MATHEMATICAL PROGRAMMING AND STATISTICAL LEARNING APPROACHES FOR MULTIPLE INSTANCE LEARNING

by

Emel Şeyma Küçükaşcı

B.S., Industrial Engineering, Istanbul Commerce University, 2010

M.S., Industrial Engineering, Boğaziçi University, 2013

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2018

ACKNOWLEDGEMENTS

I would first like to thank my supervisor Assist. Prof. Mustafa Gökçe Baydoğan for his invaluable guidance, endless support and mentorship throughout this study. He has truly inspired me during my research and I have learned lots of things from him. I would like express my deepest gratitude to my co-supervisor Prof. Z. Caner Taşkın, for being a tremendous mentor for me helping to shape my own identity and path, and for his critical interventions that have prevented me from getting lost during this work.

I am very grateful to my thesis monitoring committee members, Prof. Necati Aras and Assoc. Prof. Erhun Kundakçıoğlu for their guidance and contributions throughout the study. I would like to give special thanks to Assoc. Prof. Semra Ağralı and Prof. Taylan Cemgil for their most valuable and constructive comments in my thesis defense jury.

I feel myself as a part of a big family during research assistantship years in the Industrial Engineering Department of Istanbul Commerce University. I am grateful to all my professors, my colleagues, my students and the staff of the department for their support and smiling faces. I owe this University a deep debt of gratitude. I am also very grateful to have been a Ph.D. student at the Department of Industrial Engineering at Boğaziçi University. I would like to thank the faculty members, research assistants and all the people I have met during my thesis studies.

Finally and most importantly, special thanks go to my parents Mine Küçükaşcı and Mustafa Sabri Küçükaşcı, my sisters Ebrar Küçükaşcı and Nurbanu Küçükaşcı, and my brother Ziya Küçükaşcı, who offer unconditional support and always be patient over the years that it has taken me to get to this point. This thesis is dedicated to them. I would like to thank my aunt Beyza Özyeşil, my grandparents Sevim Küçükaşcı and A. Ziya Küçükaşcı, my deceased grandfather, Mukadder Meram, and my grandmother Fatma Meram, who always encouraged me and shared my excitement.

ABSTRACT

MATHEMATICAL PROGRAMMING AND STATISTICAL LEARNING APPROACHES FOR MULTIPLE INSTANCE LEARNING

Many real-world applications of classification require flexibility in representing complex objects to preserve the relevant information for class separation. Multiple instance learning (MIL) aims to solve classification problem where each object is represented with a bag of instances, and class labels are provided for the bags rather than individual instances. The aim is to learn a function that correctly labels new bags. In this thesis, we propose statistical learning and mathematical optimization methods to solve MIL problems from diversified application domains. We first present bag encoding strategies to obtain bag-level feature vectors for MIL. Simple instance space partitioning approaches are utilized to learn representative feature vectors for the bags. Our experiments on a large database of MIL problems show that random tree-based encoding is scalable and its performance is competitive with the state-of-the-art methods. Mathematical programming-based approaches to MIL problem construct a bag-level decision function. In this context, we formulate MIL problem as a linear programming model to optimize bag orderings for correct classification. Proposed formulation combines instance-level scores to return an estimate on the bag label. All instances are solved to optimality on various data representations in a reasonable computation time. At last, we develop a quadratic programming formulation that is superior to previous MIL formulations on underlying assumptions and computational difficulties. Proposed MIL framework models contributions of instances to the bag class labels, and provide a bag class decision threshold. Experimental results verify that proposed formulation enables effective classification in various MIL applications.

ÖZET

ÇOKLU ÖRNEKLE ÖĞRENME İÇİN MATEMATİKSEL PROGRAMLAMA VE İSTATİSTİKSEL ÖĞRENME YAKLAŞIMLARI

Gerçek hayattaki pek çok sınıflandırma uygulaması sınıf ayrımı için gerekli olan bilginin saklanması için karmaşık yapılı nesnelerin temsilinde bir esneklik gerektirir. Çoklu Örnekle Öğrenme (ÇÖÖ), sınıflandırma problemini her nesnenin bir örnek torbası ile temsil edildiği durumda çözmeyi amaçlar ve sınıf etiketleri bireysel örnekler için değil, sadece torbalar için sağlanmaktadır. Amaç, yeni torbaları doğru şekilde etiketleyen bir sınıflandırıcının öğrenilmesidir. Bu tezde, farklı uygulama alanlarına ait ÇÖÖ problemlerini çözmek için istatistiksel öğrenme ve matematiksel eniyileme yöntemleri öneriyoruz. İlk olarak, ÇÖÖ için torba seviyesinde öznitelik vektörleri üreten torba kodlama izlemleri sunuyoruz. Torbaların temsili öznitelik vektörlerini öğrenmek için örnek uzayını bölümleyen basit yaklaşımlar kullandık. Geniş bir ÇÖÖ veritabanı üzerinde yaptığımız deneyler rastgele ağaç tabanlı kodlamanın ölçeklenebilir olduğunu ve sınıflandırma başarımı açısından bilinen yöntemlerle rekabet edebildiğini göstermektedir. Matematiksel programlama temelli ÇÖÖ yaklaşımları torba seviyesinde bir karar fonksiyonu oluşturur. Bu bağlamda, ÇÖÖ probleminin doğru sınıflandırma için torba sıralamasını eniyileyen bir doğrusal programlama gösterimini sunuyoruz. Önerilen gösterim örnek seviyesindeki tahminleri birleştirerek torba etiketini tahmin eder. Çeşitli veri gösterimleri üzerindeki problem örnekleri makul bir hesaplama zamanında eniyiye çözülmektedir. Son olarak, dikkate alınan varsayımlar ve hesaplama zorluğu açısından önceki ÇÖÖ gösterimlerinden üstün olan bir karesel programlama gösterimi geliştiriyoruz. Önerilen ÇÖÖ yöntemi torba sınıfı etiketlerine örneklerin katkısını modeller ve bir torba sınıfı karar eşiği sağlar. Deneysel sonuçlar önerilen gösterimin farklı ÇÖÖ uygulamalarındaki sınıflandırma etkinliğini doğrulamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\odot$	Hadamard product
$\|\|\cdot\|\|$	ℓ_1-norm
$\|\|\cdot\|\|^2$	ℓ_2-norm
$\langle\cdot,\cdot\rangle$	Dot product
$1(\cdot)$	1 if the parameter is true, 0 otherwise
$\top$	Vector transpose
b	Bias term
$\mathbf{b}$	Bag vector
B	Bag of instances
$\mathbf{B}'$	Set of bag vectors
$\mathbf{c}$	Cluster centers
C	Trade-off parameter
$\mathcal{C}$	Set of classes
d	Dimensionality of the original feature space
$d(\cdot,\cdot)$	Squared Euclidean distance
D^{RS}	Random subspaces formed by dissimilarities to prototypes
$f(\cdot)$	Instance-level decision function
$g(\cdot)$	Bag-level decision function
$G(\mathbf{x})$	Set of all trees in the random forest, where instance $\mathbf{x}$ is out-of-bag
h	Depth of a tree
$H(\cdot,\cdot)$	Class separating hyperplane
$\mathcal{H}$	Embedding space
I	Index set of instances
I^+	Index set of instances in positive bags
I^-	Index set of instances in negative bags
J^+	Index set of positive bags
J^-	Index set of negative bags
k	Dimensionality of the mapped feature space

K	Number of generalized Benders cuts
$K(\cdot,\cdot)$	Kernel function
l	Bag class label
$L(\boldsymbol{\alpha})$	Lagrangean dual value for solution $\boldsymbol{\alpha}$
m	Number of training bags
$\mathbf{m}$	Instance pseudo class memberships
m^+	Number of positive bags
m^-	Number of negative bags
M	Sufficiently large positive number
n	Number of training instances
n_j	Number of instances in j-th bag
$p_t^c(\mathbf{x})$	Estimated proportion of class c in the corresponding leaf of the t-th tree
$\Re$	Real numbers
$\mathrm{R}^{\mathrm{cluster}}$	Cluster center-based data representation
$\mathrm{R}^{\mathrm{instance}}$	Instance dissimilarity-based data representation
r_t	t-th decision tree in a random forest
T	Number of decision trees in the ensemble
$\mathbf{u}$	Normalized weights of instances
v	Auxiliary variable in generalized Benders cut
$\mathbf{v}$	Representative vector of instance feature space
$\mathbf{w}$	Feature weights
W	Wilcoxon-Mann-Whitney statistic
$\mathbf{x}$	Data instance
X	Set of training instances
y	Instance class label
$\hat{y}_t$	Prediction of a t-th decision tree
$\mathbf{z}$	Auxiliary variables replacing $\xi \odot \eta$
$\boldsymbol{\alpha}$	Variable of the dual SVM problem
$\boldsymbol{\beta}$	Bag class memberships
γ	Width parameter of the Gaussian kernel

δ_d	Randomly selected proportion of features
δ_m	Randomly selected proportion of bags
$\boldsymbol{\delta}^+$	Slack variables for the positive bag deviations
$\boldsymbol{\delta}^-$	Slack variables for the negative bag deviations
ε	Threshold controlling parameter
$\boldsymbol{\eta}$	Variables identifying witness instances
$\boldsymbol{\theta}$	Lagrange multiplier corresponding to constraint (5.4b)
κ	Number of clusters
$\boldsymbol{\lambda}$	Lagrange multiplier corresponding to constraint (5.4h)
$\boldsymbol{\mu}$	Lagrange multiplier corresponding to constraint (5.4e)
$\boldsymbol{\nu}$	Lagrange multiplier corresponding to constraint (5.4f)
$\boldsymbol{\xi}$	Slack variable for instance misclassifications
$\boldsymbol{\xi}^+$	Slack variable for positive bag misclassifications
$\boldsymbol{\xi}^-$	Slack variable for negative bag misclassifications
$\mathcal{O}$	Computational complexity ("of the order of")
$\boldsymbol{\pi}$	Lagrange multiplier corresponding to constraint (5.4g)
$\boldsymbol{\rho}$	Lagrange multiplier corresponding to constraint (5.4c)
$\boldsymbol{\sigma}$	Bag class membership differences
τ	Decision threshold for bag classification
$\phi(\cdot)$	Mapping function
$\Phi(\boldsymbol{\eta})$	Lagrangean dual value for solution $\boldsymbol{\eta}$
χ	Set of training bags

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
AUC	Area under Receiver Operating Characteristic Curve
BoF	Bag-of-Features
BoW	Bag-of-Words
CART	Classification and Regression Trees
CCCP	Concave Convex Procedure
CCE	Constructive Clustering Ensemble
CD	Critical Difference
CPU	Central Processing Unit
CV	Cross Validation
DD	Diverse Density
EM-DD	Expectation Maximization Diverse Density
FV	Fisher Vector
GBD	Generalized Benders Decomposition
GD-MIL	Generalized Dictionaries for Multiple Instance Learning
GMM	Gaussian Mixture Model
GPMIL	Gaussian Process Multiple Instance Learning
kNN	K-Nearest Neighbors
LP	Linear Programming
LP-MIL	Linear Programming-based Multiple Instance Learning
MIDL	Multi-Instance Dictionary Learning
miFV	Multi-Instance Learning based on Fisher Vector Representation
miGraph	Multiple Instance Graph
MIGraph	Multiple Instance Graph (version 2)
MIHLSVM	Multiple Instance Hinge Loss Support Vector Machine
MIHMSVM	Multiple Instance Hard Margin Support Vector Machine
MILD	Multiple Instance Learning via Disambiguation
MILDS	Multiple Instance Learning via Dominant Sets

MILES	Multiple Instance Learning via Embedded Instance Selection
MIL-IBRT	Multi-Instance Learning via Instance-based and Bag-based Representation Transformations
MILIS	Multiple Instance Learning with Instance Selection
MILSD	Multiple Instance Learning on Structured Data
MInD	Multiple Instance Dissimilarity
MINLP	Mixed Integer Nonlinear Programming
MIQP	Mixed Integer Quadratic Programming
MIRLSVM	Multiple Instance Ramp Loss Support Vector Machine
MissSVM	Multi-Instance Learning by Semi-Supervised Support Vector Machine
MP	Master Problem
NC-MINLP	Non-Convex Mixed Integer Nonlinear Programming
NLP	Nonlinear Programming
MISVM	Multiple Instance Support Vector Machine
mi-SVM	Mixed-Integer Support Vector Machine
MI-SVM	Multi-Instance Support Vector Machine
MITI	Multi-Instance Tree Induction
MMDL	Max-Margin Multiple-Instance Dictionary Learning
OOB	Out-of-Bag
PCA	Principal Component Analysis
QP	Quadratic Programming
QP-MIL	Quadratic Programming-based Multiple Instance Learning
RF	Random Forest
RL	Representation Learning
ROC	Receiver Operating Characteristic
RS	Random Subspace
RSIS	Random Subspace Instance Selection
RT	Random Tree
sbMIL	Sparse Balanced Multiple Instance Learning
SCCE	Sparse Coding and Classifier Ensemble
SP	Subproblem

stMIL	Sparse Transductive Multiple Instance Learning
SVM	Support Vector Machine
TLC	Two Level Classification
VOC	Visual Object Classes
WMW	Wilcoxon-Mann-Whitney

1. INTRODUCTION

The interplay between learning systems and scientific discovery processes has received growing attention respecting the availability of large scaled and complex structured data. *Learning* is the process of efficiently and repeatability representing a system by properly taking the collected information as an input in a special format. *Machine learning* develops algorithms for numerous learning purposes such as understanding and improving the learning, discovering new patterns and completing the missing parts of the learning processes. The intersection of computational sciences and machine learning is referred to as *statistical machine learning*. Statistical machine learning is divided into two main categories: *supervised learning* and *unsupervised learning*. Primary goal of *supervised learning* is to model a functional relationship between the input sample and their already given outputs, which are collectively named as *data*. The whole data used during the development of the model forms the *training set*. The known outputs, which are named as the *labels*, can take discrete values. In order to inspect the success of the model, another set of inputs with known labels is evaluated by the model and its resulting outputs are compared with the actual labels. The described set is the *test set*, being a portion of the whole input, which is the *dataset*. Then, the last step is to detect the label of an unlabeled object. Each input vector belongs to an object, which is also entitled as an *instance*. Each instance includes values of different measurements referred to as *features*.

Classification is a supervised learning task of identifying the categories of instances by means of estimating their labels with discrete values, named as the *classes*. If only two classes exist, *binary classification* takes place. Besides, *multi-class classification* classifies the objects into more than two classes. *Unsupervised learning* copes with identification and improvement of dataset structure with neither benefiting from the labels, nor splitting the whole dataset into training set and test set. Its main difference from the supervised learning is that there is no model building and no parameter estimation. One of the most famous unsupervised learning approaches is *clustering* where the input sample is categorized based on its inherent similarities.

Applying machine learning algorithms to big datasets for discovery of existing patterns can be defined as *data mining*. Classification often takes place in data mining tasks. However, it is sometimes costly to obtain labeled data. This fact attracted researchers to benefit from the potentially supportive unlabeled data. In this sense, *semi-supervised learning* (SSL) is taken into consideration to enable extraction of useful information from unlabeled data for classification. Since it is expensive and time consuming to acquire labeled data and unlabeled data is naturally much more frequent, SSL covers a wide range of applications such as bioinformatics [1], text classification [2] and image retrieval [3]. In short, proposed SSL methods either alter the supervised classifiers and the data representation to include the information obtained from unlabelled data, or iteratively labels the unlabelled instances.

Multiple instance learning (MIL) is a variation of supervised learning, where instead of the instances, there are *bags* and each bag has certain number of instances as exemplified in Figure 1.1. Bags are labeled whereas the individual instances inside the bags do not have to be labeled. The lack of instance label information converts the problem into a SSL problem where supervised learning procedures can be facilitated with some unlabeled data [4]. Most of the MIL approaches generally solve the binary classification problem, where there are only two classes such that the positive class and the negative class. Figure 1.2 illustrates the difference between traditional supervised learning and MIL in a binary classification problem.

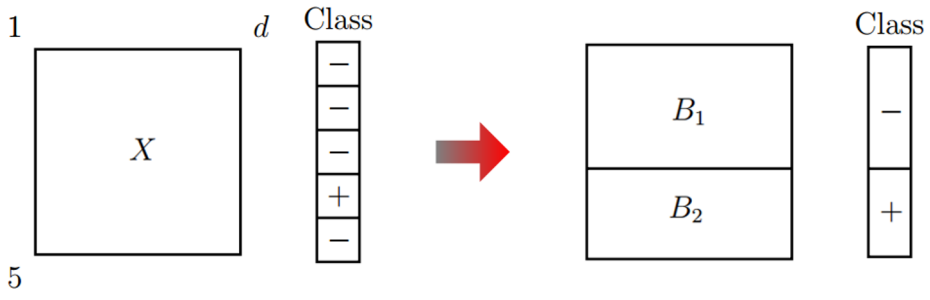


Figure 1.1. Representation of 2 bags formed by 5 instances with d many features.

In the original MIL problem, musky and non-musky molecules form a collection of bags [5]. For each molecule, instances describe molecule shapes and some of which decide the smell of the molecule. A molecule with at least one effective shape induces

a musky smell and is classified as positive. Otherwise, the molecule is classified as negative. The described formal definition of first MIL application brings about the most common MIL assumption on relating the instance labels to the bag label. Correspondingly, *standard MIL assumption* states that a bag is labeled positive if it contains at least one positive instance, and otherwise labeled as negative.

Later on, researchers follow common hypothesis of standard MIL and adapt this setting to several machine learning applications. In the context of text classification, authors of [6] treat text documents as a bag of passages and some of the passages about relevant topics assigns a positive label to a document. In adaptation of MIL framework for image classification [7–9], each image is represented by a bag of small image segments (i.e. patches). An example on MIL for medical imaging is classification of histopathology images to recognize cancer [9]. As an illustration, images belonging to either malignant or benign cells are the bags and square patches of them are the instances as shown in the top figure of Figure 1.4. Images of malignant cells form the positive class.

