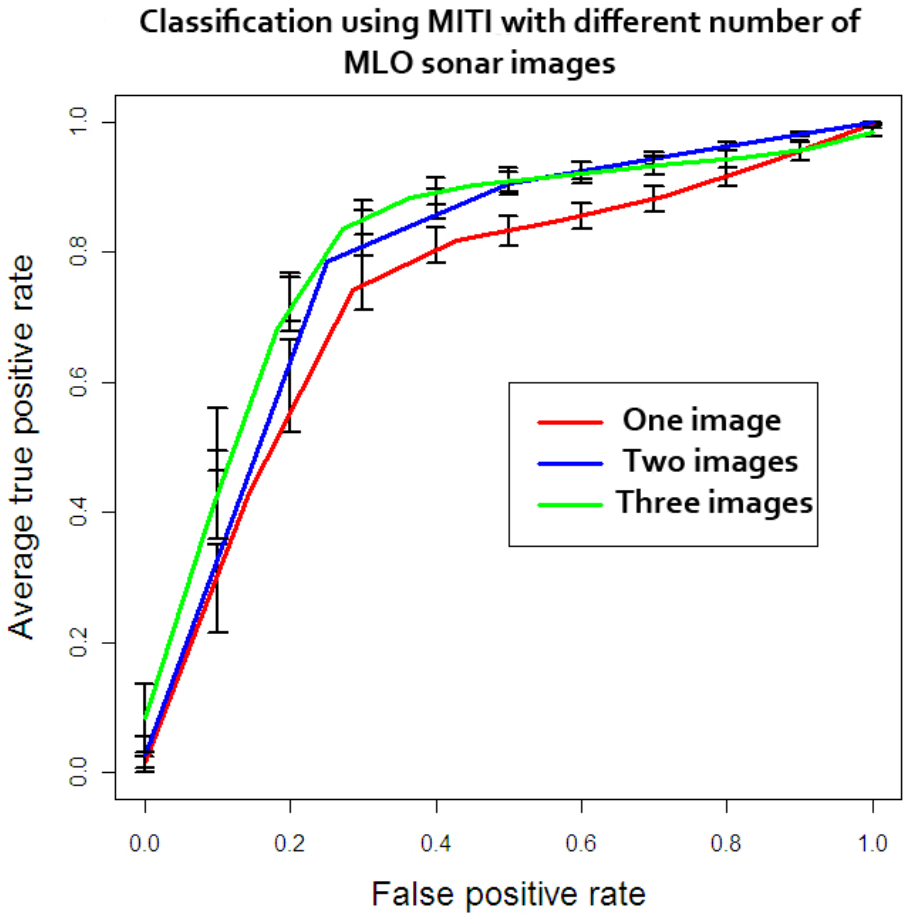

Figure 8.3: Classification performances as a function of the number of images using MITI as the classifier

ROC curves are chosen as the measure technique for the classification and the experimental results are shown in Figure 8.3 and Figure 8.4.

Figure 8.3 shows the ROC curves as a function of the number of aspects using MITI as the classifier and Figure 8.4 shows the ROC curves using MI_fusion with decision tree as the classifier. We can find that for both classifiers, with more images used for classification, the performance is better.

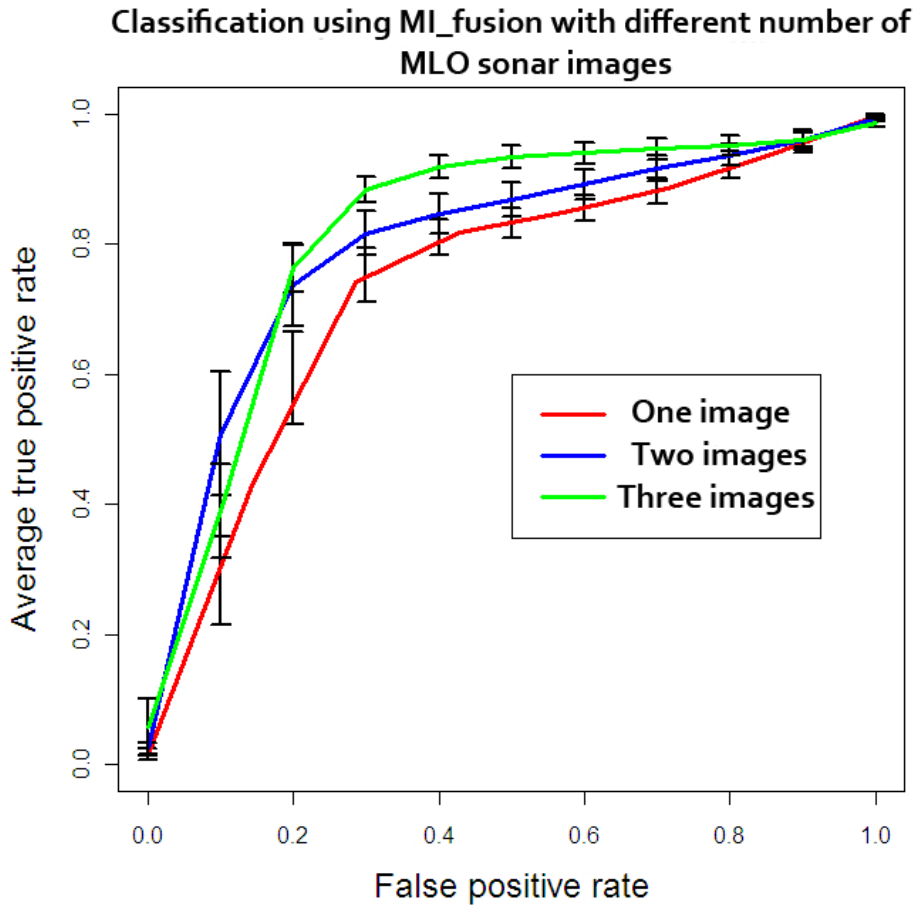

Figure 8.4: Classification performances as a function of the number of images using MI_fusion with decision tree as the classifier

8.1.4 Classification on different shape of MLOs with multiple images

After making a classification of MLOs and non_MLOs, we can keep on making a classification on what kind of shape the MLO belongs to. MLO_6 is in Table 8.3 is chosen as the dataset for this experiment. MLO_6 has 279 objects. We have three different shapes of MLOs which are cylinder, manta and wedding_cake shapes and each shape has 93 objects respectively.

Table 8.3: Details of dataset MLO_6

Datasets	#objects	#attribute	#cylinder	#manta	#wedding_cake
MLO_6	279	16	93	93	93

This study includes two parts, the first one is using tree method as classification algorithm in this study and the second part is using Logistic Regression instead. For each method, single-aspect and multi-aspects classification are studied respectively. For single-aspect classification, each object has one image. For multi-aspects classification, each object has more than one image and we choose the number of images randomly between two and three.

Tree method

Table 8.4 to Table 8.6 show the confusion matrices resulting from single-aspect classification using decision tree, multi-aspects classification using MITI and multi-aspects classification using MI_fusion with decision tree respectively.