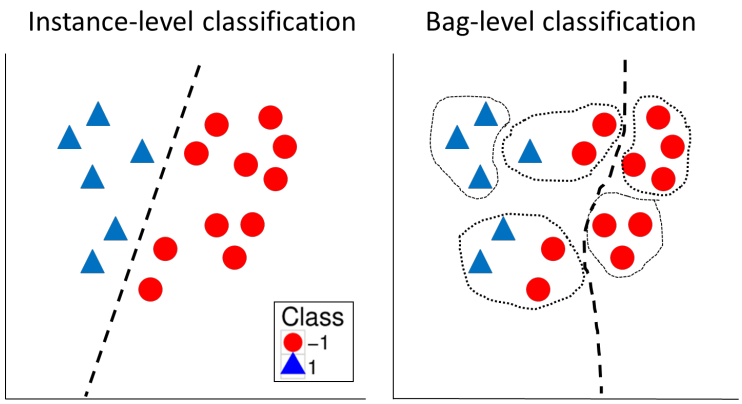


Figure 1.2. Difference between regular supervised classification and multiple instance classification. In the instance-feature space, blue color indicates positive instances and red color indicates negative instances.

A more recent MIL application is classification of birdsong recordings [10]. A spectrogram of each recording is a bag and segments of spectrogram are the instances. Other MIL applications are protein identification [11], hard drive failure prediction [12], stock selection [13], music information retrieval [14], prediction of student performance

[15] and more recently medical video analysis [16], video event detection [17] and co-saliency detection [18], etc.

A major challenge in MIL problems is ambiguity regarding instance labels. Specifically, there is uncertainty on determination of the instances that are responsible for the bag labels. In aforementioned MIL applications, relationships between the instances inside bags need to be modeled to summarize the bag-level information. Standard MIL assumption is the simplest way of representing bag label information. Definition of standard MIL is compatible with molecular activity prediction since existence of an effective shape is a strong evidence of bag positivity. However, incompatibility with the standard assumption may occur in many other learning applications on complex objects.

Consider an MIL problem of detecting person-object interactions in images such as a person riding a horse. A positively classified image must contain not only a person or a horse, but both. Besides, instances from person concept and horse concept must reside in neighboring patches, and must be subjected to position measurements [19]. Therefore, instances from different data regions are required to observe a positively labeled image. As a second example, consider event detection in video recordings. Each video is a bag and the defined video segments of temporal intervals are the instances. Furthermore, the event to be detected is attempting a board trick. To classify a bag as positive, a certain portion of instances representing jumping is required, and negative bags can also contain these instances such as a parkour event video [17]. Consequently, standard MIL assumption is not suitable for this problem, either.

To overcome the limitations of standard MIL assumption, researchers advocate new settings about modeling instance-level interactions to predict bag labels, which are categorized in the context of generalized MIL [20–22]. Several assumptions with increasing complexity are proposed to model MIL problems when there exists a higher dependency of the bag label to the instance-level information. Considerably positive instances in positive bags belong to a specific data region and jointly form the *concept*.

In standard MIL, representative instances in positive bags are similar to each other, and at least one instance from the concept region is enough to decide the positive label. A hierarchy of MIL assumptions is proposed in [20] to raise different MIL problems to generalize the standard MIL assumption. The assumptions in Weidmann's hierarchy [20] are sorted by an increasing generality as standard MIL, presence-based MIL, threshold-based MIL and count-based MIL. Figure 1.3 shows example positive and negative bags following single instance learning and MIL under different assumptions.

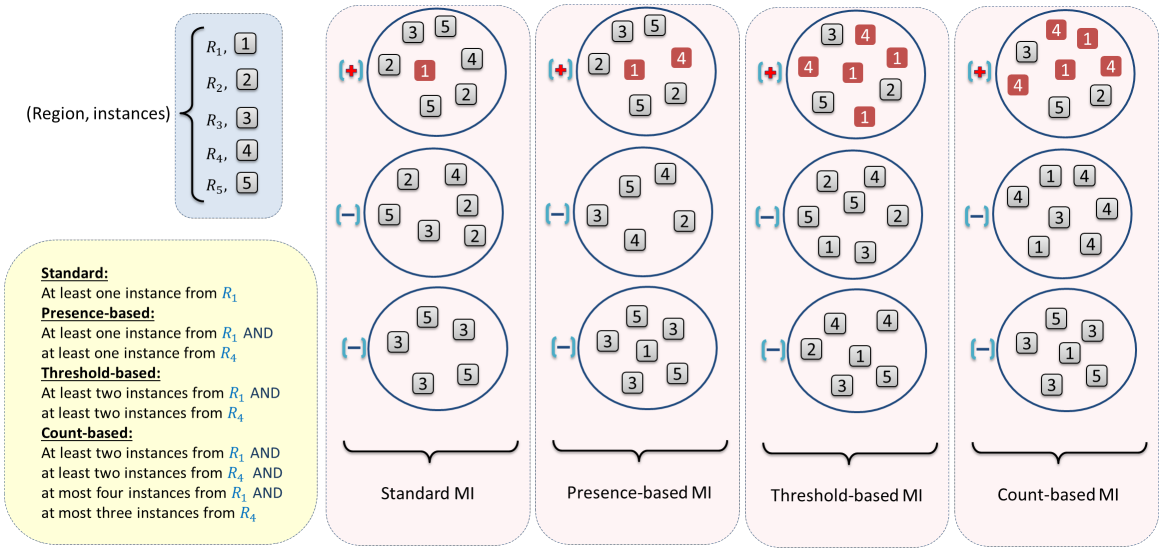


Figure 1.3. An illustration of MIL assumptions.

Different from the previous complex definitions of positive bags, more general hypotheses on bag label determination are required in some MIL applications. Alternative ways of defining positive bags considering the bags as a whole are introduced in [21]. Under *collective* MIL assumption [21], the proportion of instances or all of the instances in a bag decide the bag label collectively. Namely, the learning approach combines instance-level decisions to come up with an estimation of bag labels. Figure 1.4 shows learning process following the collective MIL assumption. For each histopathology image, a bag-level classification score is computed for cancer disease diagnosis. In particular, a large value of bag score is associated with a potentially malignant cellular image. During learning, a linear function determines an instance-level score for each square patch of a training image. After model building, output classifier averages the instance scores to assess bag-level class predictions for test bags.

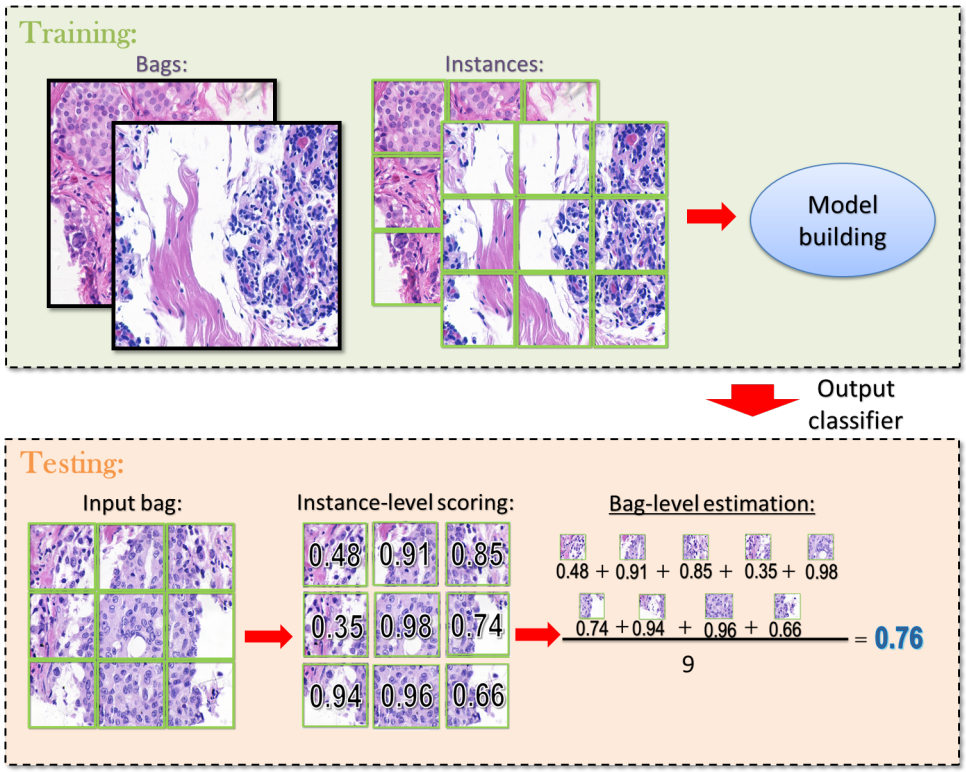


Figure 1.4. Generic description of MIL classification under collective MIL assumption.

There is a vast literature on MIL methods following the standard MIL assumption, which are categorized as standard MIL methods. These methods are reviewed in comprehensive surveys [21–23], and may not be suitable for certain types of MIL problems arising in more complex real-world applications [24]. Therefore, generalized MIL methods constitute the second main category of the proposed methods in the literature. Figure 1.5 displays such a taxonomy of the state-of-the-art methods for MIL. Generalized MIL methods do not require a specific MIL assumption and mostly favour bag-level learning. Use of bag representations as an input to classical supervised learning algorithms are shown to work well in MIL problems [8, 25–29].

The early standard MIL methods are dominated by generative approaches, which depend on describing probability distributions of the instances in bags. The first MIL algorithm [5] defines an axis parallel rectangle (APR) along with minimization of the number of negative instances and maximization of positive instances inside APR. Diverse Density (DD) algorithm [13] assumes Gaussian distribution of positive instances and extended by EM-DD [30]. DD is maximized in these methods by considering the

instances in positive bags close to each other in the instance-feature space. MIL with Gaussian processes (GP) latent variables GPMIL [31] generates a bag class likelihood model.

A number of standard MIL methods primarily utilize discriminative approaches via adaptation of supervised learning models. mi-SVM method in [7] estimates the unknown instance labels, while a support vector machine (SVM) classifier is simultaneously being updated. MI-SVM is a second method in [7], where parameters of an instance-level classifier are learned by SVMs. A sparse-MIL approach, sMIL is proposed in [32] by mitigating the sparse structure of the positive bags, while adapting the SVM formulation to the MI setting.

Mathematical programming formulations of sparse transductive MIL (stMIL) [32] and multi-instance learning via semi-supervised support vector machine (MissSVM) [4] are equivalent to MI-SVM [7], except for the additional constraints to improve class separation and to prevent misclassification of witness instances. A witness selection mechanism is employed in KI-SVM [33] by learning a convex combination of positive instances, referred to as key instances. Another SVM-MIL formulation [34] outputs a nonlinear kernel classifier in their proposed multiple instance classification algorithm (MICA).

Standard MIL assumption is encoded in margin maximization based MIL formulation proposed in [35]. A single-instance learner is used to classify bags without integrating bag information. Later on, authors of [36] present robust SVM-based models highlighting a generalizable classification performance by utilization of different loss functions and maximization of the bag margin. MIHMSVM [36] enforces a bag-level hard margin, whereas MIHLSVM [36] enables bag misclassifications by the usage of hinge loss.

A recent survey [37] compares and discusses extensions of SVM-based models for MIL. Scalability to large-scaled datasets and generalizability to the assumptions other than the standard MIL assumption are the main concerns on the mathematical models,

which are mostly oriented on standard MIL assumption.

Most of the generalized MIL methods transform MIL problems into single instance learning problems. First group of algorithms in this category either employs extra constraints as in multiple instance online (MIO) algorithm [38] or utilizes kernel functions as in MI-Kernel [39] and regularized MI-kernels [40].

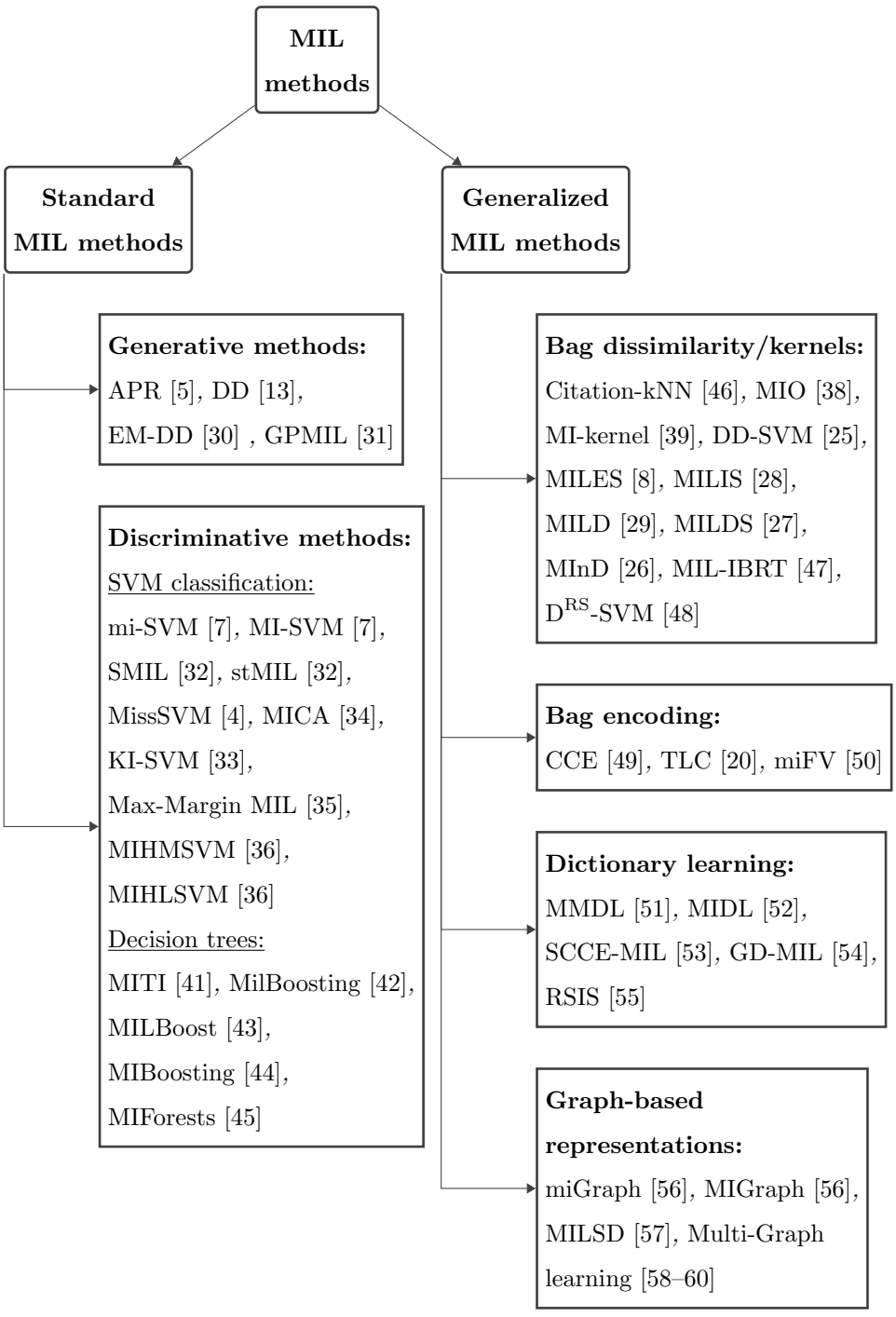


Figure 1.5. A taxonomy of MIL methods.

A second category of generalized MIL methods introduce different ways of selecting representative instances and represents the bags in a new feature space, which includes DD-SVM [25], MILES [8], MILD [29], MILIS [28] and MILDS [27]. These methods commonly train SVM classifiers in the bag space. RSIS [55] selects instance prototypes by finding random subspaces that are obtained using clustering. An ensemble of SVM classifiers are trained on subsets of the instances. Instead of clustering, random subspaces are constructed by dissimilarities to prototypes in D^{RS}-SVM [48]. Citation-kNN [46] measures the minimum distance between bags with a Hausdorff distance. MInD [26] constitute bag-level representations by defining alternative bag-to-bag dissimilarity measures. Unlike MInD, MIL-IBRT [47] assigns different weights to each instance of a bag during measurements.

Bag encoding-based MIL forms the third group of generalized MIL methods. TLC [20] is a two step classification approach with encoding instance-level information using a decision tree in the first step and recovers bag class label in the second step. Constructive clustering ensemble (CCE) [49] encodes each bag into a binary feature vector after clustering the original instances. Another bag-level approach perform MIL using Fisher vector encoding of bags (miFV) [50]. A Gaussian Mixture Model (GMM) is learned in the instance space to estimate parameters of the density function to be utilized to derive Fisher vector representation of bags. Recently, dictionary learning methods such as MMDL [51], MIDL [52] and GD-MIL [61], and a sparse coding method, SCCE [53] are proposed for MIL.

Graph-based representations are also utilized in several MIL methods [56–60]. Briefly, instances correspond to the nodes and edges between the node pairs are defined based on instance relationships. miGraph [56] and MIGraph [56] employ different graph kernels and MILSD [57] uses additional structural information on bags and instances. Alternatively, a number of recent MIL methods represent a bag with multiple subgraphs [58–60].

The main concern of MIL is to assign labels to bags rather than instances. More importantly, interactions of the instances decide the bag label and this issue motivates

generalized MIL. Generally, it is not obvious which MIL assumption holds for a real-life MIL problem. The previous efforts to model and solve learning problems under MIL framework mostly focus on resolving the uncertainty on instance labels, rather than regarding internal bag structure information and modeling the instance-level contributions to the bag labels.

A conventional way of bag-level classification is to encode the instance-level information and obtain vectorial representation of the bags. The idea is to extract the necessary information for bag encoding and thereby perform bag classification. After each bag is represented with a feature vector, anyone can train a supervised learner on the new representation. In this thesis, we first consider strategies to represent bags without imposing any parameter estimation, and observe high classification accuracy with requirements of less computational effort. Note that performance of the classifier highly depends on the way of bag representation and selection of machine learning algorithms to be utilized, which are heuristic approaches.

Correct classification of bags is a statistical learning problem and is closely related with resolution of optimization problems. For instance, existing SVM classification-based models for MIL require numerical optimization. Furthermore, solving optimization problems promotes interpretability and reproducibility of the classification scheme [62]. We briefly review existing optimization algorithms for MIL in terms of model building and solution process. To overcome limitations of the previous approaches regarding computational requirements and adaptability to real-world scenarios, we come up with promising alternative optimization frameworks for MIL. Solutions to proposed formulations provide a linear mapping of instances to model their contributions to the bag labels. This way, implementation of novel mathematical formulations present a simpler and more tractable process of learning under MIL paradigm.

In this thesis, we propose learning frameworks specialized to solve real world learning problems that can be generalized to classify bags of multiple instances. The main contributions of this thesis can be summarized as follows:

- We propose a robust framework for solving multiple instance learning (MIL) problems that uses random trees to partition the feature space together with a path-based representation. Simple instance feature space partitioning approaches are utilized to learn representative and MIL assumption-free feature vectors to encode the bags (see Chapter 3).
- A mathematical programming-based method is presented for solving MIL problems that models instance level contributions to assess bag labels. Linear programming models are solved to obtain a simple linear mapping of instance-level information, which provides bag-level estimates and handles MIL problems from various application domains (see Chapter 4).
- We present an efficient quadratic programming (QP) model and optimization algorithm for MIL. Our QP-based approach to MIL aggregates instance-level estimations to obtain a bag label estimation score along with a bag-level class decision threshold. The proposed MIL framework is efficient in terms of solution time, overcoming the computational difficulties in previous MIL formulations (see Chapter 5).
- The proposed MIL approaches are robust to changes in parameter settings and do not rely on any specific assumptions on bag formations and class labels. After either bag encoding or optimization algorithms, resulting decision functions provide solutions to bag classification in diverse application areas. Experimental results demonstrate classification success of the proposed methods compared to the previous state-of-the-art MIL approaches on a wide range of real world datasets.
- To promote reproducible research, implementations of proposed algorithms, benchmark datasets, computational results and the information regarding the experimental settings are made available on our supporting website [63]. To the best of our knowledge, our experimental setup considers the largest database in evaluation of the approaches.

The rest of this dissertation is structured in the following way. Chapter 2 formalizes the MIL problem and provides the necessary background. In Chapter 3, we

describe alternative strategies for bag-level representations to solve MIL problems. A comprehensive performance comparison is also provided including experiments on large-scaled benchmarks and detailed sensitivity analysis on both proposed and state-of-the-art methods. Chapter 4 presents what is, to the best of our knowledge, the first linear programming-based approach for MIL. We describe the optimization model embedding notion of bag ranking and introduce various data representation alternatives. In Chapter 5, we propose a novel quadratic programming-based approach to classify bags. This chapter also includes performance comparisons with an existing SVM-based MIL method and leading machine learning algorithms for MIL. Chapter 6 draws the conclusions by summarizing the contributions of thesis, and presents possible research directions.

2. BACKGROUND

This chapter presents formal description of the MIL problem and the details about basic tools of machine learning that are relevant to our study. In Chapter 3, random tree-ensembles and k-means clustering are utilized to transform multiple instance representation to a single bag vector. This way, we simply employ a supervised classification algorithm, random forests on the bag-level representations. Moreover, we benefit from k-means clustering in Chapter 4 and Chapter 5 to obtain simplified data representations and to introduce nonlinear relationships in the original data to the classification model. In Chapter 5, we present a detailed overview of mathematical programming approaches to MIL, which commonly extend support vector machines to MIL setting.

2.1. Problem Description

Let $\mathbf{x}_i$ be a d-dimensional feature vector of instance i and $X = \{\mathbf{x}_i : i = 1, .., n\}$ be a set of instances. Also let y_i be a single, discrete-valued feature, specifically the label of instance i. Then, instance set $X = \{\mathbf{x}_i : i = 1, .., n\}$ forms the training set. This set can be labeled with y_i, $i = 1, ..., n$ or can be unlabeled. A bag B_j is a set of $\mathbf{x}_i$'s and n_j is the number of the instances in B_j. Therefore, $\chi = \{(B_j, l_j) : j = 1, .., m\}$ is a training bag set containing instances and a label l_j of each bag. Let an instance-based classifier be a function from instances to labels $f(\mathbf{x}_i) \rightarrow y_i$, and let $g(B_j) \rightarrow l_j$ be the function of a bag-based single classifier. Concisely, given a training set of bags with given label information $\chi = \{(B_j, l_j) : j = 1, .., m\}$, our MIL task is to learn a classifier $g(B_j)$ to predict the labels of input bags.

2.2. Tree-based Ensembles

Decision tree learners are one of the classifiers that are used to partition the instance feature space to learn a bag-level representation in this study. Univariate trees such as CART [64] and C4.5 [65] are built using an algorithm performing a

series of axis-parallel splits, which are determined greedily by minimizing an impurity measure, and divide the space formed by the input vectors to subregions at each node of the tree.

Tree ensembles are proposed to mitigate the greedy nature of traditional decision tree learners. A random forest (RF) classifier [66] will be used in this study to classify the encoded bags. RF is a collection of multiple decision trees that are independent from each other and claimed to be robust compared to a single tree. RF is a special case of bagging (bootstrap aggregating) of decision trees, where at each node a random subset of features is selected for splitting purposes. Formally, a random forest is an ensemble of T decision trees, $\{r_t, t = 1, \ldots, T\}$. Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the single tree. These are called out-of-bag (OOB) samples. The prediction for instance $\mathbf{x}$ from tree r_t is $\hat{y}_t(\mathbf{x}) = \text{argmax}_c\, p_t^c(\mathbf{x})$, where $p_t^c(\mathbf{x})$ is the estimated proportion of class c in the corresponding leaf of the t-th tree, for $c \in \mathcal{C}$. Let $G(\mathbf{x})$ denote the set of all trees in the RF, where instance $\mathbf{x}$ is OOB. The OOB class probability estimate of $\mathbf{x}$ is

$$p^c(\mathbf{x}) = \frac{1}{|G(\mathbf{x})|} \sum_{r_t \in G(\mathbf{x})} 1(\hat{y}_t(\mathbf{x}) = c)$$

where $1(\cdot)$ is an indicator function that equals one if its argument is true, and zero otherwise. The predicted class is $\hat{y}(\mathbf{x}) = \text{argmax}_c\, p^c(\mathbf{x})$. In summary, an instance is labeled through a majority voting approach using the tree results for which it is OOB.

Random tree (RT) embedding is a variation of tree-based ensembles, where features to be splitted and the corresponding splitting points are selected randomly. RT-encoding works under the unsupervised setting, and provides a sparse representation of the instance feature space. Sparsity of the data representation is a desired property for classification to support linear separability in a new space. In some MIL datasets, non-linear boundaries are inevitable since there exists common concepts between classes and complex objects are represented by multiple instances. Thus, projecting the datasets

to a higher dimensional feature space is beneficial. Instances are embedded to the new space by encoding binary vectors extracted from node visits of the instances for each tree in the forest. When the nodes in a tree are considered as instance clusters, nonlinear separability is provided by this representation.

2.3. Support Vector Machines

SVMs are built on the idea of maximizing the margin between the convex hulls of the classes. The margin boundaries are represented by some instances from each class. These instances are referred to as the support vectors. It is shown in [67] that support vector set contains all the information that a classifier needs to build the decision function.

Formally, given $\chi = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, SVM classification finds a hyperplane $H(\mathbf{w}, b) = \{\mathbf{x} : \mathbf{w}^T\mathbf{x}+b = 0\}$, which is formed by selected $\mathbf{w} \in \Re^d$ and $b \in \Re$ to separate the classes. Then, the goal is to build a decision function $f(\mathbf{x}) = sgn(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ by maximizing the misclassification margin $\frac{2}{||\mathbf{w}||}$, or equivalently minimizing a quadratic loss $\frac{1}{2}||\mathbf{w}||^2$ subject to the constraints $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ for $i = 1, \ldots, n$. Note that the described margin is a hard margin since there exists no misclassified instances after training the SVM. If we allow misclassifications, soft margin takes place with resolving a trade-off between the misclassification margin width and number of misclassified instances by minimizing $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$ subject to the constraints $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ for $i = 1, \ldots, n$. The trade-off parameter is C, which controls the effect of slack variables $\xi_i \geq 0, i = 1, \ldots, n$ to the solution. The Lagrangean dual of this formulation is $L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n} \alpha_i\alpha_j y_i y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ where $\alpha_i \geq 0,\ i = 1, \ldots, n$ are the Lagrange multipliers satisfying $\sum_{i=1}^{n} \alpha_i y_i = 0$.

Generally, the classes are not linearly separable in the underlying space. In order to obtain a separation in another space, an evaluation of a kernel function maps the feature space to a higher dimensional embedding space $\mathcal{H}$ with a mapping $\phi : \Re^d \to \mathcal{H}$ by calculating the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in the dual formulation using a selected kernel function $K(\cdot, \cdot)$. In other words, the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in the formulation can be

replaced with corresponding evaluation of a kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.

2.4. K-Means Clustering

Clustering is an unsupervised process that groups together similar objects so that intra-cluster objects are similar, whereas inter-cluster objects are dissimilar to each other. K-means method employs Euclidean distance as a measure of similarity and starts with a random or previously determined initial partition of the data. The aim is to minimize the sum of squared distances between each point and its nearest cluster center.

The algorithm of k-means clustering is a hard partitioning algorithm, which divides a dataset into a set of exhaustive and mutually exclusive clusters. That is, for a given dataset $X = \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, k-means algorithm iteratively divides X into κ clusters $C_1, \ldots, C_\kappa$, subject to $X = \cup_{i=1}^{\kappa} C_i$ and $C_i \cap C_j = \emptyset$, for all $1 \leq i \neq j \leq \kappa$. The steps of k-means clustering algorithm are summarized below:

Input: An instance set $X = \{\mathbf{x}_i : i = 1, \ldots, n\}$, number of clusters κ
Initialize the centers of clusters $\mathbf{c}_j, j = 1, \ldots, \kappa$ randomly.
Repeat
  for all $\mathbf{x}_i, i = 1, \ldots, n$ **do**
    Assign point $\mathbf{x}_i$ to the nearest center such that
    $\mathbf{x}_i \in C_j \iff j = \text{argmin}_{j \in 1, \ldots, \kappa} ||\mathbf{x}_i - \mathbf{c}_j||$.
  end for
  for all $\mathbf{c}_j, j = 1, \ldots, \kappa$ **do**
    Recalculate the cluster centers as
    $\mathbf{c}_j = \frac{1}{|C_j|} * \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$.
  end for
Until $\mathbf{c}_j, j = 1, \ldots, \kappa$ converge.

Figure 2.1. K-means algorithm.

3. BAG ENCODING STRATEGIES FOR MULTIPLE INSTANCE LEARNING

3.1. Introduction

Classification, one of the important class of supervised learning problems, often takes place in data mining tasks. In traditional classification tasks, each object is represented with a feature vector, and the aim is to predict the label of the object given some training data. However, this representation is not flexible when the data has a certain structure. For example, in image classification, images are segmented into patches and instead of a single feature vector, each image is represented by a set of feature vectors derived from the patches. This way, important information regarding the certain invariances such as location and scale can be taken into account [68]. Change of object representation provides benefits for a wide range of applications such as bioinformatics [5], document retrieval [7], computer vision [69] and etc. This type of applications fits well to *Multiple Instance Learning* (MIL) setting where each object is referred to as *bag* and each bag contains certain number of *instances*.

Most of the MIL approaches generally solve the binary classification problem, where bags are labeled as either positive, or negative [5,13,30,43]. The firstly described formal MIL problem is a drug activity prediction problem, which considers molecules as bags and distinct shapes of the same molecule as instances [5]. A molecule is positively labeled if it includes at least one effective shape, otherwise it is negatively labeled. In text categorization problems [56], each document can be considered as a bag and its instances are the collection of relevant passages inside it. In all these applications, training bags are labeled and instances belonging to each bag do not necessarily have labels. The aim of MIL is to learn a classifier on the training bags to predict the label of a test bag.

Ambiguity about the instance labels has made researchers focus on certain assumptions regarding the instance labels. The so called *standard MIL assumption* is given as: if a bag is labeled positive, then at least one instance in that bag is labeled as positive; otherwise, labels of all instances in negative bags are negative [5]. It is obvious that when a bag is known to be positively labeled, the labels of its instances are not completely known. Standard MIL assumption is too restrictive to handle real-life problems. For example, optimal combination therapy is used in cancer treatment to overcome drug resistance. An optimal combination of drugs is considered to be capable of circumventing drug resistance among individual patients. Since there exists enormous number of possible drug combinations, the prediction problem of optimal combinatorial therapy can be modeled as a MIL problem where drugs are the instances, and collections of drugs are the bags. A bag is positive if a subset of its instances forms an effective drug combination, otherwise the bag is negative. Optimal combination therapy discovers an effective combination of drugs, rather than identifying a single type of drug that supports the treatment. Instead of a single positive instance, this MIL problem searches for a combination of multiple instances referring to various drugs.

Criticizing the potential problems with the standard MIL assumption, MIL problems are categorized as *presence-based*, *threshold-based* and *count-based* MIL problems [20]. A specific region of feature space where the positive instances are located are referred to as a *concept* by [20]. *Presence-based* MIL has the standard MIL assumption for multiple concepts, whereas *threshold-based* MIL forces a lower bound on the number of necessary instances of each concept. Finally, in addition to the previous lower bound, *count-based* MIL requires an extra upper bound on the number of necessary instances from each concept. Extensions and variations of the described categorization of generalized MIL problems are presented in [21, 22, 24]. Based on the experiments on synthetic and real datasets following various assumptions, the bag-level classification is indicated to be successful on datasets from different categories [24]. These approaches require each bag to be represented with a feature vector which summarizes the instance level information. Since bag-level methods are competitive, we focus on bag classification by representing each bag with a single feature vector in this study.

Earlier, many approaches from the computer vision literature utilized the well-known Bag-of-Features (BoF) or Bag-of-Words (BoW) representations to perform similar tasks. After clustering the patches (i.e. instances), the image (i.e. bag) is represented by the frequency of cluster assignments of the corresponding instances in the simple BoW setting [70]. These approaches implicitly transform the instance-level probabilistic distribution information to a bag-level summary [13, 50]. Recently, [26] has approached the problem by considering the geometric view of the instance space and obtain a bag-level summary using the similarities between the instances. Motivated by the success of the bag-level representations and their robustness to the MIL assumptions, this study proposes bag encoding strategies for MIL problems [71]. Figure 3.2 presents a summary of the bag representation algorithms, each of which will be discussed in detail in Section 3.2.

Most of the existing proposals to obtain bag-level summary require numerical features as an input since they involve transformations such as principal component analysis [50], density estimation [13] or distance calculations [26,50]. However, a MIL dataset can have features other than numeric. When there are categorical features, dummy variables are required to be introduced. Moreover, standardization/normalization is required but standardization of the dummy variables introduced to represent categorical variables is not well-defined. Hence, an approach that can treat each variable without any modification may be required for certain applications. Considering this fact, our approach utilizes tree-based ensembles to partition the instance feature space. A tree learner trained on the raw data assigns each instance to a terminal node of the tree.

Use of trees for feature induction is a relatively new research direction, which is also named as hashing [72]. This method transforms each node in the tree to a feature. Moreover, the new representation is easy to be modified by changing the tree parameters. Each level of the tree provides a different partition of the instance feature space as they imply simple splitting rules on the features. An instance traverses the tree based on the splitting rules (i.e. follows a path in the constructed tree). The *path* followed by an instance implies regions of the feature space an instance belongs to and it provides an hierarchical information regarding the feature space an instance

resides. Tree-based encoding of the feature space does not require scaling of the data as opposed to the approaches requiring distance calculations or density estimation. Figure 3.1 illustrates the path-encoding of an instance. Next to each tree in Figure 3.1, the traversed path by an instance is detected, and a binary vector is encoded conceiving whether a node is on that path, or not. Thus, these paths can be used to learn a BoW type representation. Our approach inherits the properties of tree-based learners. That is, it can handle numerical or categorical data. Besides, tree-based encoding is scale invariant and robust to missing values. The same tree can be used to encode the instances based only on terminal nodes. Earlier, [73] used a similar strategy for image classification problems using supervised randomized trees and has shown to provide successful results. This strategy has potential to lose information since two instances residing at sibling terminal nodes follow the same path, and therefore, they are closely similar to each other.

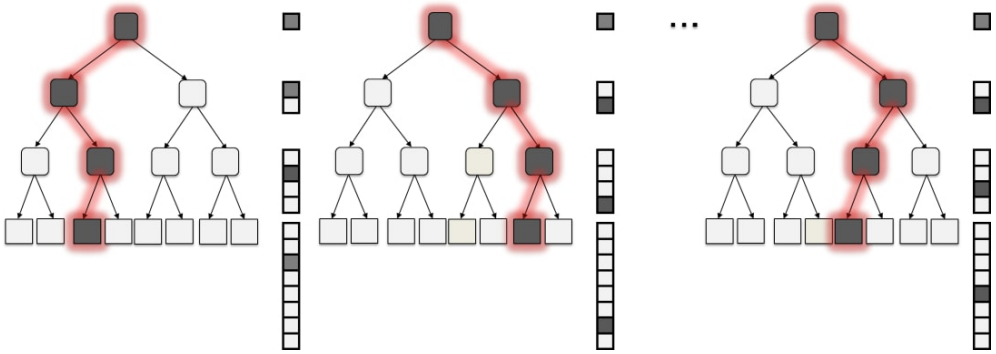


Figure 3.1. An example of path-encoding.