Table 8.4: The Confusion Matrices Resulting From Single-Aspect Classification Using Decision Tree

(a)	Single Aspect		
	Cylinder	Manta	Wedding_Cake
Cylinder	58.9	24.9	16.2
Manta	11.9	73.5	14.6
Wedding_Cake	11.4	16.2	72.4

Table 8.5: The Confusion Matrices Resulting From Multi-Aspects Classification Using MITI

(b)	Multi-Aspects		
	Cylinder	Manta	Wedding_Cake
Cylinder	80.6	11.8	7.5
Manta	8.6	83.3	8.1
Wedding_Cake	10.2	3.2	86.6

Table 8.6: The Confusion Matrices Resulting From Multi-Aspects Classification Using MI_fusion with Decision Tree

(c)	Multi-Aspects		
	Cylinder	Manta	Wedding_Cake
Cylinder	73.6	17.2	9.2
Manta	4.8	84.4	10.8
Wedding_Cake	8.6	7.0	84.4

Logistic Regression method

Table 8.7 to Table 8.9 give the confusion matrices resulting from single-aspect classification using Logistic Regression, multi-aspects classification using MILR and multi-aspects classification using MI_fusion with Logistic Regression respectively.

Table 8.7: The Confusion Matrices Resulting From Single-Aspect Classification Using Logistic Regression

(a)	Single Aspect		
	Cylinder	Manta	Wedding_Cake
Cylinder	58.9	22.7	18.4
Manta	30.8	55.1	14.1
Wedding_Cake	4.9	15.1	80.0

Table 8.8: The Confusion Matrices Resulting From Multi-Aspects Classification Using MILR

(b)	Multi-Aspects		
	Cylinder	Manta	Wedding_Cake
Cylinder	72.6	16.7	10.7
Manta	10.3	76.3	5.4
Wedding_Cake	8.1	3.8	88.1

Table 8.9: The Confusion Matrices Resulting From Multi-Aspects Classification Using MI_fusion with Logistic Regression

(c)	Multi-Aspects		
	Cylinder	Manta	Wedding_Cake
Cylinder	69.4	19.9	10.7
Manta	16.7	75.3	8.0
Wedding_Cake	7.0	8.1	84.9

From these classification results we can see that the classification performance, both on using the MI framework and data fusion framework, were improved by using more "images" in the classification.

8.2 Conclusion

In this chapter, we have demonstrated the performance improvements in the classification of side scan sonar images obtained by using feature sets corresponding to multiple sonar images of the same object. In our experiments, we have transformed the multiple aspects classification problem into a multiple-instance learning problem and we have found that the MI_fusion method can be an alternative and effective framework for the multiple-instance learning problem.

The experimental results of section 8.1.2 show that the two proposed solutions in chapter 6 (i.e. the classifier independent method–sampling technique and the classifier dependent method–cost-sensitive boosting technique) can improve the performance of the base learner for class imbalanced MI learning. The results also show that the weighted updating strategy of cost-sensitive boosting algorithms increases the weights on the misclassified bags from the minority class more than it does on those from the majority class. Similarly, it decreases the weights on correctly classified bags from the minority class less than on those from the majority class. Moreover, the experimental results of section 8.1.2 also show that the proposed approach in chapter 7 (i.e. the IB_SVM algorithm) can also be used as an alternative method in this application.

We have also studied the classification performances as a function of the number of aspects. From the experimental results we find that for both frameworks, the performance is enhanced when more images are used for classification.

Comparing the single aspect classification rates with two multiple-aspect approaches on different shapes, we see that collecting multiple images produces a significant increase in hit rate and a significant decrease in error rate for all mine shapes.

Our future works will involve applying the presented frameworks in the thesis on other multiple aspect learning applications and using more classifiers.

Chapter 9

Conclusions and future work

9.1 Extended Summary

In this thesis, we have presented a comprehensive and systematic analysis of two challenging and critical issues in the knowledge discovery and data mining field: the MI learning problem and the class imbalance problem in MI.

For the general MI problem, we have presented three novel frameworks: the Multi-View Two-level MI framework, the multi-kernel MI framework and the MI fusion framework.

For the imbalanced MI problem, we have also provided three novel solution frameworks for this problem: sampling technology, cost-sensitive boosting method and adaptive instance weighted support vector machines framework.

Based on these methods, we conducted experimental analysis using different learning algorithms with real-world benchmark datasets from a variety of application domains, and datasets from a real-world project. Experimental evidence derived from standard datasets was presented to support the improved performance of the proposed algorithms. We found that the presented algorithms are effective on dealing with the general MI problem and class imbalance problem respectively.

9.2 Thesis Contributions

In this section we will examine whether we achieved the expected contributions described in section 1.3.

A novel multi-view solution for the general MI problem

Under the *generalized MI assumption*, we introduced a multi-view, two-level-classification approach to deal with the MI problem in chapter 3. This multi-view, two-level classification (MV-TLC) strategy transforms a MI dataset into a multi-view, single meta-instance dataset, and learns from multiple views (the feature set) of this meta-dataset. The information acquired by view learners is then integrated to construct a final classification

model. Our empirical studies show that our method compares well to other popular MI classifiers.

A novel fusion solution for the general MI problem

In chapter 4 we presented a new MI assumption— fusion assumption model for the MI problem. The fusion assumption is based on the Bayesian theory of subjective probability. Under this assumption, a bag is a sample of an underlying population specific to the particular bag and a bag can be treated as a group of different information sources with different degrees of reliability. Under this assumption, we presented a MI_fusion algorithm based on the Dempster-Shafer (DS) theory [88]. Our experimental results show that the MI_fusion approach is competitive and takes more advantage on computational complexity than other MI approaches.

A Multi-kernel SVM algorithm for the general MI problem

In chapter 5 we formulated a novel method to solve the classification problem within the MI learning context using multiple kernel learning. To improve the classification precision of the SVM method with regard to the MI learning problem, this chapter introduced a multiple kernel learning method to the process of multiple instance learning. It proposed a new SVM model (MKMI), which is based on the two-level MI model. Computational results on a number of datasets indicate that the proposed algorithm is competitive with other SVM methods.

A solution using Re-sampling Techniques for the class imbalanced MI problem

There are very few related discussions about the multi-instance class imbalance problem. In chapter 6 we proposed two algorithms using the sampling technology: Instance_SMOTE and Bag_SMOTE. The experimental results show that the proposed solutions can improve the performance of the base learner for the class imbalanced multi-instance learning. The results also support our expectation that the class imbalance problem occurs at the bag-level, so the solution should also be found at the bag-level.

A Cost-sensitive Boosting algorithm for the class imbalanced MI problem

To deal with the class imbalanced MI problem, in chapter 6 we also provided a cost-sensitive boosting algorithm. We proposed a generalized version of the Cost-sensitive Boosting-MI algorithm. Our goal is to reduce the bias inherent in the learning procedure due to the class imbalance, and to increase the weights for the minority class in each round of boosting. A systematic study on cost and the coefficient parameters were made in section 6.5. The coefficient parameters of the cost items were found to be related to the classification performance of class imbalanced MI learning. The combination of these two coefficient parameters gives us opportunities to achieve better classification performance. The experimental results show that the proposed solutions can improve the performance of the base learner for class imbalanced multi-instance learning. Moreover, it even has a better performance than the re-sampling techniques for many data sets.