The first implementation of tree-based encoding to generate bag representations is proposed by TLC [20] in MIL setting. However, TLC builds a supervised tree on all instances in the training data assuming that the instances share the same label with their owner bags. This strong assumption regarding the instance labels and greediness of a single tree is problematic. To avoid potential problems with these assumptions, we propose randomized tree ensembles to convert MIL problem into a supervised learning problem. To best of our knowledge, this is the first study exploring the use of multiple unsupervised trees together with path-encoding to solve MIL problems. As shown by the seminal work by [74], unsupervised randomized trees are a generalization of

Gaussian Mixture Models (GMM), where each leaf of a randomized clustering tree is considered as a Gaussian component. Hence, our representation implicitly takes the density information into account. This way, parametrized optimization processes that are common in generative learning models are avoided in tree-based encoding. The proposed approach scales well with large datasets and it is embarrassingly parallel. Once the bags are encoded, a supervised learning algorithm can be trained on the new representation. Our experiments on a large database of MIL problems show that performance of the proposed representations is competitive with the state-of-the-art algorithms. Classification of bags instead of individual instances is reasonable while solving MIL problem on large datasets. We also present experimental results on PASCAL Visual Object Classes (VOC) 2007 dataset [69] to verify the scalability of our proposed bag encoding algorithms.

The earliest MIL algorithm [5] maximizes the number of positive instances residing in a single axis parallel rectangle (APR), and minimizes the number of negative instances inside APR. Then, Diverse Density (DD) algorithm is proposed in [13], where positive instances are assumed to follow a Gaussian distribution. In DD and its variant EM-DD [30], gradient descent with multiple starts is employed to maximize the diverse density, which is the aggregation of closeness of instances to every positive bags' distance to the negative bags. MILBoost [43] solves MIL problem with deriving the bag probabilities through a Noisy-OR model. A dictionary-based MIL method [75] benefits from bag-labels and inactivates irrelevant instances during dictionary learning for bag representation. These approaches follow the standard MIL assumption and it has been discussed in many papers that this assumption may not reflect the reality for certain types of MIL problems. Hence, the methods requiring standard MIL assumption are left out of scope of this study. We refer reader to [21, 22] for details of the methods requiring the standard MIL assumption.

Use of bag-level representations as an input to supervised learning algorithms are shown to perform well in MIL problems [8, 26–29]. Exploitation of the similarity or dissimilarity information relevant to bags and their instances forms the common ground of these methods. For instance, MILES [8] measures bag similarities by their minimum

instance distances. Instances in the training set are used as prototypes, and the bags are represented by using distances to these prototypes. Since the distances to all instances in the training set are calculated, MILES is time consuming and become impractical for large datasets. The instance selection in MILDS [27] is done by identification of the most representative examples in the positive and negative training bags using a pairwise clustering algorithm. A kernel density estimator on the negative instances is used for instance selection in MILIS [28]. MILD [29] identifies the true positive instances in positive bags to reveal instance-level and bag-level representation schemes. Another instance selection method, Random Subspace Instance Selection (RSIS) [55] finds random subspaces using clustering to identify witness instances in positive bags and determines bag labels by training an ensemble of SVM classifiers on subsets of instances.

A recent dissimilarity-based method D^{RS}-SVM [48] finds dissimilarities of bags in a random subspace (RS) formed by random selection of instances. However, calculated dissimilarities may include noisy or uninformative features, which may deteriorate the performance of the final classifiers. Another recent method, MInD [26] solves MIL problem using different dissimilarity-based bag representations. Bags are compared by defining various bag-to-bag dissimilarity measures. Then, logistic regression or SVM classifiers are employed for bag classification. Utilization of alternative bag dissimilarity functions provides diversification across MIL datasets with different characteristics. MInD [26] performs significantly better than the existing MIL methods.

Another class of approaches introduce graph-based representations to solve MIL problem. Similar to [26], spatial relationships between instances are modeled using similarity information. miGraph [56] builds a graph where nodes are the instances, and edges are weighted by the affinities between the instances. A graph kernel is defined in miGraph [56] by forming cliques of similar instances. MIGraph [56] also employs a graph kernel to capture underlying manifold structure of the data. In addition to the instance relationships, a MIL approach on structured data (MILSD) [57] examines the relational structure between bags, or instances in different bags. A recent graph-based proposal targets MIL via multi-graph learning such that every bag consists of several

graphs [60]. Representative subgraphs are generated in various ways to differentiate positive and negative bag classes.

The closest works to our bag representation approaches also follow simple strategies to transform instance feature space [20, 49, 50]. Then, bags are represented by the number of instances in the transformed space. A two step classification algorithm, TLC [20] represents the bags by their instances using a standard decision tree. The first step learns instance-level concepts using a decision tree. The tree is built on the instances in training bags after setting the label of each instance as the label of its owner bag. This study trains a single tree in a supervised manner, and fails to be generalized to the MIL problems with assumptions other than the standard MIL assumption [20]. The second approach, constructive clustering ensemble (CCE) [49], clusters the instances to encode the bags by a binary feature vector indicating that whether a bag has an instance in a cluster or not. A feature value is set to one if at least one instance of the bag appears in the corresponding cluster. This method is very similar to the BoW representation with k-means clustering, namely k-means-encoding. However, CCE [49] takes only the presence of the cluster member into consideration, which might be problematic for the MIL problems encoding threshold-based or count-based assumptions. CCE [49] repeats clustering to encode much more information, and obtains an improved representation in a higher dimensional space. In k-means-encoding, all the training instances are clustered into a fixed number of clusters. Then, each bag is considered individually by counting its neighboring instances to each cluster center. The encoded bag vector conveys more precise and distinctive information and its dimension equals to the number of clusters. A single repeat of clustering in k-means-encoding is enough to generate better results compared to CCE [49].

Finally, MIL based on the Fisher vector representation (miFV) is proposed in [50], where the bags are mapped to a higher dimensional feature space by benefiting from a Fisher kernel in a GMM. In miFV, Fisher vector (FV) is characterized as a gradient vector describing a bag. Generation of the FV is modeled by a density function. Parameters of the density function are estimated by a GMM. Density forests are a generalization of GMMs, where each leaf of a randomized clustering tree is considered as a

Gaussian component. A comprehensive study on random decision forests [74] demonstrates that better density estimations can be achieved by generating unsupervised randomized trees. In that sense, both approaches share similar ideas. Partitioning the instances inside bags by a random tree ensemble is capable of representing bags without imposing any parameter estimation. It also pursues the advantages of tree based learning such as fast application without data preprocessing, handling all types of data and easy interpretation. The resulting bag vector is numeric and any classifier can be trained on the new representation. Although miFV scales well with large number of instances, its time and memory utilization rapidly increases on datasets with large number of features due to the covariance calculations.

The remainder of this chapter is organized as follows: Section 3.2 introduces the bag representation schemes and the proposed solution algorithms. Density modeling success of randomized trees are discussed in Section 3.3. Description of real world datasets, the results of the carried out experiments followed by parametric and computational analysis are demonstrated in Section 3.4. Finally, Section 3.5 draws the conclusions.

3.2. Multiple Instance Learning with Bag Encoding

3.2.1. Summary of Bag Encoding for MIL

The proposed approach has two main stages: *bag encoding* and *bag-level classification*. The first stage provides bag representation vectors $\{\mathbf{b}_j : j = 1, \ldots, m\}$ by transforming the instance level information. The bag set can be described as $\mathbf{B}' = \{\mathbf{b}_j : j = 1, \ldots, m\}$, which will be denoted as the encoded bag set. The second stage learns a classifier $g(\mathbf{b}_j)$ to predict the labels of input bags. The underlying technical steps of *bag encoding* are illustrated in Figure 3.2, consisting of k-means-encoding, path-encoding and terminal node-encoding. Initially, all of our proposals for bag encoding simply partition the feature space of instances as shown in stage one of Figure 3.2. In k-means-encoding, k-means algorithm is utilized to cluster the instances. In RT-based encoding, multiple randomized decision trees are built to parti-

tion the instances into different nodes. This partitioning can be performed using either terminal nodes, which cannot be splitted any further, or all the nodes in trees of the RT-ensemble.

After partitioning the instance feature space, bags are represented as the number of instances residing at each partition, as shown in stage two of Figure 3.2. In k-means-encoding, bags are represented by the number of instances belonging to different clusters. In a tree, each node on a path or a terminal node indicates an instance space region. Bags are represented either by the path vectors, or the terminal node vectors. Namely, number of instances residing in a terminal node, or a leaf node on the traversed path forms a feature of the bag-level representation, as illustrated for a single bag in stage two of Figure 3.2.

Once the bag representation is obtained, we utilize an RF classifier for *bag-level classification*, which scales well with large number of features and instances. After obtaining the encoded feature vector for training bags in the first main stage, a supervised RF is learned as a final bag classifier in the training phase. For testing, the bag-level representation is obtained after mapping instances of a test bag to the new feature space using the trained encoding model. Then, the test bag is classified by simply traversing the trees of the RF classifier. All aforementioned encoding schemes output simple and sparse vectors describing internal structures of training bags through time-efficient computations.

3.2.2. A Baseline Encoding Approach: K-Means-Encoding

Success of BoW representations using k-means-encoding has been illustrated for several computer vision applications [70]. As a baseline approach, k-means-encoding is considered for comparison purposes. We provide the details of k-means-encoding, and a simple extension of it. In the original k-means-encoding, instances are clustered into κ clusters and each instance is assigned to its closest cluster center based on Euclidean distance. Each bag is then represented by a feature vector counting the number of its instances' cluster assignments. In other words, a feature vector of size κ represents

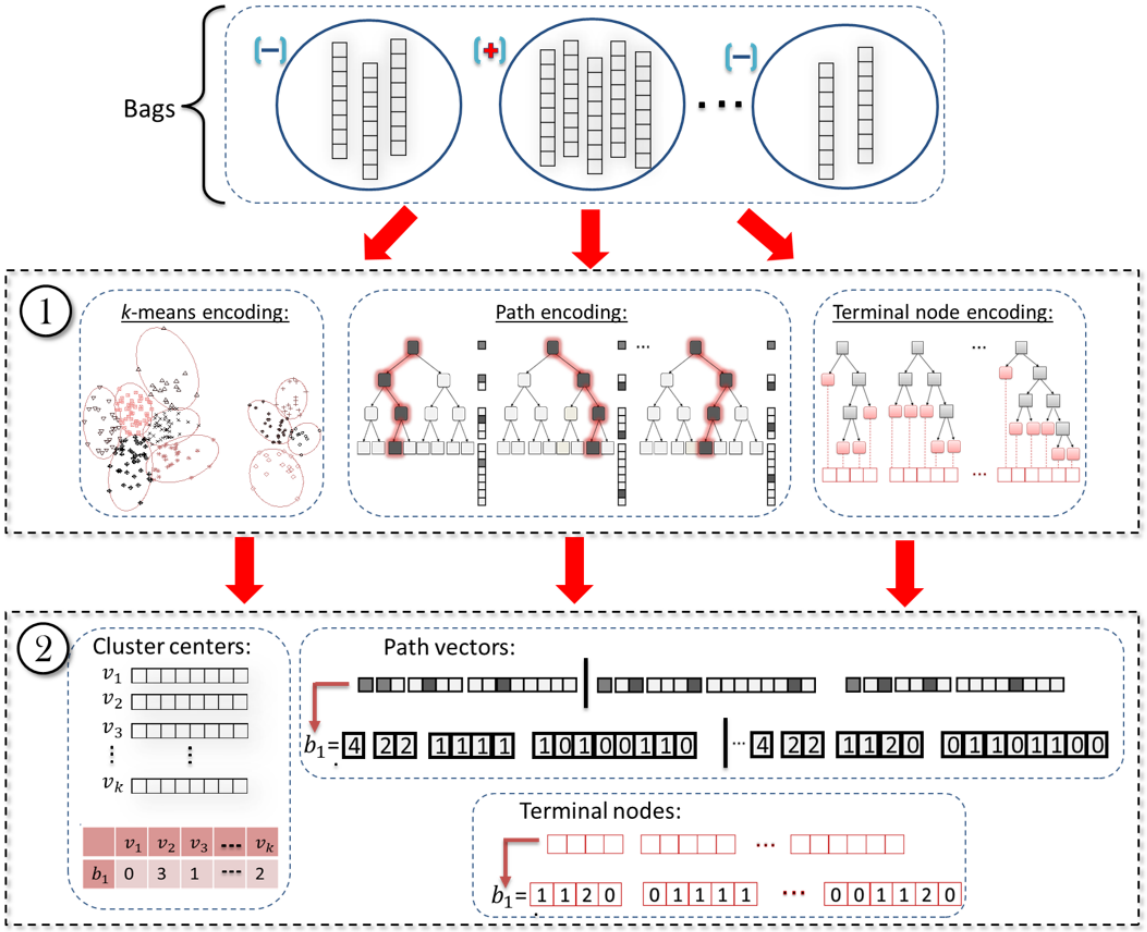


Figure 3.2. Bag representation algorithms.

each bag, entry of which counts the number of instances of the bag that are one-nearest neighbor to the corresponding cluster center. miFV [50] uses a similar information. However, since the bags are represented by using Fisher vectors, the information loss is less when compared to the frequency-based representation.

We also extend the traditional k-means-encoding, in which the instances of positive and negative bags are clustered separately. Since our MIL experiments are on binary classification datasets, we entitled this encoding algorithm as k-means two class-encoding. Bags are represented with a feature vector of size 2κ with this encoding. For MIL problems following the standard MIL assumption, this approach provides benefits since potential noisy instances in positive bags cannot distort the clustering of the instances in negative bags. Since both approaches require similarity calculations, they

also suffer from the same problems discussed for similarity-based approaches.

3.2.3. Random Tree (RT) Encoding

To avoid from aforementioned problems with the earlier proposals, we make use of tree learners as they are robust to different variable types, noise and missing values. Each terminal node of a decision tree defines axis-parallel decision boundaries based on certain rules learned on the training data. Furthermore, each non-terminal node of a tree agglomerates the similar instances. As in BoW representations, the instance frequencies at terminal nodes constitute the new representation. For each tree, the count of the instances at each terminal node can be used to represent the bag as shown in Figure 3.4(a).

Our tree learners (i.e. ensemble) do not use the bag labels and partition the feature space in a randomized manner. Given the depth of the tree (h), we select one random feature and a random splitting point at each tree building step. In order to capture the information from different regions of the feature space, we train T trees. This strategy is computationally efficient and multiple concepts can be explored through the randomization. The steps of RT-encodings are described in Figure 3.3. Compared to approaches requiring distance calculations (i.e. k-means-encoding), RT-encodings are computationally more efficient. In particular, RT-encodings only require the traversal of trees in the forest to determine the corresponding terminal nodes or visited paths. All bag vectors are normalized with the total number of instances residing in the corresponding bag.

A terminal node of the tree implies a region in the feature space, which can be considered as a cluster. However, since each terminal node is considered as a separate cluster, the relational information between these clusters is lost. To avoid this problem, we propose a path-based representation as described in Figure 3.4(b). Suppose there are seven instances in a bag, and the instances reside in the corresponding terminal nodes. Consider two instances following the same path except the last split (i.e. instance 1 and 7). Although these instances share similar information, they reside in different

Input: A bag set $\chi = \{(B_j, l_j) : j = 1, \ldots, m\}$, number of trees T

Output: RT-encoded bag set $B' = \{\mathbf{b}_j : j = 1, \ldots, m\}$

Build T unsupervised random trees of depth h using instance features

for all bags B_j **do**

  for all instances $\mathbf{x}_i \in B_j$ **do**

    Find the node frequencies of $\mathbf{x}_i$ by traversing the trees

  end for

  Terminal node-encoding: Construct the RT-encoded bag vector $\mathbf{b}_j$ by combining the terminal node frequencies

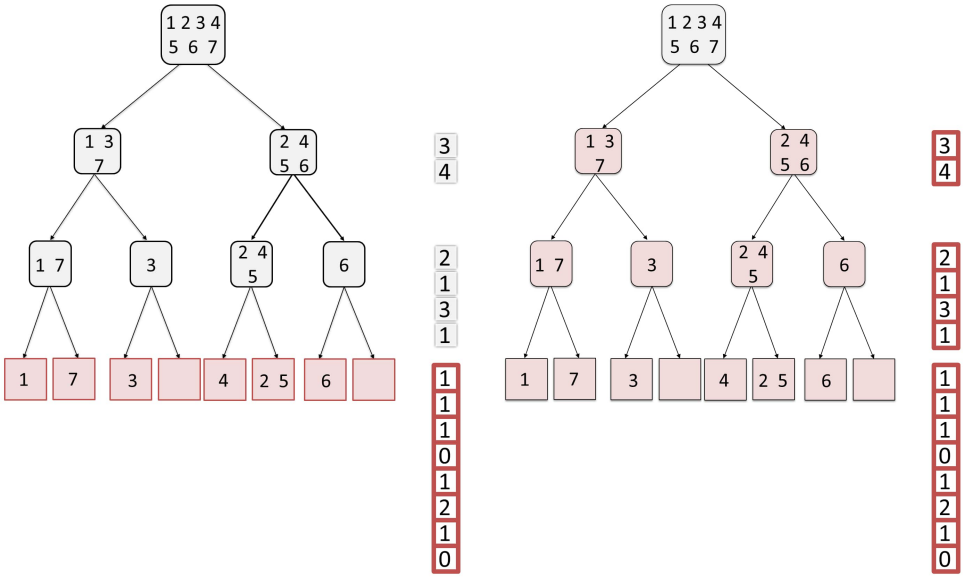
  Path-encoding: Construct the RT-encoded bag vector $\mathbf{b}_j$ by combining the path node frequencies

end for

Figure 3.3. RT-encoding algorithm.

terminal nodes. Hence, a representation based on the terminal nodes has potential to lose information. We encode each instance according to the visited nodes by that instance. The encoding basically forms a binary vector indicating whether the rule leading to a node is satisfied by an instance or not. Figure 3.4(b) illustrates the path-based representation. As opposed to the terminal node-encoding with a feature vector of length eight, instance frequencies at all nodes on the traversed paths constitute the new representation, which forms a feature vector of length fourteen.