An instance-weighted SVM algorithm for imbalanced MI problem

In chapter 7 we applied an instance-weighted variant of the SVM with both 1-norm and 2-norm format to deal with the imbalanced MI learning problem. We developed an

asymmetric boosting method on the weights of the trade-off parameters to optimize the instance-weighted SVM. This method gives us more flexibility than only adjusting the weights of false positives vs. false negatives. Experimental results show that this method can improve the prediction performance of the classification on both measurements of Gmean and AUC.

A case study on MLO detection

In chapter 8 we conducted a case study on detecting mine like objects. The algorithms we have presented were applied on this application successfully.

9.3 Future Research

Two objectives of this research are:

1) Providing practical guidance to machine learning practitioners when they want to use new algorithms or build classifiers from MI datasets

2) Providing researchers with possible directions for study on class imbalanced MI datasets.

Our presented work and future work involve around these two objectives. Our future work includes:

1. **For solving the general MI problem**, the MV-TLC framework, the Multiple-kernel MI SVM framework and the data fusion framework have been presented. The performance of these frameworks needs more experimental results to compare with existing MI learning approaches. In our future work, we will make sufficient experiments on comparing these frameworks with other existing MI learning frameworks.

   Moreover, for the MV-TLC algorithm, after the first step, we can build a new model combining the results obtained with all the models discovered before for the second step which learns bag-level concepts. Here we plan to apply another MI learning algorithm-MILES [26] to this transformed data.

   MILES [26] embeds bags into a single-instance feature space based on similarity scores, and applies the 1-norm support vector machine algorithm to the transformed dataset. It uses the instances in the training bags as candidates for target points. A feature-space mapping is defined, where every attribute represents the closeness of an instance to a candidate target point (i.e. training instance). Each training bag is mapped into this space (with class labels appended), and a single-instance base learner is built on the transformed dataset. At testing time, bags are similarly mapped into the instance-based feature space, and classification predictions are made by the single-instance base learner. Moreover, a further (optional) refinement to the algorithm is to use attribute selection to try to eliminate the attributes that do not contribute to the instance-level classification problem learned by the first level classifiers.

2. **For solving class imbalanced MI problem,** we have presented cost sensitive boosting algorithms at the bag level and bag level SMOTE algorithms which have been shown to be effective. We have also presented an instance-weighted boosting SVM MI framework. In the future, it would be worthwhile to investigate if the methods presented are suitable for datasets with extremely high imbalance ratios. Moreover, tree methods are used as the base learners for the techniques presented in this thesis. Although the presented sampling algorithms and cost-sensitive boosting algorithms are learner independent, the setup of the parameters could be influenced by choosing base learners with different technologies. The relationship between the presented techniques and different base learners will be investigated in our future work. We also intend to research other applications of the presented algorithms for MI classification, and investigate other related algorithms.

   On the other hand, a MV-TLC framework for the class imbalanced MI problem could also be researched as the future work. In MV-TLC, single-instance learning methods are used in both of level one and level two. General solutions for the single instance class imbalanced problem can be used in this framework. However, we would need to implement related algorithms and complete more experiments in the next steps.

Moreover, we would need to apply the developed algorithms to other applications and different types of MI datasets, such as stream data, multi-label MI data, multi-view MI data and large-scale MI datasets. The related research can also be extended to other learning tasks such as ranking, unsupervised learning and semi-supervised learning.

APPENDICES

Appendix A

Table A.1: Comparison of all presented algorithms with the base learner (MITI[14]) on the benchmark datasets