In Figure 3.4(b), a sample bag has seven instances. For each instance in Figure 3.4(b), all trees in the forest are traversed. For each path in a tree, the nodes in this path are converted to a binary feature vector, which takes value one for a visited node, and takes value zero otherwise. For each instance in a bag, the path vectors are encoded as illustrated in Figure 3.4(b). Accordingly, the path vectors are summed to obtain the final bag representation in a tree. Each tree is traversed by all instances in a bag. When we consider the traversal of a tree for a single bag, all instances visit the root node of the tree. Since we count the visits of instances at each node and the root node is visited by all instances, the feature value corresponding to the root node will be the same for all trees. Hence, collected traversal information of the root nodes of



(a) Terminal node-encoding (b) Path-encoding

Figure 3.4. An example of bag-level RT-encoding.

trees is excluded from the representation. As a last step, constructed vectors of each tree in the ensemble are concatenated to obtain path-encoding of bags.

3.2.4. Classification

After the bags are encoded, a classifier is trained on the new representation. Depending on parameter settings, the number of variables in the bag-level representation might be very large. Therefore, a scalable classifier, which can handle large number of variables and their interactions such as RF is preferred for this task. Also, RFs are known to be robust to outliers [66] and they are embarrassingly parallel.

Alternatively, any classifier can be trained on the proposed representation. However, an important issue regarding path-encoding is the correlated features in the representation. Each node on a traversed path is related to its predecessor node. Resulting features emanating from the paths in RT-ensemble are correlated to each other in the final bag representation. Although this kind of encoding stores a richer information, potential problems related to feature redundancy reveal during bag classification. Therefore, we recommend to use a supervised learner that inherits a feature selection

procedure to classify bag-level feature vectors.

3.2.5. Computational Complexity

The time complexity of training an RT-ensemble takes $O(Tdh)$ times assuming that tree depth is h. Since we randomly select a single feature at each split, time complexity of RT generation becomes independent from d. Therefore, complexity $O(Tdh)$ of training RT becomes $O(Th)$. The testing complexity of RT ensemble equals to the complexity of traversing all of the trees, which takes $O(Th)$ times. For the ensemble, finding the frequencies for all instances takes $O(Tnh)$ times. Although the time complexity of obtaining two RT-encodings is the same, space requirements for path-encoding is large compared to terminal node-encoding. However, using a sparse matrix representation efficiently handles this difficulty.

3.3. Comparison of miFV and RT-encoding in Density Estimation

In miFV [50], density of the instance feature space is modeled by a GMM. The parameters of Gaussian components are estimated to obtain the Fisher vector (FV) representation. Similarly, unsupervised RTs perform instance partitioning to represent the data, where each leaf implies a Gaussian component. We provide a discussion on parameter insensitivity and density modeling success of randomized trees on a toy example in this section. As mentioned in [74], density forests are generalizations of GMM models, which estimate the densities by randomly partitioning instance feature space.

Our experiments include a toy dataset, which has 1500 points on 2D space implying a spiral shape. The main motivation behind this selection is to test both encoding approaches in highly nonlinear datasets. We expect miFV [50] performance to deteriorate since miFV [50] relies on covariance information, which is a linear measure. The nonlinearity can be captured by introducing more components in GMM, however estimation of parameters is problematic in such cases. Both of the approaches encode given data, and output a new high dimensional representation. miFV [50]

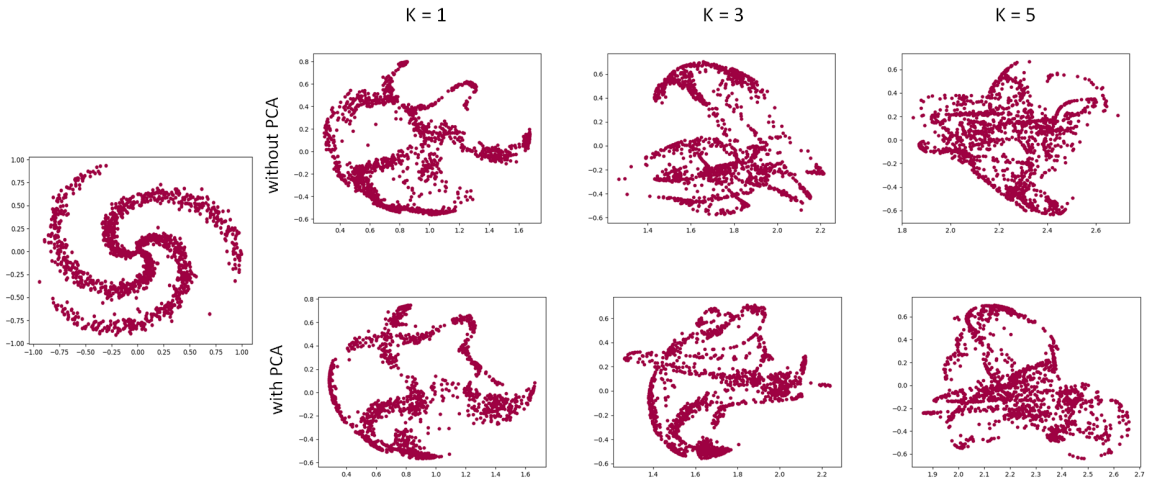


Figure 3.5. Application of FV-based encoding on spiral data. Input instances are plotted in the left. Other plots are 2D projections of FV-based encoding with varying number of components (K) performed with or without PCA.

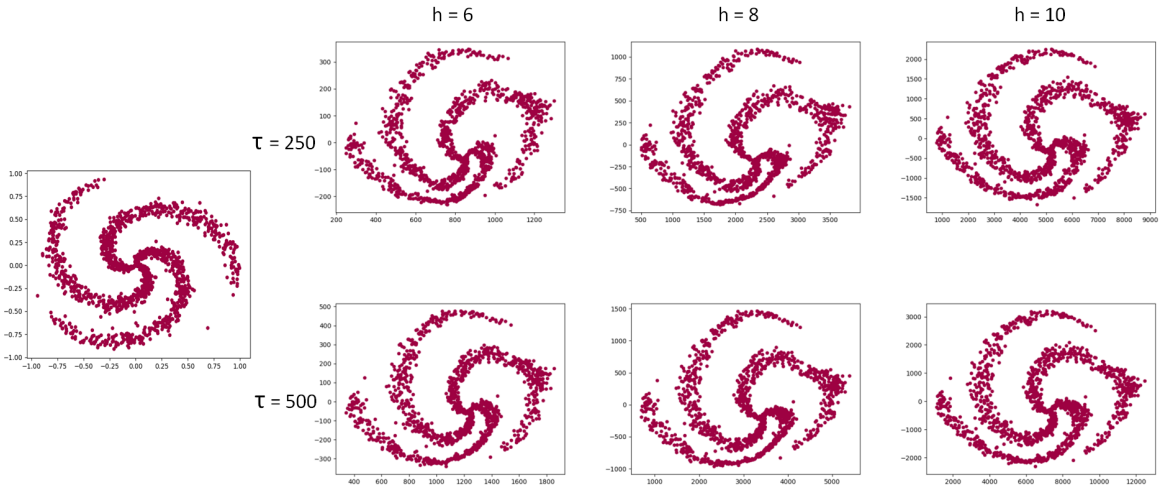


Figure 3.6. Application of terminal node-encoding on spiral data. Input instances are plotted in the left. Other plots are 2D projections of terminal node representations with varying number of trees (T) and tree depths (h).

estimates GMM parameters and obtains FVs, whereas a RT-ensemble is learned in terminal node-encoding. Then, principal component analysis is used to map the new representations on a 2D space to inspect the reconstructions. FV and terminal node representations are schematized in Figure 3.5 and Figure 3.6, respectively for different parameter combinations.

In FV-based transformation, increasing the number of components affects the reconstruction. However, the 2D projection is still a deformed version of the left plot of Figure 3.5. Additionally, applying principal component analysis (PCA) before learning the FV representation is also ineffective to describe the original data structure. Evaluated parameters of terminal node-encoding are number of trees and depth of the trees in the forest. Compared to original scatter plot on left, all remaining plots in Figure 3.6 demonstrate that the form of spiral data is reconstructed independently from varying depths and number of trees. The underlying nonlinear structure of spiral data is easily captured by terminal node representation. Unsupervised trees of RT-based encoding represent every instance with multiple partitions, and therefore, model the data with nonlinear relations. In GMM, Gaussian distribution of instances in the components is assumed. However, this assumption is not realistic for datasets with complex distributions, as in the case of spiral data.

As shown in Figure 3.6, terminal node representation preserves the spatial structure of the original data. In GMM, Gaussian components are generated from all instances with covariance matrices, mean vectors and mixture weights. After parameter estimation, FVs are obtained to map the input data to the new representation of miFV [50]. In contrast, RT generation does not require parameter estimation and is insensitive to user specified parameters as depicted in Figure 3.6. Besides, terminal node-encoding implicitly learns a generalization of GMM more accurately compared to miFV [50].

3.4. Experiments and Results

We test our proposals on a wide range of MIL datasets from different categories to avoid the application bias. Unfortunately, most of the MIL studies follow different strategies for experimentation, which complicates the comparisons. For instance, some studies do not report performance on certain datasets because of its computational requirements, others use different settings for the cross-validation or split train data randomly and report test performance. Due to this fact, a comprehensive database is created for the future work in this area. Most of the MIL datasets are provided by [26].

The datasets are considered under two categories to highlight the success and the scalability of our proposals. First category consists of the union of the datasets that are commonly used for comparison and it is referred to as "Common MIL datasets". The data characteristics and their references are summarized in Table 3.1. The second category includes "The PASCAL VOC challenge 2007" dataset, which consists of 9963 images containing 20 object classes in realistic scenes. The goal of the challenge is to predict object classes given the labeled natural images for training. To solve object recognition problem using MIL, we use the same setting as [76] including a training set of 5011 images and a test set of 4952 images. The bag classes are decided based on a one-versus-all approach in binary classification. Each image is a bag containing 50 region proposals, which are represented by a feature vector of size 500. Ultimately, there are 250550 training instances and 247600 test instances. The scalability of the proposed approach is illustrated on the datasets from this category.

We utilize a stratified ten-fold cross-validation to demonstrate the success of proposed and existing approaches. The performance of a classifier is evaluated based on the procedure described by [77], when there are multiple parameters to tune. For each fold in cross-validation, we run an inner cross-validation to tune the parameters. Once the parameters are optimized, we re-run training with the selected parameters, and thereafter measure the performance on the test fold. To promote reproducible research, our codes, datasets, results and the information regarding the cross-validation indices are made available on our supporting page [63], together with the implementations of k-means-encodings in R [78], and RT-encodings in Python using scikit-learn [79] library. Our second set of experiments illustrates the empirical performance of the encoding approaches in terms of their computation requirements and sensitivity to the problem characteristics and algorithm parameters in Sections 3.4.3 and 3.4.4.

Popular metrics used to compare the approaches in the literature are area under the receiver operating characteristic curve (ROC) [80], and accuracy. ROC curve plots the true positive rate versus the false positive rate of a classifier depending on a threshold parameter. Area under ROC curve (AUC) guides the learner to select the most suitable classifier for an individual dataset. When the number of test bags is small, [81]

emphasizes the necessity for AUC to measure the success of a bag-level classifier.

Moreover, bag classes are highly imbalanced in some MIL datasets such as Web recommendation [82]. Hence, calculated accuracies for cross-validation folds may be misleading. For instance, an audio recording dataset, Hermit thrush, has only 15 positive bags in a total of 548 bags. With stratified sampling, there is only one positive bag in each fold and classifying all bags as negative will achieve high accuracy. Therefore, our primary performance measure of interest is AUC. We replicate the cross-validation five times and report average and standard deviation of these metrics.

We compare the performance of tree-based encoding with the following approaches: BoW representations with k-means-encoding, citation-kNN [46], MILES [8], CCE [49], miFV [50] and Dissimilarity-based representations (MInD) [48] with D_{maxmin}, $D_{meanmin}$ and D_{minmin} dissimilarity measures. RSIS [55] is not included for comparison because of the run-time considerations. The computational effort of RSIS method is investigated in Section 3.4.4. Citation-kNN [46], MILES [8] and MInD [48] are implemented in PRTools [83] and the MIL toolbox [84]. Other compared approaches are also implemented in MATLAB [85]. It is important to emphasize that the same cross-validation indices are used for all approaches.

3.4.1. Parameter Settings

We modify the experimentation strategy proposed by [77] slightly for k-means and RT-encodings to avoid additional computation required for inner cross-validation. Instead of performing an inner cross-validation, the parameters are tuned based on the out-of-bag (OOB) performance of RF in the training fold. OOB predictions are known to provide a good approximation of the generalization error [66]. After obtaining the representation for each parameter combination, an RF is trained and the evaluation is done based on OOB performances.

Table 3.1. Common MIL datasets

Name	Instances	Min	Max	Features	Bags	+ bags	- bags
Musk 1 [5] ♣	476	2	40	166	92	47	45
Musk 2 [5] ♣	6598	1	1044	166	102	39	63
Mutagenesis 1 (easy) [86] ♣	10486	28	88	7	188	125	63
Mutagenesis 2 (hard) [86] ♣	2132	26	86	7	42	13	29
Protein [11] ♣	26611	35	189	8	193	25	168
Elephant [7] ♥	1391	2	13	230	200	100	100
Fox [7] ♥	1302	1	13	230	200	100	100
Tiger [7] ♥	1220	2	13	230	200	100	100
Corel, African [8] ♥	7947	2	13	9	2000	100	1900
Corel, Antique [8] ♥	7947	2	13	9	2000	100	1900
Corel, Battleships [8] ♥	7947	2	13	9	2000	100	1900
Corel, Beach [8] ♥	7947	2	13	9	2000	100	1900
Corel, Buses [8] ♥	7947	2	13	9	2000	100	1900
Corel, Cars [8] ♥	7947	2	13	9	2000	100	1900
Corel, Desserts [8] ♥	7947	2	13	9	2000	100	1900
Corel, Dinosaurs [8] ♥	7947	2	13	9	2000	100	1900
Corel, Dogs [8] ♥	7947	2	13	9	2000	100	1900
Corel, Elephants [8] ♥	7947	2	13	9	2000	100	1900
Corel, Fashion [8] ♥	7947	2	13	9	2000	100	1900
Corel, Flowers [8] ♥	7947	2	13	9	2000	100	1900
Corel, Food [8] ♥	7947	2	13	9	2000	100	1900
Corel, Historical [8] ♥	7947	2	13	9	2000	100	1900
Corel, Horses [8] ♥	7947	2	13	9	2000	100	1900
Corel, Lizards [8] ♥	7947	2	13	9	2000	100	1900
Corel, Mountains [8] ♥	7947	2	13	9	2000	100	1900
Corel, Skiing [8] ♥	7947	2	13	9	2000	100	1900
Corel, Sunset [8] ♥	7947	2	13	9	2000	100	1900
Corel, Waterfalls [8] ♥	7947	2	13	9	2000	100	1900
UCSB Breast Cancer [9] ♥	2002	21	40	708	58	26	32
Newsgroups 1, alt.atheism [56] ♠	5443	22	76	200	100	50	50
N.g. 2, comp.graphics [56] ♠	3094	12	58	200	100	50	50
N.g. 3, comp.os.ms-windows.misc [56] ♠	5175	25	82	200	100	50	50
N.g. 4, comp.sys.ibm.pc.hardware [56] ♠	4827	19	74	200	100	50	50
N.g. 5, comp.sys.mac.hardware [56] ♠	4473	17	71	200	100	50	50

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

Table 3.1. – Common MIL datasets (cont.).