Datasets	Measures	Base_ Learner	Instance_ SMOTE	Bag_Over Sampling	Bag_ SMOTE	Adaboost	Ab(0,1)	Ab(1,0)	Ab(1,1)	Ab(1,2)	Ab(2,1)	Ab(2,2)
Elephant	TPR_{min}	28.7±6.0	19.3±3.3	41.3±4.4	41.3±4.0	31.3±3.6	35.3±3.3	29.3±1.8	60.7±4.7	71.3±8.9	74.7±8.0	74.7±6.2
	TNR_{min}	94.5±1.2	94.7±1.3	93.7±1.2	93.5±1.5	98.5±1.2	95.2±1.5	97.2±1.2	82.3±1.6	68.0±4.7	64.7±5.7	53.7±5.2
	Gmean	52.0±5.5	42.8±3.6	62.2±3.4	62.2±3.0	55.6±3.2	58.0±2.7	53.4±1.8	70.7±2.5	69.6±5.2	69.5±2.1	63.3±4.1
	Precision	83.2±4.5	78.5±3.9	86.6±3.0	86.4±3.0	95.6±3.5	88.1±3.4	91.2±3.7	77.4±1.9	68.9±4.4	68.0±2.0	61.7±3.7
	Recall	60.7±7.1	48.6±5.3	73.5±3.4	73.6±3.2	64.2±4.0	68.4±3.1	62.3±2.0	85.8±2.4	90.4±3.8	91.8±3.1	91.9±2.8
	F-measure	70.2±6.3	60.0±4.6	79.5±3.1	79.5±2.9	76.9±3.1	77.0±2.2	74.0±2.2	81.4±1.8	78.2±4.1	78.2±1.4	73.8±3.3
Fox	TPR_{min}	5.6±1.3	6.3±2.6	23.8±4.8	27.8±5.3	11.1±6.3	19.8±4.5	15.1±4.0	59.5±2.3	60.3±9.5	68.3±7.9	66.7±4.8
	TNR_{min}	96.7±0.8	97.3±1.3	91.2±1.4	91.2±1.2	96.5±1.2	93.8±0.9	95.5±0.7	67.2±2.6	55.2±3.8	53.5±5.5	51.2±6.6
	Gmean	23.2±2.4	24.9±4.5	46.6±4.7	50.3±4.6	32.7±9.9	43.1±4.5	37.9±5.1	63.2±5.5	57.7±5.5	60.4±5.0	58.4±2.2
	Precision	62.4±3.7	68.5±3.5	72.2±4.0	75.2±4.0	70.0±5.2	75.2±4.2	76.0±4.4	64.0±4.2	57.0±4.9	59.3±4.4	57.9±2.0
	Recall	21.6±3.9	23.4±6.9	58.8±6.8	63.8±5.4	34.6±5.4	52.9±5.8	44.9±7.9	86.8±4.8	87.2±4.4	90.7±3.1	90.3±1.9
	F-measure	32.1±4.2	34.9±8.5	64.8±5.5	69.1±4.5	46.3±7.3	62.1±5.4	56.5±7.5	73.7±4.5	69.0±4.6	71.7±4.0	70.5±1.5
Tiger	TPR_{min}	31.4±2.1	38.5±6.4	48.1±5.1	55.1±3.8	49.4±4.5	54.5±4.5	52.6±5.1	65.4±3.8	73.1±7.7	70.5±3.8	82.7±4.5
	TNR_{min}	95.5±1.8	95.0±1.0	93.8±1.2	89.5±1.5	96.5±1.0	93.7±1.8	96.0±1.0	86.3±2.0	72.3±2.1	71.5±3.0	59.5±5.0
	Gmean	54.8±1.7	60.4±4.9	67.2±3.8	70.2±2.0	69.0±3.2	71.4±2.3	71.0±3.6	75.1±2.3	72.7±3.4	71.0±1.5	70.1±3.7
	Precision	87.8±3.8	88.3±2.5	88.4±2.6	84.1±1.6	93.3±1.8	89.8±2.2	92.9±1.8	82.8±2.3	72.4±1.3	71.3±1.6	67.3±3.3
	Recall	63.7±2.3	69.8±5.6	77.8±3.9	82.4±2.2	78.7±2.9	82.0±2.6	80.8±3.2	87.8±1.9	92.9±3.3	90.1±1.6	94.8±1.6
	F-measure	73.8±2.0	78.0±3.9	82.7±3.0	83.2±1.0	85.4±2.2	85.7±1.0	86.4±2.3	85.2±1.6	80.6±2.1	79.6±1.0	78.7±2.6
M_atoms	TPR_{min}	37.7±2.0	35.3±6.6	48.4±1.9	55.2±2.8	43.3±1.3	51.2±2.0	51.2±4.0	70.6±6.3	81.3±3.6	86.1±1.3	88.9±4.0
	TNR_{min}	95.6±0.4	95.2±1.1	87.1±1.2	92.9±0.4	95.7±0.8	87.2±1.3	86.5±0.9	76.8±2.4	58.4±3.2	59.2±3.5	41.7±2.5
	Gmean	60.0±1.6	58.0±5.4	64.9±1.1	71.6±1.7	64.3±1.1	66.8±1.4	66.6±2.6	73.7±2.2	68.9±2.6	71.4±2.2	60.9±1.6
	Precision	89.5±1.1	87.7±2.3	78.9±1.8	88.6±0.6	91.0±1.5	80.0±1.7	79.0±1.9	75.3±0.7	66.2±2.1	67.9±1.9	60.4±0.9
	Recall	64.2±1.9	61.3±6.6	73.6±1.4	78.5±1.9	69.4±1.1	75.7±1.5	75.5±2.9	87.5±3.3	92.8±1.6	94.9±0.5	95.9±1.6
	F-measure	74.8±1.6	72.1±5.3	76.2±1.2	83.3±1.2	78.7±1.1	77.8±1.2	77.3±2.2	80.9±1.3	77.3±1.9	79.1±1.3	74.1±0.9
M_bonds	TPR_{min}	30.5±3.8	28.6±2.9	60.0±1.9	56.7±4.4	48.6±1.9	60.5±6.0	59.0±4.8	74.8±3.5	76.2±5.1	77.1±3.8	78.1±2.9
	TNR_{min}	95.6±0.9	96.4±0.7	91.3±0.4	92.0±0.8	95.7±0.4	88.8±0.5	88.9±0.7	79.3±3.1	72.3±2.1	72.0±2.9	70.7±3.1
	Gmean	54.0±3.5	52.5±2.7	74.0±1.3	72.2±3.1	68.2±1.4	73.3±3.5	72.5±2.8	77.0±2.6	74.2±2.4	74.5±1.7	74.3±2.7
	Precision	87.2±2.9	88.8±1.9	87.3±0.9	87.5±1.6	91.9±0.6	84.3±1.0	84.2±1.0	78.4±2.5	73.3±2.0	73.4±1.6	72.7±2.3
	Recall	60.7±4.2	58.6±3.5	84.2±1.1	82.2±2.7	77.1±1.4	84.3±3.5	83.6±2.7	91.3±1.5	91.8±2.1	92.2±1.7	92.7±1.1
	F-measure	71.6±3.8	70.6±2.9	85.7±1.0	84.8±2.2	83.8±1.0	84.3±2.1	83.9±1.7	84.3±1.9	81.5±1.7	81.8±1.2	81.5±1.9
M_chains	TPR_{min}	17.9±3.3	18.5±1.2	45.1±5.6	43.8±4.3	27.8±3.1	46.3±4.9	40.1±4.1	67.3±5.8	68.5±4.3	71.0±5.6	73.5±4.3
	TNR_{min}	97.3±0.4	97.9±0.6	93.1±1.3	93.7±0.7	95.3±0.7	94.4±0.5	94.5±0.7	83.7±2.1	82.8±2.1	80.1±2.8	78.9±4.0
	Gmean	41.7±4.1	42.6±1.5	64.8±4.1	64.1±3.2	51.5±2.9	66.1±3.7	61.6±3.2	75.1±2.4	75.3±2.2	75.4±2.4	76.1±2.4
	Precision	86.1±4.2	89.8±2.3	86.4±2.7	87.4±1.4	85.4±2.6	89.0±1.9	87.9±1.8	80.6±1.2	80.0±1.5	78.2±1.8	78.0±3.4
	Recall	49.5±5.9	51.1±2.2	78.7±3.8	78.1±3.0	63.8±3.6	79.6±3.5	75.4±3.0	90.3±2.2	90.8±1.7	91.7±2.1	92.7±1.5
	F-measure	62.9±5.9	65.1±2.2	82.4±3.2	82.5±2.2	73.0±3.0	84.0±2.5	81.2±2.3	85.2±1.2	85.1±1.1	84.4±1.4	84.7±2.1
Component	TPR_{min}	41.0±2.5	44.8±1.7	47.2±2.5	54.5±4.5	45.1±2.8	60.4±2.8	50.7±3.2	82.6±3.0	86.1±2.1	89.9±3.8	88.2±5.6
	TNR_{min}	93.5±0.6	92.5±0.8	91.0±0.9	90.1±2.6	96.0±0.9	92.0±1.5	94.9±0.4	70.6±2.0	62.2±2.8	61.0±1.5	58.5±3.7
	Gmean	61.9±1.9	64.4±1.1	65.5±1.9	70.1±2.2	65.8±1.9	74.6±2.1	69.3±2.2	76.4±1.2	71.2±2.4	74.1±1.6	71.8±2.8
	Precision	86.3±1.2	85.7±1.1	83.9±1.9	84.9±3.2	91.9±1.6	88.3±2.2	90.7±1.1	73.8±1.0	69.5±1.9	69.7±1.1	68.1±2.2
	Recall	77.0±1.8	79.7±1.1	81.2±1.5	85.2±2.3	79.9±1.8	88.1±1.2	83.2±1.9	95.8±0.9	96.8±0.5	97.7±0.9	97.2±1.4
	F-measure	81.4±1.2	82.6±0.8	82.5±1.5	85.0±1.3	85.5±1.2	88.2±1.5	86.8±1.4	83.3±0.8	80.9±1.5	81.4±0.9	80.1±1.7
Function	TPR_{min}	41.1±2.8	58.1±5.3	56.5±2.8	61.4±6.1	51.6±3.7	61.8±4.3	58.9±2.0	79.7±4.3	84.6±3.3	85.4±2.4	85.4±3.3
	TNR_{min}	96.8±0.5	95.4±0.8	92.4±0.8	91.8±1.6	95.5±0.6	93.0±0.7	95.2±0.8	81.5±1.1	77.3±2.0	76.1±1.0	74.2±2.4
	Gmean	63.0±2.1	74.5±3.6	72.2±1.9	75.1±4.4	70.2±2.3	75.8±2.9	74.9±1.3	80.6±2.5	80.8±1.5	80.6±1.2	79.6±1.9
	Precision	92.7±1.2	92.5±1.5	88.1±0.1	88.1±3.0	92.0±0.7	89.7±1.3	92.5±1.1	81.1±1.4	78.8±1.6	78.2±0.9	76.8±1.9
	Recall	76.4±2.1	86.5±2.5	85.8±1.4	87.9±2.7	83.2±2.0	88.2±1.8	87.0±1.0	94.8±1.4	96.2±0.9	96.4±0.7	96.4±0.9
	F-measure	83.8±1.5	89.4±2.0	86.9±1.1	88.0±2.9	87.4±1.0	89.0±1.6	89.7±0.6	87.4±1.4	86.7±1.0	86.3±0.7	85.5±1.3
Process	TPR_{min}	33.9±3.8	48.9±3.6	41.4±3.8	51.6±5.4	38.7±2.2	53.2±6.5	47.3±1.4	71.5±1.8	76.3±2.5	78.0±2.7	80.1±2.9
	TNR_{min}	96.0±0.4	90.7±1.0	93.2±0.8	90.1±0.4	96.0±0.8	92.1±0.7	95.2±0.7	81.1±1.1	75.5±2.6	73.3±3.1	71.4±2.8
	Gmean	57.0±3.2	66.6±2.6	62.1±2.8	68.2±3.5	61.0±1.7	70.0±4.2	67.1±0.9	76.2±1.1	75.9±1.4	75.6±2.0	75.6±1.5
	Precision	89.4±0.8	84.0±1.6	85.8±1.9	83.9±1.4	90.7±1.5	86.8±2.0	90.9±1.3	79.1±1.0	75.8±1.7	74.5±2.4	73.8±1.4
	Recall	75.4±3.2	85.3±1.9	81.0±2.4	86.5±2.5	79.3±1.5	87.1±0.3	84.5±0.8	93.8±0.5	95.1±0.7	95.5±0.7	96.1±0.7
	F-measure	81.8±2.2	84.6±1.7	83.3±1.9	85.2±1.9	84.6±1.2	87.0±2.3	87.6±0.7	85.9±0.7	84.4±1.1	83.7±1.5	83.5±0.9