Name	Instances	Min	Max	Features	Bags	+ bags	- bags
N.g. 6, comp.windows.x [56] ♠	3110	12	54	200	100	50	50
N.g. 7, misc.forsale [56] ♠	5306	29	84	200	100	50	50
N.g. 8, rec.autos [56] ♠	3458	15	39	200	100	50	50
N.g. 9, rec.motorcycles [56] ♠	4730	22	73	200	100	50	50
N.g. 10, rec.sport.baseball [56] ♠	3358	15	58	200	100	50	50
N.g. 11, rec.sport.hockey [56] ♠	1982	8	38	200	100	50	50
N.g. 12, sci.crypt [56] ♠	4284	20	71	200	100	50	50
N.g. 13, sci.electronics [56] ♠	3192	12	58	200	100	50	50
N.g. 14, sci.med [56] ♠	3045	11	54	200	100	50	50
N.g. 15, sci.space [56] ♠	3655	16	59	200	100	50	50
N.g. 16, soc.religion.christian [56] ♠	4677	21	71	200	100	50	50
N.g. 17, talk.politics.guns [56] ♠	3558	13	59	200	100	50	50
N.g. 18, talk.politics.mideast [56] ♠	3376	15	55	200	100	50	50
N.g. 19, talk.politics.misc [56] ♠	4788	21	75	200	100	50	50
N.g. 20, talk.religion.misc [56] ♠	4606	25	79	200	100	50	50
Web recommendation 1 [82] ♠	2212	4	131	5863	75	17	58
Web recommendation 2 [82] ♠	2212	5	200	6519	75	18	57
Web recommendation 3 [82] ♠	2212	5	200	6306	75	14	61
Web recommendation 4 [82] ♠	2291	4	200	6059	75	55	20
Web recommendation 5 [82]	2546	5	200	6407	75	61	14
Web recommendation 6 [82] ♠	2462	4	200	6417	75	59	16
Web recommendation 7 [82] ♠	2400	4	200	6450	75	39	36
Web recommendation 8 [82] ♠	2183	4	200	5999	75	35	40
Web recommendation 9 [82] ♠	2321	5	200	6279	75	37	38
Birds, Brown creeper [10] ♦	10232	2	43	38	548	197	351
Birds, Chestnut-backed chickadee [10] ♦	10232	2	43	38	548	117	431
Birds, Dark-eyed junco [10] ♦	10232	2	43	38	548	20	528
Birds, Hammonds flycatcher [10] ♦	10232	2	43	38	548	103	445
Birds, Hermit thrush [10] ♦	10232	2	43	38	548	15	533
Birds, Hermit warbler [10] ♦	10232	2	43	38	548	63	485
Birds, Olive-sided flycatcher [10] ♦	10232	2	43	38	548	90	458
Birds, Pacific slope flycatcher [10] ♦	10232	2	43	38	548	165	383
Birds, Red-breasted nuthatch [10] ♦	10232	2	43	38	548	82	466
Birds, Swainsons thrush [10] ♦	10232	2	43	38	548	79	469
Birds, Varied thrush [10] ♦	10232	2	43	38	548	89	459
Birds, Western tanager [10] ♦	10232	2	43	38	548	46	502
Birds, Winter Wren [10] ♦	10232	2	43	38	548	109	439

In k-means-encoding, the only parameter is the number of clusters (i.e. κ), which is selected from the set $\{50, 100, 200\}$. RT-encodings require the setting of the depth h and the number of trees T to learn the representation. Although our proposal is robust to these settings, h is selected from the set $\{6, 8, 10\}$ and T is fixed to 500. For tuning h, 50 trees are used to obtain the encoding. Then, an RF with 500 trees is trained on the encoded bags and the best value of h is determined based on resulting OOB predictions. We further discuss the robustness of RT-encodings to the parameter selection in Section 3.4.3. Similar strategy is employed for tuning κ in k-means-encoding. Final bag-level classifier is an RF with 1000 trees.

Default parameters are used for citation-kNN [46], MILES [8], CCE [49] and MInD [48]. Based on the authors' advice, we select the parameters of miFV [50] by using an inner ten-fold cross-validation. The percentage of information left after PCA is selected from the set $\{0.8, 0.9, 1\}$ whereas evaluated number of Gaussian components are $\{1, 2, 3, 4, 5\}$. Finally, cost parameter levels for the bag-level linear SVM classifier are $\{0.05, 1, 10\}$.

3.4.2. Classification Accuracy

3.4.2.1. Path-Encoding vs Terminal Node-Encoding. RT-encodings aggregate the instance level information to obtain a new feature representation for bags. Both algorithms use the same information content, but their way of structural representation differs as described in Section 3.2.3. Dimensionality of the encodings does not directly depend on the dimensionality of the input data. Namely, size of the encoding vector depends on the parameters of the RT-ensemble according to the experimental setting provided in Section 3.4.1. Terminal node-encoding is a smaller sized representation compared to path-encoding, and therefore, renders computational simplicity.

We perform an experiment to compare the success of terminal node-encoding and path-encoding. Since similar information content is encoded with these representations, we generate equal number of trees, T for both to ensure a fair comparison. The value of h is determined as $\lceil \log_2 n \rceil$ and T is fixed to 50. Based on the AUC results of the two

encoding algorithms, we conclude that there is no significant performance difference between them. Table 3.2 serves the dominance results of RT-encodings based on their AUC rankings on each MIL problem category. We count the wins, ties and losses for both encodings on each problem category. Path-encoding is better in 43 datasets, whereas terminal node-encoding is better in 23.

In particular, terminal node-encoding obtains superior results on Web recommendation benchmark [82], which has the highest number of features as shown in Table 3.1. For many instances, most of the features rarely have a nonzero value since each feature is related to frequency of a word on the web page. In path-encoding of these datasets, nodes in lower depth levels form too general features in the bag-level representation. The inferior performance of path-encoding for Web recommendation datasets is mostly due to overfitting as the final random forest classifier tends to put more emphasis to the features from lower depth levels. In these datasets, number of bags is relatively small (i.e. 75) and class imbalance is abundant. Under these settings, proposed RF classifier suffers from overfitting. This is illustrated in Figure 3.7. For each Web recommendation dataset, we draw a bar plot of average AUC results after RT-encodings. The number of instances in minority class are also shown in Figure 3.7. Path-encoding has lower AUC values compared to terminal node-encoding especially when class imbalance is present. Please note that we report the average AUC for each fold (i.e. average of 50 AUC values) and each fold contains approximately 8 bags. Therefore, in Table 3.2, we observe the similar behavior in Protein [11], which has a severe class imbalance.

In contrast, path-encoding offers the following utilities. A frequent challenge in MIL problems is a low rate of concept instances in positive bags [55]. For instance, in Newsgroups datasets [56], most of the instances are not effective on bag class label. Thus, relationships between concept instances become substantial for bag-level classification [8, 26]. A possible way of identifying instance relationships is based on the use of decision trees. However, since number of the concept instances is low in Newsgroups datasets, these instances may not reside in the same terminal-node depending on the tree depth level. The authors of [72] argue that similar instances have a higher probability of jointly following the same path on a tree due to the random selection of a

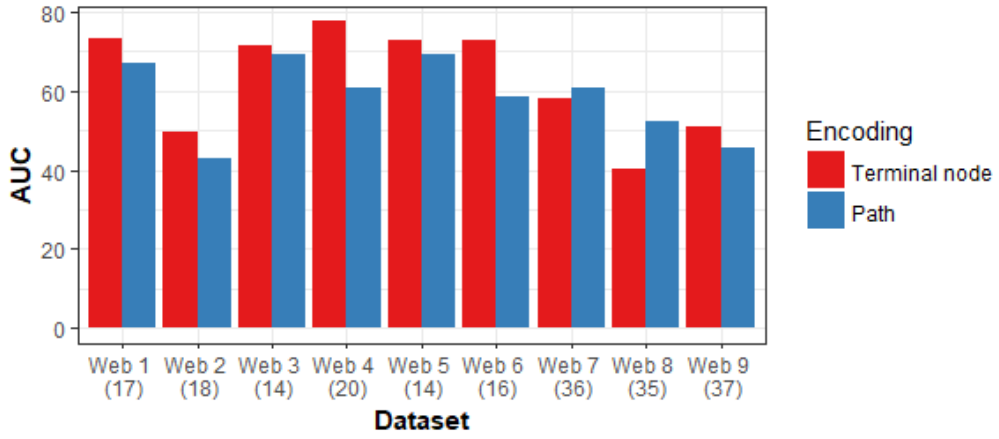


Figure 3.7. Bar plot of classification AUC after RT-encodings of Web recommendation datasets. y-axis show average AUC values for each dataset. x-axis list the dataset names (number of bags for minority class). Red bars indicate terminal node-encoding, and blue bars show path-encoding.

feature and a split value in each level. In a similar fashion, path-encoding identifies the similarities between informative instances using the richer information level obtained by the hierarchical representation as described in Section 3.2.3. Compared to terminal node-encoding, a better classification ranking of path-encoding in Newsgroups datasets is observed in Table 3.2.

3.4.2.2. Common MIL Datasets. We first compare the variations of k-means encoding and RT-encoding. k-means two class encoding and path-encoding provide more detailed information regarding the instance distribution compared to their counterparts as discussed in the earlier sections. Pairwise comparisons of AUC in Figure 3.8 support the superiority of these approaches. Each axis corresponds to a method and a dot represents the average AUC for a particular dataset. $x = y$ line represents the region, where both methods perform about the same. A point above the line indicates that the approach on the y axis has better accuracy than the one on the x axis for the corresponding dataset. As illustrated, k-means two class-encoding provides better AUC values than k-means-encoding in 42 datasets, and performs equally in two of them. On this regard, we determine to report and analyze the overall results only for

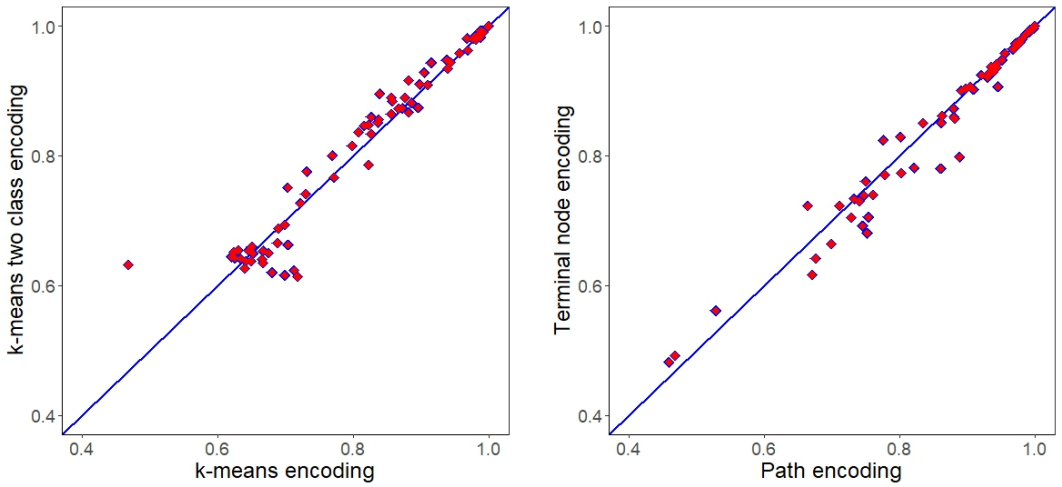
Table 3.2. Win-loss table of bag classification AUC results after RT-encodings of datasets from different problem categories. Each row represents wins, losses and ties of path-encoding compared to terminal node-encoding.

Dataset	Path-encoding		
	Wins	Losses	Ties
Musk ♣	2	0	0
Mutagenesis ♣	2	0	0
Protein ♣	0	1	0
Elephant, Fox, Tiger ♥	0	3	0
Corel ♥	15	4	1
UCSB Breast Cancer ♥	0	1	0
Newsgroups ♠	13	6	1
Web recommendation ♠	2	7	0
Birds ♦	9	1	3
Total	43	23	5

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

k-means two class-encoding. When RT-encodings are compared, path-encoding provides better results in 41 datasets, whereas the performance is equal for four datasets. Moreover, pairwise differences of RT-encodings are not statistically significant. We only consider path-encoding into multiple comparisons since proposed statistical comparison is affected negatively by addition of alternative approaches. Concordantly, with the discussions of Section 3.4.2.1, we exclude terminal node-encoding from the reported numerical results and statistical tests.

The performances based on AUC and accuracy are summarized in Tables 3.3 and 3.4, respectively. In most of the papers, AUC results are not reported for both the proposed or the state-of-the art algorithms. Only accuracy performances are reported on the paper of miFV, which is one of the methods included in our comparison. Since the probabilistic classification estimates in the original implementation are not realistic due to the used classifier, we do not report the AUC performance for this method. Comparison of multiple classifiers over all datasets is done using a procedure suggested by [87]. We first employ a Friedman test [88], followed by the Nemenyi test [89] if a significant difference in the average ranks is identified by Friedman test. This approach



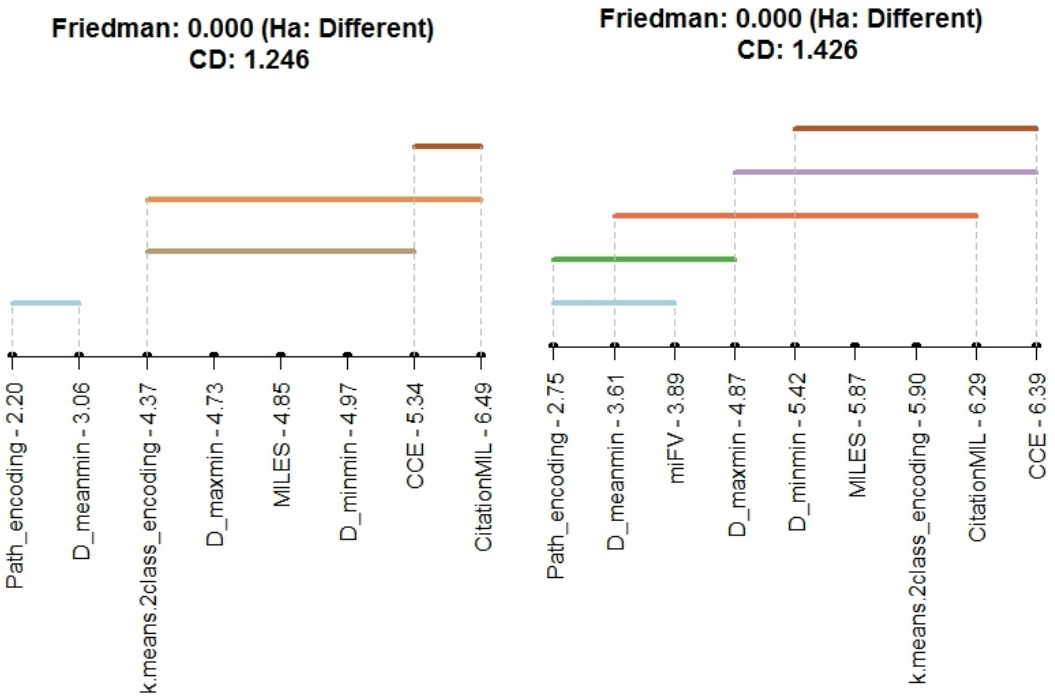
(a) AUC comparison of k-means-encoding and k-means two class-encoding

(b) AUC comparison of path-encoding and terminal node-encoding

Figure 3.8. Pairwise AUC comparison of similar bag encoding algorithms on 71 real-world datasets.

is a non-parametric form of Analysis of Variance based on ranks of the methods on each dataset.

Figure 3.9(a) and Figure 3.9(b) show the average ranks for all classifiers on 71 datasets based on AUC and accuracy, respectively. Based on the Friedman test, we find that there is a significant difference between the classifiers at 0.05 level. Proceeding with the Nemenyi test, we compute the critical difference (CD). This test concludes that two classifiers have a significant difference in their performances if their average ranks differ by at least the critical difference [87]. Figure 3.9(a) demonstrates that critical difference of AUC results at significance level 0.05 is 1.246. Path-encoding is significantly different from other methods, except $D_{meanmin}$. Path-encoding has best average rank, $D_{meanmin}$ is the second best and k-means two class-encoding is the third best as shown in Figure 3.9(a). When accuracies are compared, similar discussion holds for $D_{meanmin}$ and miFV is ranked third as shown in Figure 3.9(b). The accuracy performance of path-encoding is not significantly different from $D_{meanmin}$ and miFV at significance level 0.05, and corresponding critical value is 1.426.



(a) Mean AUC performance

(b) Mean accuracy performance

Figure 3.9. The average ranks for all classifiers on 71 datasets based on mean AUC and accuracy measures. The critical difference at 0.05 is 1.246 for AUC and is 1.426 for accuracy.

Path-encoding is quite successful compared to the other methods on molecular activity prediction datasets (i.e. Mutagenesis and Protein). In Musk 1, highest accuracy and AUC values are obtained by path-encoding, whereas MInD with $D_{meanmin}$ is the leading method in Musk 2. In most of the Corel datasets, path-encoding and $D_{meanmin}$ perform around the same. In other image datasets such as UCSB Breast Cancer, Elephant and Tiger, path-encoding has the best performance. Fox is one of the hardest datasets and results of path-encoding is very close the leading method MILES. In text classification, MInD performs better than bag encoding especially for Newsgroups datasets, where there are large number of instances and small number of features. However in Web recommendation datasets, there are higher number of features and less number of instances compared to Newsgroups and k-means two class-encoding performs better than others in most of the cases. In audio recording classification, path-encoding is the leading method with the best classification results on the extent of Birds datasets.

3.4.2.3. PASCAL VOC 2007 Dataset. Path-encoding is also evaluated on PASCAL VOC 2007 dataset to illustrate performance of the approach on large-scaled datasets. We also report the average Precision (AP) for this dataset since AP is the selected evaluation metric in PASCAL VOC 2007 challenge, as proposed in [69]. AP is given by area under the precision/recall curve. Table 3.5 summarizes the performance of path-encoding on 20 classes. As mentioned in previous sections, parameter selection part of miFV method is computationally expensive. For example, our experiment on miFV with the inner cross-validation takes more than 24 hours to encode training bags and learn the bag classifier. Similarly, D_{meanmin} involves distance calculations between instances and computation times increase significantly for this dataset in our experimental setup. Therefore, D_{meanmin} is also excluded from performance comparisons. Potential problems about the computational requirements of these approaches are further discussed in Section 3.4.4. Path-encoding provides satisfactory results with an average AUC of 97.0%, accuracy of 96.2% and AP of 81.5%.

3.4.3. Parameter Sensitivity

To elaborate on the robustness of RT-encodings to the parameter settings, we report the AUC performances with changing number of trees $T \in \{100, 250, 500, 1000\}$ and depth $h \in \{2, 4, 6, 8\}$. Figure 3.10 shows the average AUC results of RT-encodings on Elephant dataset. Both parameters control the size of the final bag representation, and a larger representation is expected to provide a more detailed view of the feature space. Increasing number of trees in the ensembles of both RT-encodings increases the AUC. Since Elephant is a small dataset with less number of instances, AUC value starts decreasing after a certain representation size. Although the decrease in the performance is not significant due to the randomness in tree learning, the approach slightly suffers from overfitting with the increase in the number of trees. On the other hand, the sensitivity to depth decreases significantly with sufficiently large number of trees. Nevertheless, our parameter selection procedure based on OOB probability estimates provides slightly better results with efficient computation times.