Table A.2: Comparison of all presented algorithms with the base learner (MITI[14]) on the MLO datasets

Datasets	Measures	Base_ Learner	Instance_ SMOTE	Bag_Over Sampling	Bag_ SMOTE	Adaboost	Ab(0,1)	Ab(1,0)	Ab(1,1)	Ab(1,2)	Ab(2,1)	Ab(2,2)
MLO_1	TPR_{min}	7.8±2.0	23.9±4.8	18.4±4.6	34.2±3.3	9.8±2.9	22.4±3.4	12.1±2.9	54.6±2.3	56.6±1.5	55.7±2.7	54.9±2.7
	TNR_{min}	97.8±0.2	93.5±0.5	90.8±1.3	85.9±1.2	97.2±0.2	93.1±0.4	94.9±0.4	75.8±1.4	75.0±1.8	72.7±2.4	75.6±1.9
	Gmean	27.5±3.6	47.2±4.6	40.9±5.1	54.2±2.7	30.8±4.6	45.7±3.6	33.8±3.9	64.3±1.8	65.1±0.6	63.6±2.3	64.4±1.5
	Precision	76.6±5.0	77.6±4.1	65.6±5.9	70.6±3.1	75.9±6.2	75.7±3.0	69.1±4.3	69.3±2.1	69.4±1.0	67.1±2.6	69.2±1.3
	Recall	41.4±6.9	72.0±4.5	64.7±7.2	81.6±2.4	47.1±8.3	70.6±4.6	53.2±6.6	91.2±0.7	91.9±0.5	91.6±0.9	91.3±0.9
	F-measure	53.7±7.1	74.7±4.3	65.2±6.5	75.7±2.6	58.1±8.1	73.1±3.8	60.1±5.6	78.7±1.6	79.1±0.6	77.5±2.0	78.7±0.9
MLO_2	TPR_{min}	43.9±5.9	59.4±2.1	47.7±4.7	59.6±4.5	45.1±2.3	68.8±3.6	59.4±1.6	84.4±3.1	85.2±1.3	83.6±3.4	84.4±1.0
	TNR_{min}	96.7±0.3	95.6±0.4	94.7±0.2	95.0±0.6	96.1±0.4	93.4±0.4	94.0±0.4	83.3±1.7	83.1±1.3	83.3±1.1	82.5±0.8
	Gmean	65.2±4.3	75.3±1.5	67.2±3.4	75.3±2.9	65.8±1.6	80.1±2.3	74.7±0.9	83.8±1.3	84.1±0.3	83.5±1.5	83.4±0.8
	Precision	92.9±0.8	93.1±0.8	89.9±1.2	92.3±1.0	92.0±0.7	91.2±0.9	90.9±0.3	83.5±1.1	83.4±0.9	83.4±0.6	82.8±0.7
	Recall	85.5±3.1	91.8±0.6	87.3±2.2	91.8±1.3	86.2±1.1	94.4±0.9	91.8±0.5	97.6±0.5	97.8±0.2	97.5±0.6	97.6±0.2
	F-measure	89.1±2.1	92.4±0.7	88.5±1.7	92.0±1.2	89.0±0.7	92.7±0.9	91.3±0.2	90.0±0.5	90.0±0.5	89.9±0.5	89.6±0.5
MLO_3	TPR_{min}	49.5±1.4	64.9±2.3	59.2±3.1	68.2±4.8	56.2±1.5	72.3±0.5	67.9±3.8	87.7±2.1	89.5±1.8	89.7±2.1	91.3±2.1
	TNR_{min}	95.9±0.5	94.4±0.6	91.9±0.5	94.3±0.9	95.4±0.4	90.7±0.7	91.9±0.6	82.7±1.7	79.5±3.3	79.1±1.9	77.6±2.7
	Gmean	68.9±1.1	78.3±1.5	73.8±2.1	80.2±2.8	73.2±1.0	81.0±0.6	79.0±2.3	85.2±1.3	84.4±1.8	84.2±1.2	84.2±1.7
	Precision	92.4±1.1	92.0±1.0	88.0±1.1	92.3±0.9	92.4±0.7	88.6±0.9	89.4±0.8	83.6±1.4	81.5±2.3	81.1±1.2	80.3±2.0
	Recall	84.4±0.7	91.1±0.8	88.9±1.2	92.1±1.8	87.6±0.7	93.5±0.2	92.1±1.3	97.5±0.5	97.9±0.4	98.0±0.4	98.3±0.4
	F-measure	88.2±0.8	91.5±0.8	88.4±1.1	92.2±1.1	90.0±0.6	91.0±0.5	90.7±1.0	90.0±0.9	88.9±1.4	88.8±0.8	88.4±1.2
MLO_4	TPR_{min}	45.0±3.7	50.8±1.6	48.1±2.8	57.4±1.9	49.2±2.1	67.5±3.4	54.2±3.5	91.3±1.3	89.7±2.1	88.4±1.4	79.4±2.1
	TNR_{min}	96.3±0.3	95.8±0.3	96.1±0.3	95.3±0.3	96.5±0.2	94.8±0.2	95.4±0.4	92.5±0.4	93.0±0.2	93.1±0.3	94.3±0.2
	Gmean	65.8±2.8	69.7±1.1	68.0±2.0	73.9±1.3	68.9±1.5	80.0±2.1	71.9±2.3	91.9±0.8	91.3±1.0	90.7±0.7	86.5±1.2
	Precision	92.2±1.2	92.3±0.6	92.4±0.8	92.4±0.5	93.4±0.4	92.8±0.6	92.2±0.7	92.4±0.5	92.8±0.2	92.7±0.3	93.3±0.3
	Recall	89.9±1.4	91.9±0.5	91.1±0.9	93.7±0.4	91.5±0.7	95.8±0.6	92.8±0.9	99.1±0.1	99.0±0.2	98.8±0.2	97.7±0.3
	F-measure	91.1±1.3	92.1±0.4	91.7±0.8	93.0±0.5	92.4±0.5	94.3±0.6	92.5±0.8	95.7±0.3	95.8±0.2	95.7±0.2	95.5±0.3

Table A.3: Experiment results (Accuracy by percent)

Classifier	Elephant	Fox	Tiger	M_atoms	M_bonds	M_chains	Musk1	Musk2
MV-TLC	82.5±2.7	68.0±1.5	79.0±1.9	79.7±2.1	86.2±1.2	85.2±0.8	84.8±2.9	83.3±1.1
MI_fusion(RF)	81.5±0.9	64.2±2.6	79.0±2.4	79.5±1.4	80.7±2.2	80.0±1.3	83.9±1.4	78.2±1.8
MKMI	84.7±0.4	57.8±2.2	82.9±0.4	79.2±0.9	85.2±1.9	83.0±0.8	82.4±2.4	81.4±2.6

Table A.4: Experiment results (AUC)

Classifier	Elephant	Fox	Tiger	M_atoms	M_bonds	M_chains	Musk1	Musk2
MV-TLC	91.4±1.7	69.0±1.5	81.9±1.9	86.4±1.3	91.2±0.6	89.8±0.9	90.9±2.4	87.2±2.3
MI_fusion(RF)	89.9±1.6	66.5±4.1	85.7±1.8	77.5±1.4	80.5±1.8	78.4±1.8	92.4±2.5	82.9±2.4
MKMI	84.7±0.4	57.8±2.2	82.9±0.4	74.6±1.0	84.3±0.5	80.1±0.8	82.3±2.5	82.3±2.1

Table A.5: Comparison of IB_SVM and weighted_SVM with MI_fusion on the benchmark datasets

Dataset	Kernel	Gmean				AUC			
		Weighted_SVM		IB_SVM		Weighted_SVM		IB_SVM	
		1-Norm	2-Norm	1-Norm	2-Norm	1-Norm	2-Norm	1-Norm	2-Norm
Elephant	Sigmoid	72.2±2.3	76.3±0.5	76.7±0.8	72.2±1.5	73.0±1.6	77.5±0.7	77.5±0.4	77.1±0.6
Fox	Sigmoid	44.6±1.3	35.1±3.4	44.7±2.3	50.7±2.1	45.2±3.1	35.2±4.2	44.8±3.7	51.4±2.8
Tiger	RBF	74.6±1.9	79.4±1.7	76.3±1.1	83.3±0.4	74.9±2.1	79.5±1.5	79.8±1.4	80.9±0.8
M_atoms	Polynomial	67.5±1.0	68.3±2.7	69.7±1.9	67.3±2.3	67.5±1.0	68.4±2.6	75.1±1.9	73.9±1.3
M_bonds	Polynomial	79.5±1.9	80.3±2.1	80.2±2.7	81.5±2.8	79.5±1.9	80.4±2.1	86.2±1.5	87.8±1.9
M_chains	Polynomial	80.0±1.6	78.4±2.6	77.7±3.9	79.3±1.8	80.1±1.6	78.7±2.5	84.0±2.1	84.4±2.2
Component	Linear	79.7±0.9	81.1±1.6	80.1±2.8	77.7±2.0	79.9±0.8	81.4±1.5	84.6±2.8	85.6±1.2
Function	Linear	84.8±1.6	83.2±2.0	78.6±2.2	81.2±1.2	85.0±1.5	83.5±1.8	85.0±1.7	87.2±0.8
Process	Linear	83.8±0.9	83.4±0.8	77.5±3.2	79.3±6.0	84.0±0.8	83.6±0.8	83.5±2.0	83.1±1.1

Table A.6: Comparison of IB_SVM and weighted_SVM with MI_fusion on the MLO datasets

Dataset	Kernel	Gmean				AUC			
		Weighted_SVM		IB_SVM		Weighted_SVM		IB_SVM	
		1-Norm	2-Norm	1-Norm	2-Norm	1-Norm	2-Norm	1-Norm	2-Norm
MLO_1	Linear	67.9±1.6	67.2±2.1	68.0±1.4	67.8±2.3	68.4±1.8	68.5±2.4	68.4±2.1	69.6±1.0
MLO_2	Linear	85.2±1.4	85.3±0.8	86.4±0.6	87.0±0.5	84.4±2.1	84.1±1.4	85.9±1.1	85.8±0.7
MLO_3	Linear	87.6±2.1	88.9±0.2	88.5±1.6	88.1±2.3	86.7±1.2	87.0±0.4	86.7±1.4	87.5±1.3
MLO_4	Linear	91.3±0.4	91.9±0.5	91.2±1.1	90.2±2.1	91.4±0.4	92.1±0.5	95.7±0.3	94.9±0.6

References

[1] Steven Abney. Bootstrapping. In *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 360–367, 2002.

[2] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *In Proceedings of the 15th European Conference on Machine Learning (ECML*, pages 39–50, 2004.

[3] Stuart Andrews and Thomas Hofmann. Multiple instance learning via disjunctive programming boosting. In *In Advances in Neural Information Processing Systems (NIPS*16*, pages 65–72, 2004.

[4] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pages 561–568. MIT Press, 2003.

[5] Peter Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 21–29, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[6] Peter Auer and Ronald Ortner. A boosting approach to multiple instance learning. In *Machine Learning: ECML 2004*, pages 63–74. Springer, 2004.

[7] M.R. Azimi-Sadjadi, AA Jamshidi, and G.J. Dobeck. Adaptive underwater target classification with multi-aspect decision feedback. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, volume 2, pages 1554–1559 vol.2, 2001.

[8] Francis R. Bach and Gert R. G. Lanckriet. Multiple kernel learning, conic duality, and the smo algorithm. In *In Proceedings of the 21st International Conference on Machine Learning (ICML*, 2004.

[9] Ricardo Barandela, José Salvador Sánchez, Vicente García, and Francesc J Ferri. Learning from imbalanced sets through resampling and weighting. In *Pattern Recognition and Image Analysis*, pages 80–88. Springer, 2003.

[10] Jeffrey A Barnett. Computational methods for a mathematical theory of evidence. In *Classic Works of the Dempster-Shafer Theory of Belief Functions*, pages 197–216. Springer, 2008.

[11] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1):20–29, 2004.