Table 3.3. AUC and standard error (× 100) results of various MIL methods.

Dataset	**Algorithm AUC (%)**							
			MInD				Bag encoding	
	Citation-kNN	MILES	D_{maxmin}	$D_{meanmin}$	D_{minmin}	CCE	k-means two class	Path
Musk 1 ♣	87.1 (1.7)	93.5 (1.0)	92.0 (1.9)	94.5 (1.2)	93.1 (1.1)	88.2 (1.7)	87.4 (1.6)	**95.6 (1.1)**
Musk 2 ♣	85.0 (1.4)	96.7 (0.7)	95.6 (0.8)	**97.6 (0.8)**	95.6 (1.0)	78.1 (2.0)	78.6 (2.0)	93.3 (1.2)
Mutagenesis 1 ♣	82.7 (1.5)	78.0 (1.5)	82.0 (1.4)	85.1 (1.2)	76.6 (1.5)	83.3 (1.6)	90.9 (1.1)	**93.5 (1.0)**
Mutagenesis 2 ♣	68.8 (4.3)	62.7 (5.0)	59.0 (5.7)	64.7 (5.3)	34.5 (5.2)	72.7 (4.6)	**86.7 (2.6)**	83.5 (3.9)
Protein ♣	55.2 (2.4)	**95.3 (1.1)**	56.1 (2.7)	52.3 (3.7)	87.6 (1.4)	64.3 (2.6)	87.3 (1.6)	94.6 (0.8)
Elephant ♥	88.2 (1.1)	92.7 (0.7)	87.6 (1.1)	93.6 (0.9)	91.5 (0.9)	85.4 (1.1)	88.2 (1.1)	**94.4 (0.7)**
Fox ♥	56.9 (2.0)	**73.8 (1.6)**	45.7 (1.3)	61.2 (1.7)	70.4 (1.6)	62.3 (1.6)	65.5 (1.7)	71.2 (1.4)
Tiger ♥	75.2 (1.5)	86.8 (1.0)	76.1 (1.3)	85.3 (1.1)	85.0 (1.0)	83.2 (1.1)	86.0 (1.2)	**91.0 (1.0)**
Corel, African ♥	86.2 (1.0)	95.7 (0.5)	93.9 (0.7)	96.7 (0.4)	96.6 (0.3)	84.0 (1.1)	85.1 (1.0)	**97.2 (0.3)**
Corel, Antique ♥	74.7 (1.0)	87.3 (0.7)	87.7 (0.8)	92.2 (0.6)	91.0 (0.6)	72.2 (1.3)	80.1 (1.0)	**93.4 (0.4)**
Corel, Battleships ♥	86.6 (1.0)	94.9 (0.5)	97.2 (0.3)	**98.1 (0.2)**	96.6 (0.4)	91.2 (0.9)	88.9 (0.8)	97.4 (0.3)
Corel, Beach ♥	92.6 (0.8)	**99.3 (0.1)**	97.2 (0.5)	98.3 (0.4)	99.0 (0.1)	96.9 (0.3)	98.0 (0.3)	99.1 (0.2)
Corel, Buses ♥	84.8 (1.1)	97.5 (0.4)	96.2 (0.4)	97.3 (0.4)	98.0 (0.3)	96.3 (0.4)	94.4 (0.5)	**98.2 (0.2)**
Corel, Cars ♥	83.5 (1.0)	91.9 (0.7)	93.0 (0.7)	94.8 (0.5)	91.6 (0.7)	87.8 (0.9)	86.4 (0.9)	**95.1 (0.4)**
Corel, Desserts ♥	86.6 (0.9)	93.9 (0.7)	95.0 (0.7)	97.4 (0.3)	96.6 (0.4)	92.2 (0.7)	94.3 (0.5)	**97.5 (0.3)**
Corel, Dinosaurs ♥	80.2 (1.0)	97.5 (0.3)	94.8 (0.5)	**98.3 (0.2)**	98.2 (0.2)	89.7 (0.6)	91.6 (0.6)	97.3 (0.3)
Corel, Dogs ♥	75.1 (1.1)	87.4 (1.1)	89.4 (0.8)	91.9 (0.7)	91.1 (0.6)	80.5 (1.1)	85.6 (0.8)	**93.0 (0.6)**
Corel, Elephants ♥	87.0 (0.7)	95.7 (0.4)	96.6 (0.4)	**98.2 (0.2)**	97.0 (0.2)	89.3 (0.8)	88.4 (0.6)	97.4 (0.2)
Corel, Fashion ♥	90.9 (0.9)	99.0 (0.1)	98.3 (0.4)	99.0 (0.1)	99.1 (0.1)	95.0 (0.5)	92.8 (0.7)	**99.3 (0.1)**
Corel, Flowers ♥	83.3 (1.1)	94.3 (0.6)	93.2 (0.6)	94.7 (0.6)	95.3 (0.6)	84.4 (1.0)	89.0 (0.8)	**97.2 (0.3)**
Corel, Food ♥	93.4 (0.6)	99.4 (0.1)	99.3 (0.1)	**99.8 (0.1)**	99.6 (0.0)	98.5 (0.2)	98.4 (0.3)	99.6 (0.1)
Corel, Historical ♥	94.2 (0.6)	99.3 (0.1)	98.4 (0.2)	**99.8 (0.0)**	99.4 (0.1)	97.8 (0.3)	98.6 (0.2)	99.6 (0.1)
Corel, Horses ♥	74.6 (1.1)	89.6 (0.7)	86.1 (0.8)	**92.0 (0.6)**	91.7 (0.5)	80.9 (1.2)	81.5 (1.2)	90.5 (0.6)

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

Table 3.3. – AUC and standard error (× 100) results of various MIL methods (cont.).

Dataset	**Algorithm AUC (%)**							
			MInD				Bag encoding	
	Citation-kNN	MILES	D_{maxmin}	$D_{meanmin}$	D_{minmin}	CCE	k-means two class	Path
Corel, Lizards ♥	89.7 (0.7)	97.0 (0.4)	96.0 (0.6)	**98.0 (0.3)**	97.1 (0.4)	93.8 (0.5)	93.4 (0.6)	97.7 (0.3)
Corel, Mountains ♥	97.9 (0.4)	99.9 (0.0)	99.8 (0.0)	**100 (0.0)**	99.9 (0.0)	98.7 (0.1)	99.3 (0.1)	**100 (0.0)**
Corel, Skiing ♥	77.0 (1.2)	94.7 (0.4)	95.6 (0.4)	96.0 (0.3)	95.3 (0.5)	87.5 (0.8)	87.3 (0.8)	**97.0 (0.2)**
Corel, Sunset ♥	70.6 (1.2)	76.3 (1.2)	80.3 (1.0)	83.7 (1.0)	75.1 (1.2)	72.8 (1.3)	75.1 (0.9)	**86.3 (1.0)**
Corel, Waterfalls ♥	86.5 (1.0)	94.5 (0.5)	96.2 (0.4)	97.5 (0.2)	96.6 (0.3)	83.6 (0.9)	89.6 (0.7)	**97.6 (0.2)**
UCSB Breast Cancer ♥	70.6 (3.2)	83.3 (2.6)	72.5 (2.8)	83.1 (2.7)	79.1 (2.7)	64.4 (2.9)	84.8 (2.4)	**88.0 (2.2)**
Newsgroups 1, alt.atheism ♠	80.3 (2.1)	41.6 (2.3)	90.5 (1.4)	**94.1 (1.0)**	50.0 (0.0)	79.2 (1.9)	65.4 (2.0)	86.1 (1.7)
N.g. 2, comp.graphics ♠	63.8 (2.8)	52.8 (2.2)	89.5 (1.5)	**89.8 (1.6)**	55.4 (1.8)	66.0 (1.9)	61.4 (1.9)	67.1 (2.2)
N.g. 3, comp.os.ms-windows.misc ♠	58.4 (2.8)	47.9 (2.7)	**81.8 (2.1)**	81.0 (2.1)	50.0 (0.0)	62.2 (2.7)	66.0 (1.8)	74.6 (1.9)
N.g. 4, comp.sys.ibm.pc.hardware ♠	63.6 (2.6)	68.2 (2.4)	82.2 (2.4)	**85.7 (2.2)**	47.9 (2.0)	67.8 (2.5)	66.3 (2.1)	75.2 (2.2)
N.g. 5, comp.sys.mac.hardware ♠	58.5 (2.4)	58.9 (2.8)	**85.3 (1.6)**	85.2 (1.6)	55.9 (2.6)	60.9 (2.2)	63.8 (1.6)	74.0 (1.7)
N.g. 6, comp.windows.x ♠	73.2 (2.1)	61.0 (2.4)	86.5 (1.9)	**89.0 (1.7)**	57.2 (2.2)	74.2 (1.9)	65.0 (1.9)	86.2 (1.9)
N.g. 7, misc.forsale ♠	63.3 (2.1)	51.1 (2.4)	75.2 (2.3)	**79.0 (2.0)**	54.7 (2.3)	64.8 (2.5)	65.4 (1.8)	77.9 (1.9)
N.g. 8, rec.autos ♠	57.0 (2.5)	49.8 (2.5)	84.0 (1.9)	**87.0 (1.7)**	46.0 (1.1)	69.2 (1.8)	62.3 (1.7)	80.3 (1.7)
N.g. 9, rec.motorcycles ♠	81.8 (2.0)	52.2 (2.6)	34.8 (4.6)	32.6 (3.2)	50.0 (0.0)	80.6 (2.0)	61.6 (1.8)	**89.1 (1.6)**
N.g. 10, rec.sport.baseball ♠	82.1 (1.8)	51.0 (2.7)	**91.8 (1.2)**	91.4 (1.4)	47.6 (1.9)	74.7 (2.3)	65.2 (1.8)	88.0 (1.4)
N.g. 11, rec.sport.hockey ♠	71.4 (2.0)	37.4 (2.2)	**96.8 (0.7)**	95.8 (0.8)	46.0 (2.4)	79.4 (1.8)	63.4 (1.7)	93.8 (1.0)
N.g. 12, sci.crypt ♠	80.6 (2.0)	46.2 (2.7)	**86.8 (2.1)**	84.0 (1.9)	46.6 (2.3)	75.6 (1.8)	64.2 (2.0)	80.2 (2.2)
N.g. 13, sci.electronics ♠	62.1 (2.0)	48.7 (2.2)	93.2 (1.2)	**94.6 (1.0)**	50.0 (0.0)	52.8 (0.8)	63.2 (2.0)	67.6 (2.4)
N.g. 14, sci.med ♠	78.0 (2.1)	49.8 (2.4)	92.2 (1.2)	**94.2 (0.8)**	46.5 (2.4)	77.7 (1.7)	64.1 (1.7)	88.9 (1.3)
N.g. 15, sci.space ♠	76.3 (2.1)	43.6 (2.5)	87.4 (1.6)	**90.5 (1.4)**	51.7 (2.1)	79.0 (2.1)	64.4 (1.9)	89.8 (1.7)
N.g. 16, soc.religion.christian ♠	77.3 (2.2)	46.1 (2.4)	87.9 (1.7)	**89.8 (1.4)**	50.9 (2.5)	78.7 (1.8)	62.7 (1.9)	82.2 (1.6)
N.g. 17, talk.politics.guns ♠	69.2 (2.1)	49.8 (2.5)	81.8 (1.8)	**87.4 (1.5)**	53.1 (2.7)	76.6 (1.8)	62.0 (1.6)	76.1 (1.8)
N.g. 18, talk.politics.mideast ♠	84.5 (1.8)	54.6 (3.0)	83.3 (2.0)	87.4 (1.7)	46.3 (1.6)	82.5 (1.4)	66.6 (1.8)	**88.2 (1.6)**

Table 3.3. – AUC and standard error (× 100) results of various MIL methods (cont.).

Dataset	**Algorithm AUC (%)**							
			MInD				Bag encoding	
	Citation-kNN	MILES	D_{maxmin}	$D_{meanmin}$	D_{minmin}	CCE	k-means two class	Path
N.g. 19, talk.politics.misc ♠	76.6 (1.7)	55.0 (2.3)	78.5 (2.1)	80.2 (1.9)	56.1 (2.7)	**83.1 (1.8)**	63.8 (2.0)	74.7 (2.1)
N.g. 20, talk.religion.misc ♠	64.6 (2.0)	56.0 (2.7)	**83.9 (2.0)**	83.4 (2.2)	44.3 (2.1)	77.0 (2.0)	64.2 (1.8)	75.4 (2.1)
Web 1 ♠	61.4 (3.5)	73.2 (3.0)	41.1 (3.0)	63.4 (4.2)	78.8 (3.3)	78.8 (2.5)	**83.4 (2.1)**	75.0 (2.6)
Web 2 ♠	44.5 (3.3)	54.4 (3.9)	50.1 (3.8)	47.4 (4.2)	52.1 (3.2)	47.3 (3.1)	**69.3 (2.2)**	46.8 (3.9)
Web 3 ♠	64.9 (3.3)	67.1 (4.4)	50.1 (3.4)	70.8 (4.6)	60.8 (4.5)	60.3 (4.2)	**76.7 (2.3)**	69.9 (4.3)
Web 4 ♠	70.7 (3.1)	74.3 (3.5)	54.8 (3.8)	79.9 (3.6)	62.9 (3.6)	83.4 (2.3)	**84.6 (2.3)**	77.6 (3.3)
Web 5 ♠	50.6 (3.3)	74.3 (3.4)	50.7 (3.5)	71.1 (3.7)	**79.4 (2.7)**	61.2 (4.4)	77.5 (2.4)	73.3 (4.0)
Web 6 ♠	52.0 (3.8)	55.0 (3.4)	49.4 (4.3)	52.5 (4.2)	54.3 (3.9)	62.1 (3.5)	**72.7 (2.2)**	66.5 (3.9)
Web 7 ♠	67.5 (2.5)	62.5 (2.6)	60.0 (2.8)	69.0 (2.8)	67.0 (2.8)	59.2 (2.9)	68.8 (1.9)	**72.9 (2.4)**
Web 8 ♠	49.8 (3.0)	51.1 (3.1)	57.9 (3.5)	40.9 (2.6)	36.6 (2.4)	57.5 (3.0)	**65.0 (2.0)**	52.8 (3.2)
Web 9 ♠	59.9 (3.0)	68.7 (2.3)	49.7 (2.3)	73.5 (2.7)	68.7 (2.5)	63.1 (3.9)	**74.1 (2.0)**	45.9 (3.2)
Birds, Brown creeper ♦	88.3 (0.8)	97.4 (0.3)	72.9 (1.0)	89.9 (0.5)	92.7 (0.5)	94.5 (0.4)	99.2 (0.1)	**99.4 (0.1)**
Birds, Chestnut-backed chickadee ♦	80.2 (1.4)	80.1 (1.3)	83.0 (0.9)	85.3 (0.8)	74.9 (1.7)	82.7 (1.1)	91.0 (0.6)	**94.3 (0.4)**
Birds, Dark-eyed junco ♦	59.4 (1.9)	89.1 (1.2)	69.5 (2.0)	85.6 (1.3)	87.0 (1.0)	66.7 (2.9)	94.7 (0.6)	**97.8 (0.3)**
Birds, Hammonds flycatcher ♦	88.4 (1.2)	93.9 (0.8)	71.8 (1.7)	94.4 (0.7)	88.3 (1.1)	99.2 (0.1)	99.9 (0.0)	**100 (0.0)**
Birds, Hermit thrush ♦	53.5 (2.1)	68.2 (3.0)	57.0 (3.9)	57.8 (4.4)	89.2 (2.2)	48.8 (3.6)	83.6 (1.9)	**92.1 (1.4)**
Birds, Hermit warbler ♦	69.7 (1.9)	90.4 (1.3)	73.5 (1.9)	78.1 (1.5)	92.6 (0.7)	81.8 (1.1)	97.9 (0.2)	**98.6 (0.2)**
Birds, Olive-sided flycatcher ♦	79.4 (1.1)	92.0 (0.5)	85.3 (0.8)	89.6 (0.6)	92.4 (0.5)	87.8 (0.8)	96.3 (0.3)	**97.7 (0.2)**
Birds, Pacificslope flycatcher ♦	69.9 (1.0)	84.8 (0.8)	72.3 (1.1)	75.4 (1.0)	76.9 (0.7)	83.6 (0.9)	95.8 (0.4)	**96.8 (0.3)**
Birds, Red-breasted nuthatch ♦	77.1 (1.2)	90.7 (0.7)	80.3 (0.9)	87.6 (0.7)	87.7 (0.6)	87.2 (0.9)	98.2 (0.3)	**99.3 (0.1)**
Birds, Swainsons thrush ♦	68.7 (2.1)	80.4 (1.7)	78.2 (1.6)	76.7 (1.7)	88.4 (1.0)	82.6 (1.0)	98.1 (0.3)	**99.8 (0.0)**
Birds, Varied thrush ♦	78.2 (1.3)	95.1 (0.6)	75.1 (1.5)	84.0 (1.2)	94.0 (0.6)	86.9 (0.7)	**100 (0.0)**	**100 (0.0)**
Birds, Western tanager ♦	75.5 (2.4)	89.4 (1.6)	47.5 (3.9)	84.9 (1.8)	82.4 (1.9)	87.8 (0.9)	98.2 (0.4)	**99.5 (0.1)**
Birds, Winter wren ♦	90.7 (0.6)	94.6 (0.4)	94.4 (0.5)	93.1 (0.7)	85.4 (0.9)	96.5 (0.3)	99.0 (0.2)	**99.7 (0.0)**

Table 3.4. Accuracy and standard error (× 100) results of various MIL methods.