[12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[13] Luke Bjerring and Eibe Frank. Beyond trees: Adopting miti to learn rules and ensemble classifiers for multi-instance data. In Dianhui Wang and Mark Reynolds, editors, *Australasian Conference on Artificial Intelligence*, volume 7106 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2011.

[14] Hendrik Blockeel, David Page, and Ashwin Srinivasan. Multi-instance tree learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 57–64. ACM, 2005.

[15] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[16] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[17] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

[18] Ulf Brefeld, Christoph Büscher, and Tobias Scheffer. Multi-view discriminative sequential learning. In *Machine Learning: ECML 2005*, pages 60–71. Springer, 2005.

[19] Ulf Brefeld and Tobias Scheffer. Co-em support vector learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 16. ACM, 2004.

[20] Dennis M Buede and Paul Girardi. A target identification comparison of bayesian and dempster-shafer multisensor fusion. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 27(5):569–577, 1997.

[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):27, 2011.

[22] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *arXiv preprint arXiv:1106.1813*, 2002.

[23] Nitesh V Chawla, David A Cieslak, Lawrence O Hall, and Ajay Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2):225–252, 2008.

[24] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[25] Nitesh V. Chawla, Ar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: improving prediction of the minority class in boosting. In *In Proceedings of the Principles of Knowledge Discovery in Databases, PKDD-2003*, pages 107–119, 2003.

[26] Yixin Chen, Jinbo Bi, and James Ze Wang. Miles: Multiple-instance learning via embedded instance selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):1931–1947, 2006.

[27] Yixin Chen and James Z Wang. Image categorization by learning and reasoning with regions. *The Journal of Machine Learning Research*, 5:913–939, 2004.

[28] Zhen-Yu Chen and Zhi-Ping Fan. Parallel multiple kernel learning: a hybrid alternating direction method of multipliers. *Knowledge and Information Systems*, pages 1–24, 2013.

[29] Pak-Ming Cheung and James T Kwok. A regularization framework for multiple-instance learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 193–200. ACM, 2006.

[30] Yann Chevaleyre and Jean-Daniel Zucker. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem. In *Advances in Artificial Intelligence*, pages 204–214. Springer, 2001.

[31] Mario Christoudias, Raquel Urtasun, and Trevor Darrell. Bayesian localized multiple kernel learning. *Univ. California Berkeley, Berkeley, CA*, 2009.

[32] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L 2 regularization for learning kernels. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 109–116. AUAI Press, 2009.

[33] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[34] M Couillard, JA Fawcett, M Davison, and VL Myers. Optimizing time-limited multi-aspect classification. *Proceedings of the Institute of Acoustics*, 29(6):89–96, 2007.

[35] Sanjoy Dasgupta and Michael L Littman. Pac generalization bounds for co-training. 2002.

[36] Wei Di and Melba M Crawford. View generation for multiview maximum disagreement based active learning for hyperspectral image classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(5):1942–1954, 2012.

[37] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[38] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71, 1997.

[39] Lin Dong. A comparison of multi-instance learning algorithms. Technical report, University of Waikato, 2006.

[40] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Citeseer, 2001.

[41] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[42] Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. Adacost: misclassification cost-sensitive boosting. In *ICML*, pages 97–105. Citeseer, 1999.

[43] John Fawcett, Vincent Myers, David Hopkin, Anna Crawford, Michel Couillard, and Benoit Zerr. Multiaspect classification of sidescan sonar images: Four different approaches to fusing single-aspect information. *Oceanic Engineering, IEEE Journal of*, 35(4):863–876, 2010.

[44] Maria florina Balcan, Avrim Blum, and Ke Yang. Co-training and expansion: Towards bridging theory and practice, 2004.

[45] James Foulds and Eibe Frank. Revisiting multiple-instance learning via embedded instance selection. In *AI 2008: Advances in Artificial Intelligence*, pages 300–310. Springer, 2008.

[46] James Foulds and Eibe Frank. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(01):1–25, 2010.

[47] Eibe Frank and Xin Xu. Applying propositional learning algorithms to multi-instance data. 2003.

[48] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[49] Thomas Gärtner, Peter A Flach, Adam Kowalczyk, and Alex J Smola. Multi-instance kernels. In *ICML*, volume 2, pages 179–186, 2002.

[50] Mehmet Gönen and Ethem Alpaydin. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[51] Hongyu Guo and Herna L. Viktor. Multirelational classification: A multiple view approach. *Knowl. Inf. Syst.*, 17(3):287–312, November 2008.

[52] David L Hall and James Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[53] Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[54] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing*, pages 878–887. Springer, 2005.

[55] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[56] Tin Kam Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

[57] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[58] Wayne Iba and Pat Langley. Induction of one-level decision trees. In *Proceedings of the Ninth International Workshop on Machine Learning*, ML '92, pages 233–240, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[59] Nathalie Japkowicz et al. Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. Menlo Park, CA, 2000.

[60] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms*. Cambridge University Press, 2011.

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

[62] Piotr Juszczak and Robert P. W. Duin. Uncertainty sampling methods for one-class classifiers. In *In Proceedings of the ICML03 Workshop on Learning from Imbalanced Data Sets, 2003. Sigkdd Explorations. Volume 6, Issue 1 - Page 5*, 2003.

[63] Pilsung Kang and Sungzoon Cho. Eus svms: Ensemble of under-sampled svms for data imbalance problems. In *Neural Information Processing*, pages 837–846. Springer, 2006.

[64] Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection.

[65] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 393–400, 2011.

[66] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[67] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

[68] Christian Leistner, Amir Saffari, and Horst Bischof. Miforests: Multiple-instance learning with randomized trees. In *Computer Vision–ECCV 2010*, pages 29–42. Springer, 2010.

[69] Darrin P Lewis, Tony Jebara, and William Stafford Noble. Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure. *Bioinformatics*, 22(22):2753–2760, 2006.

[70] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):464–471, 2002.

[71] Chang Liu and Pong C Yuen. A boosted co-training algorithm for human action recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(9):1203–1213, 2011.

[72] Yang Liu, Aijun An, and Xiangji Huang. Boosting prediction accuracy on imbalanced datasets with svm ensembles. In *Advances in Knowledge Discovery and Data Mining*, pages 107–118. Springer, 2006.

[73] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. *Advances in neural information processing systems*, pages 570–576, 1998.

[74] Edson Takashi Matsubara, Maria Carolina Monard, and Gustavo EAPA Batista. Multi-view semi-supervised learning: An approach to obtain different views from text datasets. In *LAPTEC*, pages 97–104, 2005.

[75] Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach: a case study in intensive care monitoring. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1999.

[76] Ion Muslea, Steven Minton, and Craig A Knoblock. Adaptive view validation: A first step towards automatic view detection. In *ICML*, volume 2, pages 443–450. Citeseer, 2002.

[77] Vincent Myers and David P Williams. Adaptive multiview target classification in synthetic aperture sonar images using a partially observable markov decision process. *Oceanic Engineering, IEEE Journal of*, 37(1):45–55, 2012.

[78] Ronaldo C Prati, Gustavo EAPA Batista, and Maria Carolina Monard. Class imbalances versus class overlapping: an analysis of a learning system behavior. In *MICAI 2004: Advances in Artificial Intelligence*, pages 312–321. Springer, 2004.

[79] John Ross Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[80] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, Yves Grandvalet, et al. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[81] Jan Ramon and Luc De Raedt. Multi instance neural networks. 2000.

[82] Bhavani Raskutti and Adam Kowalczyk. Extreme re-balancing for svms: A case study. *SIGKDD Explor. Newsl.*, 6(1):60–69, June 2004.

[83] Scott Reed, Yvan Petillot, and J Bell. Automated approach to classification of mine-like objects in sidescan sonar using highlight and shadow information. In *Radar, Sonar and Navigation, IEE Proceedings-*, volume 151, pages 48–56. IET, 2004.

[84] Scott Reed, Yvan Petillot, and Judith Bell. Model-based approach to the detection and classification of mines in sidescan sonar. *Applied optics*, 43(2):237–246, 2004.

[85] Giancarlo Ruffo. Learning single and multiple instance decision trees for computer security applications. *Master's Thesis, Department of Computer Science, University of Turin, Turin, Italy*, 2000.

[86] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[87] Stephen Scott, Jun Zhang, and Joshua Brown. On generalized multiple-instance learning. *International Journal of Computational Intelligence and Applications*, 5(01):21–35, 2005.

[88] Glenn Shafer et al. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.

[89] John Shawe-Taylor and Nello Cristianini. Further results on the margin distribution. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 278–285. ACM, 1999.

[90] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *The Journal of Machine Learning Research*, 7:1531–1565, 2006.

[91] Chao-Ton Su and Yu-Hsiang Hsiao. An evaluation of the robustness of mts for imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 19(10):1321–1332, 2007.

[92] Jiang Su and Harry Zhang. A fast decision tree learning algorithm. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 500–505. AAAI Press, 2006.

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

[94] Qingping Tao, Stephen Scott, NV Vinodchandran, Thomas Takeo Osugi, and Brandon Mueller. An extended kernel for generalized multiple-instance learning. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 272–277. IEEE, 2004.

[95] Qingping Tao and Stephen D Scott. A faster algorithm for generalized multiple-instance learning. In *FLAIRS Conference*, pages 550–555, 2004.

[96] Kai Ming Ting. A comparative study of cost-sensitive boosting algorithms. In *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer, 2000.

[97] Kai Ming Ting. An instance-weighting method to induce cost-sensitive trees. *Knowledge and Data Engineering, IEEE Transactions on*, 14(3):659–665, 2002.

[98] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, pages 935–942. ACM, 2007.

[99] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, et al. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on artificial intelligence*, volume 1999, pages 55–60, 1999.

[100] Benjamin X Wang and Nathalie Japkowicz. Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, 25(1):1–20, 2010.

[101] Jun Wang. Solving the multiple-instance problem: A lazy learning approach. In *In Proc. 17th International Conf. on Machine Learning*, pages 1119–1125. Morgan Kaufmann, 2000.

[102] Wei Wang and Zhi-Hua Zhou. Analyzing co-training style algorithms. In *Machine Learning: ECML 2007*, pages 454–465. Springer, 2007.

[103] Xiaoguang Wang, Xuan Liu, N. Japkowicz, and S. Matwin. Resampling and cost-sensitive methods for imbalanced multi-instance learning. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 808–816, Dec 2013.

[104] Xiaoguang Wang, Xuan Liu, Nathalie Japkowicz, and Stan Matwin. Ensemble of multiple kernel SVM classifiers. In *Advances in Artificial Intelligence - 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings*, pages 239–250, 2014.

[105] Xiaoguang Wang, Xuan Liu, Nathalie Japkowicz, Stan Matwin, and Bao Nguyen. Automatic target recognition using multiple-aspect sonar images. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2330–2337. IEEE, 2014.

[106] Xiaoguang Wang, Xuan Liu, Stan Matwin, and Nathalie Japkowicz. Applying instance-weighted support vector machines to class imbalanced datasets. In *Proceedings of the 2014 IEEE International Conference on Big Data, 27-30 October 2014,Washington DC, USA*, 2014.

[107] Xiaoguang Wang, Xuan Liu, Stan Matwin, Nathalie Japkowicz, and Hongyu Guo. A multi-view two-level classification method for generalized multi-instance problems. In *Proceedings of the 2014 IEEE International Conference on Big Data, 27-30 October 2014,Washington DC, USA*, 2014.

[108] Xiaoguang Wang, Hang Shao, Nathalie Japkowicz, Stan Matwin, Xuan Liu, Alex Bourque, and Bao Nguyen. Using svm with adaptively asymmetric misclassification costs for mine-like objects detection. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 78–82. IEEE, 2012.

[109] Zhe Wang, Songcan Chen, and Daqi Gao. A novel multi-view learning developed from single-view patterns. *Pattern Recognition*, 44(10):2395–2413, 2011.

[110] Nils Weidmann, Eibe Frank, and Bernhard Pfahringer. A two-level learning method for generalized multi-instance problems. In *Machine Learning: ECML 2003*, pages 468–479. Springer, 2003.

[111] David P Williams, Vincent Myers, and Miranda Schatten Silvious. Mine classification with imbalanced data. *Geoscience and Remote Sensing Letters, IEEE*, 6(3):528–532, 2009.

[112] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[113] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[114] Gang Wu and Edward Y Chang. Adaptive feature-space conformal transformation for imbalanced-data learning. In *ICML*, pages 816–823, 2003.

[115] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[116] Xin Xu. Statistical learning in multiple instance problems. Technical report, 2003.

[117] Xin Xu and Eibe Frank. Logistic regression and boosting for labeled bags of instances. In *Advances in Knowledge Discovery and Data Mining*, pages 272–281. Springer, 2004.

[118] Jun Yang. Review of multi-instance learning and its applications. Technical report, Tech. Rep, 2005.

[119] Xulei Yang, Qing Song, and Yue Wang. A weighted support vector machine for data classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(05):961–976, 2007.

[120] Shipeng Yu, Balaji Krishnapuram, Rómer Rosales, and R Bharat Rao. Bayesian co-training. *The Journal of Machine Learning Research*, 12:2649–2680, 2011.

[121] B Zerr, B Stage, and A Guerrero. Automatic target classification using multiple sidescan sonar images of different orientations. *SACLANT Undersea Research Center Memorandum SM-309 of Journal*, 1997.

[122] Min-Ling Zhang and Zhi-Hua Zhou. Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, 31(1):47–68, 2009.

[123] Qi Zhang and Sally A Goldman. Em-dd: An improved multiple-instance learning technique. In *Advances in neural information processing systems*, pages 1073–1080, 2001.

[124] Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. Multi-instance learning by treating instances as non-iid samples. In *Proceedings of the 26th annual international conference on machine learning*, pages 1249–1256. ACM, 2009.

[125] Zhi-Hua Zhou and Jun-Ming Xu. On the relation between multi-instance learning and semi-supervised learning. In *Proceedings of the 24th international conference on Machine learning*, pages 1167–1174. ACM, 2007.

[126] Zhi-Hua Zhou and Min-Ling Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowledge and Information Systems*, 11(2):155–170, 2007.