Dataset	**Algorithm accuracy (%)**								
			MInD					Bag encoding	
	Citation-kNN	MILES	D_{maxmin}	$D_{meanmin}$	D_{minmin}	CCE	miFV	k-means two class	Path
Musk 1 ♣	84.4 (1.6)	79.6 (2.0)	85.4 (1.8)	84.1 (1.8)	82.7 (1.5)	81.4 (1.9)	85.2 (1.6)	81.8 (1.5)	**87.1 (1.6)**
Musk 2 ♣	79.4 (1.6)	86.3 (1.1)	88.2 (1.3)	**92.3 (1.1)**	86.3 (1.3)	76.4 (1.6)	87.7 (1.6)	72.9 (1.9)	85.6 (1.5)
Mutagenesis 1 ♣	85.1 (1.2)	79.8 (1.2)	76.3 (1.3)	77.4 (1.3)	70.7 (0.9)	82.3 (1.2)	82.6 (1.2)	87.5 (1.0)	**89.3 (0.9)**
Mutagenesis 2 ♣	74.9 (2.6)	70.9 (2.7)	76.2 (1.9)	70.4 (2.0)	68.3 (1.7)	73.9 (2.1)	76.6 (2.2)	79.4 (2.6)	**85.6 (2.6)**
Protein ♣	83.8 (0.8)	**94.9 (0.6)**	87.1 (0.3)	87.1 (0.3)	86.8 (0.7)	86.9 (0.4)	85.4 (0.8)	87.9 (0.5)	88.5 (0.5)
Elephant ♥	83.3 (1.1)	82.7 (1.0)	80.7 (1.2)	**86.2 (1.0)**	84.8 (1.0)	78.1 (1.2)	82.9 (1.1)	80.2 (1.1)	85.8 (1.0)
Fox ♥	55.6 (1.6)	64.9 (1.4)	47.5 (0.6)	57.8 (1.2)	63.4 (1.4)	60.6 (1.3)	62.3 (1.3)	60.8 (1.4)	**65.2 (1.2)**
Tiger ♥	73.7 (1.4)	79.2 (1.2)	69.3 (1.2)	77.7 (1.2)	77.5 (1.4)	76.0 (1.1)	80.4 (1.3)	77.8 (1.3)	**85.0 (1.1)**
Corel, African ♥	95.3 (0.2)	96.6 (0.2)	96.6 (0.1)	**97.3 (0.1)**	**97.3 (0.1)**	95.1 (0.0)	96.4 (0.1)	95.2 (0.1)	96.9 (0.1)
Corel, Antique ♥	94.6 (0.1)	93.8 (0.2)	95.0 (0.1)	**95.4 (0.1)**	95.2 (0.1)	95.0 (0.0)	95.0 (0.1)	94.9 (0.1)	95.2 (0.1)
Corel, Battleships ♥	96.7 (0.2)	96.1 (0.2)	97.1 (0.1)	97.1 (0.1)	**97.2 (0.1)**	95.9 (0.1)	96.2 (0.1)	95.4 (0.1)	97.0 (0.1)
Corel, Beach ♥	97.2 (0.1)	98.1 (0.1)	95.4 (0.1)	97.8 (0.1)	98.3 (0.1)	95.1 (0.0)	97.7 (0.1)	97.4 (0.1)	**98.4 (0.1)**
Corel, Buses ♥	97.0 (0.1)	**98.1 (0.1)**	97.0 (0.1)	97.7 (0.1)	97.9 (0.1)	95.0 (0.0)	97.2 (0.2)	96.4 (0.1)	97.4 (0.1)
Corel, Cars ♥	95.8 (0.2)	95.4 (0.2)	96.3 (0.1)	**97.3 (0.1)**	96.5 (0.2)	95.6 (0.1)	96.5 (0.1)	95.6 (0.1)	96.8 (0.1)
Corel, Desserts ♥	96.4 (0.2)	96.5 (0.1)	96.8 (0.1)	**97.7 (0.1)**	97.3 (0.1)	95.1 (0.0)	97.4 (0.1)	95.9 (0.1)	96.7 (0.1)
Corel, Dinosaurs ♥	96.3 (0.1)	97.5 (0.1)	96.8 (0.1)	**97.9 (0.1)**	97.7 (0.1)	95.0 (0.0)	96.6 (0.1)	95.3 (0.1)	96.2 (0.1)
Corel, Dogs ♥	95.5 (0.1)	94.8 (0.2)	96.1 (0.1)	**96.3 (0.1)**	96.0 (0.1)	95.1 (0.0)	95.6 (0.1)	95.2 (0.1)	95.6 (0.1)
Corel, Elephants ♥	95.4 (0.1)	96.0 (0.1)	96.3 (0.1)	**96.9 (0.1)**	96.3 (0.1)	95.3 (0.1)	96.5 (0.1)	95.3 (0.1)	96.6 (0.1)
Corel, Fashion ♥	97.3 (0.1)	98.3 (0.1)	98.3 (0.1)	**98.6 (0.1)**	98.5 (0.1)	95.4 (0.1)	98.1 (0.1)	96.0 (0.1)	**98.6 (0.1)**
Corel, Flowers ♥	96.1 (0.2)	96.1 (0.2)	96.1 (0.1)	97.0 (0.1)	**97.2 (0.1)**	95.2 (0.1)	96.4 (0.2)	95.5 (0.1)	96.3 (0.1)
Corel, Food ♥	97.9 (0.2)	98.6 (0.1)	98.5 (0.1)	**98.9 (0.1)**	98.6 (0.1)	96.7 (0.1)	97.9 (0.2)	97.8 (0.1)	98.3 (0.1)
Corel, Historical ♥	98.2 (0.1)	98.7 (0.1)	97.6 (0.1)	**98.9 (0.1)**	**98.9 (0.1)**	96.9 (0.1)	97.8 (0.1)	97.5 (0.1)	98.1 (0.1)
Corel, Horses ♥	95.1 (0.2)	94.6 (0.2)	95.5 (0.1)	96.2 (0.1)	**96.3 (0.1)**	95.0 (0.0)	95.9 (0.1)	95.2 (0.1)	95.8 (0.1)

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

Table 3.4. – Accuracy and standard error (× 100) results of various MIL methods (cont.).

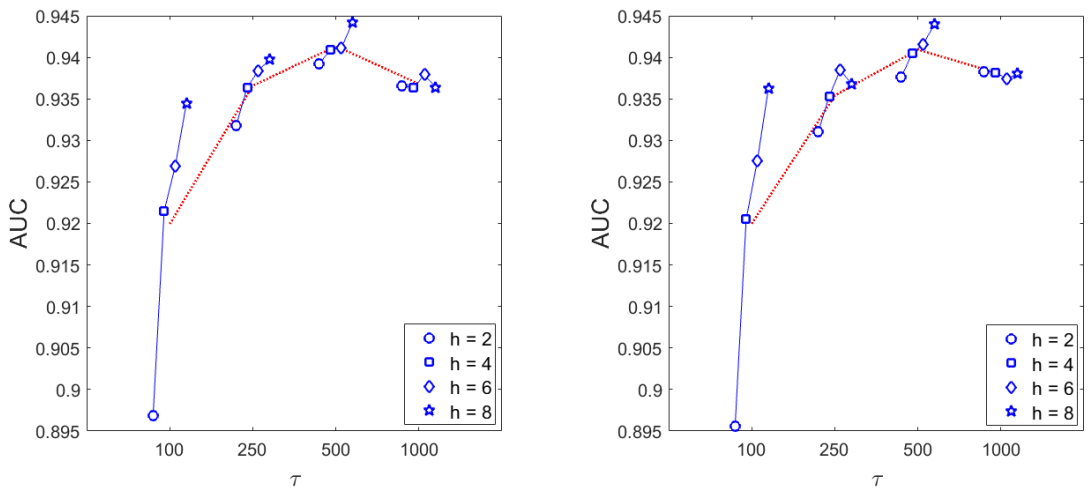
Dataset	**Algorithm accuracy (%)**								
			MInD					Bag encoding	
	Citation-kNN	MILES	D_{maxmin}	$D_{meanmin}$	D_{minmin}	CCE	miFV	k-means two class	Path
Corel, Lizards ♥	97.3 (0.1)	**97.8 (0.1)**	97.4 (0.1)	**97.8 (0.1)**	97.4 (0.1)	96.1 (0.1)	97.0 (0.1)	96.5 (0.1)	97.6 (0.1)
Corel, Mountains ♥	98.7 (0.1)	99.3 (0.1)	98.0 (0.2)	99.5 (0.1)	99.5 (0.1)	97.2 (0.2)	99.5 (0.1)	98.1 (0.1)	**99.8 (0.0)**
Corel, Skiing ♥	95.7 (0.1)	96.4 (0.1)	**96.7 (0.1)**	96.4 (0.1)	96.4 (0.1)	95.0 (0.0)	96.3 (0.1)	95.3 (0.1)	96.1 (0.1)
Corel, Sunset ♥	95.0 (0.1)	92.4 (0.2)	95.0 (0.1)	**95.2 (0.1)**	**95.2 (0.1)**	95.1 (0.0)	95.1 (0.1)	95.0 (0.0)	95.1 (0.0)
Corel, Waterfalls ♥	96.3 (0.1)	96.0 (0.2)	**97.1 (0.1)**	97.0 (0.2)	**97.1 (0.1)**	95.1 (0.0)	95.8 (0.1)	95.4 (0.1)	96.6 (0.1)
UCSB Breast Cancer ♥	68.3 (2.7)	75.8 (2.2)	62.8 (1.8)	72.2 (2.3)	70.2 (2.5)	55.3 (1.0)	79.6 (2.4)	76.3 (2.6)	**80.5 (2.3)**
Newsgroups 1, alt.atheism ♠	72.4 (1.9)	41.8 (1.8)	79.2 (1.7)	**85.6 (1.5)**	50.0 (0.0)	68.4 (1.7)	81.2 (1.4)	54.0 (2.3)	79.2 (1.8)
N.g. 2, comp.graphics ♠	60.0 (2.5)	52.2 (2.0)	76.0 (1.3)	**79.0 (1.4)**	53.6 (1.3)	54.0 (0.8)	53.4 (1.2)	51.2 (2.0)	63.6 (2.0)
N.g. 3, comp.os.ms-windows.misc ♠	57.2 (2.2)	46.6 (2.2)	53.4 (0.9)	54.0 (0.9)	50.0 (0.0)	58.0 (1.7)	55.2 (1.7)	50.6 (2.3)	**68.0 (1.5)**
N.g. 4, comp.sys.ibm.pc.hardware ♠	60.6 (2.1)	62.4 (2.5)	**78.8 (1.6)**	75.4 (1.6)	46.2 (1.5)	59.4 (1.1)	65.0 (2.1)	59.4 (2.1)	67.2 (2.1)
N.g. 5, comp.sys.mac.hardware ♠	55.0 (2.0)	56.4 (2.4)	77.2 (1.4)	**79.6 (1.2)**	56.4 (1.8)	50.8 (0.8)	59.6 (1.6)	54.4 (2.1)	65.6 (1.6)
N.g. 6, comp.windows.x ♠	66.4 (1.6)	56.8 (2.0)	58.4 (1.3)	66.8 (1.5)	51.0 (1.6)	64.4 (1.3)	75.0 (1.9)	55.2 (2.2)	**78.0 (1.8)**
N.g. 7, misc.forsale ♠	63.0 (1.7)	53.0 (2.3)	53.2 (1.5)	53.2 (1.5)	52.0 (1.9)	57.4 (1.3)	60.0 (1.8)	48.2 (2.1)	**68.6 (2.0)**
N.g. 8, rec.autos ♠	57.2 (1.9)	50.2 (1.7)	75.6 (1.6)	**77.0 (1.6)**	46.2 (1.0)	57.4 (1.1)	63.0 (1.9)	52.8 (1.9)	74.2 (1.3)
N.g. 9, rec.motorcycles ♠	74.6 (2.0)	51.8 (1.9)	51.0 (0.6)	51.0 (0.5)	50.0 (0.0)	67.2 (1.6)	74.0 (2.1)	51.4 (2.0)	**80.6 (1.8)**
N.g. 10, rec.sport.baseball ♠	68.6 (2.0)	50.6 (1.9)	**81.4 (1.4)**	80.0 (1.6)	49.6 (1.1)	66.0 (1.4)	77.2 (1.5)	53.0 (2.4)	75.0 (1.7)
N.g. 11, rec.sport.hockey ♠	67.2 (1.8)	42.0 (2.1)	84.6 (1.4)	**85.8 (1.5)**	44.6 (1.8)	67.4 (1.3)	77.0 (1.5)	50.4 (1.9)	85.2 (1.5)
N.g. 12, sci.crypt ♠	73.2 (1.6)	47.2 (2.0)	**79.6 (2.1)**	62.4 (1.6)	50.0 (1.8)	67.4 (1.5)	77.6 (2.1)	50.2 (2.2)	73.2 (2.3)
N.g. 13, sci.electronics ♠	64.2 (1.9)	46.6 (1.7)	88.4 (1.3)	**89.0 (1.3)**	50.0 (0.0)	50.0 (0.0)	57.4 (1.6)	55.2 (2.1)	55.4 (1.2)
N.g. 14, sci.med ♠	69.2 (2.1)	52.0 (1.9)	75.0 (1.4)	80.0 (1.4)	46.6 (1.7)	65.2 (1.4)	73.2 (1.5)	47.8 (2.1)	**84.0 (1.5)**
N.g. 15, sci.space ♠	69.0 (1.9)	45.2 (2.1)	79.0 (1.6)	78.4 (1.5)	53.0 (1.9)	71.4 (1.6)	77.4 (1.9)	51.4 (2.2)	**81.0 (1.7)**
N.g. 16, soc.religion.christian ♠	72.2 (1.6)	45.0 (1.9)	80.8 (1.6)	**83.8 (1.4)**	49.2 (2.0)	70.0 (1.5)	76.0 (1.4)	52.0 (2.0)	76.8 (1.5)
N.g. 17, talk.politics.guns ♠	64.6 (2.0)	48.4 (2.1)	**80.0 (1.4)**	77.8 (1.6)	52.6 (1.6)	68.4 (1.7)	73.4 (1.7)	51.0 (2.4)	72.4 (1.8)
N.g. 18, talk.politics.mideast ♠	78.0 (1.9)	53.8 (2.1)	74.6 (1.4)	78.2 (1.3)	48.0 (1.1)	68.6 (1.7)	79.0 (1.5)	54.0 (2.3)	**82.6 (1.6)**

Table 3.4. – Accuracy and standard error (× 100) results of various MIL methods (cont.).

Dataset	**Algorithm accuracy (%)**								
			MInD					Bag encoding	
	Citation-kNN	MILES	D_{maxmin}	$D_{meanmin}$	D_{minmin}	CCE	miFV	k-means two class	Path
N.g. 19, talk.politics.misc ♠	**73.0 (1.5)**	51.8 (2.1)	68.2 (1.6)	68.6 (1.7)	54.2 (1.6)	72.4 (1.6)	60.0 (2.2)	46.6 (2.3)	70.4 (1.9)
N.g. 20, talk.religion.misc ♠	60.2 (2.1)	52.8 (2.3)	56.6 (1.2)	62.4 (1.3)	44.8 (1.1)	65.4 (1.6)	69.2 (2.2)	54.6 (2.0)	**70.6 (1.9)**
Web 1 ♠	71.4 (1.8)	75.5 (1.5)	71.4 (0.7)	69.9 (0.9)	73.1 (1.0)	72.4 (1.8)	74.9 (1.3)	**77.7 (2.0)**	73.1 (0.9)
Web 2 ♠	66.4 (1.5)	75.0 (1.1)	75.5 (0.7)	75.3 (0.9)	74.9 (0.8)	75.8 (0.7)	73.4 (1.2)	71.0 (1.4)	**76.1 (0.7)**
Web 3 ♠	77.5 (1.4)	**85.1 (1.4)**	81.6 (0.8)	81.6 (1.0)	81.0 (0.8)	80.6 (1.1)	82.1 (1.1)	82.9 (1.0)	81.9 (0.8)
Web 4 ♠	74.6 (2.2)	76.5 (1.8)	72.9 (0.4)	75.8 (0.9)	74.4 (1.3)	81.2 (1.6)	**82.5 (1.7)**	80.9 (1.4)	78.6 (1.4)
Web 5 ♠	77.1 (1.4)	**82.1 (1.3)**	80.9 (1.0)	79.7 (1.1)	80.8 (0.9)	81.6 (0.8)	79.8 (1.1)	78.9 (1.1)	81.1 (0.8)
Web 6 ♠	66.2 (2.2)	**81.5 (1.0)**	78.9 (0.8)	78.9 (0.8)	77.9 (1.0)	78.9 (0.8)	74.7 (1.3)	72.2 (1.8)	78.6 (0.9)
Web 7 ♠	65.0 (2.1)	54.3 (1.9)	51.8 (1.5)	63.1 (2.3)	62.1 (2.4)	55.5 (2.8)	**65.1 (2.8)**	56.3 (2.6)	64.6 (2.1)
Web 8 ♠	48.0 (2.1)	50.1 (2.3)	**55.1 (1.0)**	50.3 (1.4)	51.9 (1.0)	53.4 (2.1)	53.3 (2.1)	51.0 (2.6)	50.8 (2.7)
Web 9 ♠	61.0 (2.4)	61.8 (2.1)	46.3 (1.4)	**69.3 (2.2)**	62.7 (2.2)	64.5 (2.6)	65.9 (2.0)	68.9 (2.2)	47.9 (2.7)
Birds, Brown creeper ♦	84.6 (0.7)	92.4 (0.5)	63.7 (0.3)	81.8 (0.7)	85.5 (0.6)	86.5 (0.7)	95.1 (0.5)	95.8 (0.4)	**96.5 (0.3)**
Birds, Chestnut-backed chickadee ♦	85.1 (0.7)	80.6 (0.8)	88.2 (0.4)	88.4 (0.5)	88.7 (0.4)	86.9 (0.5)	**91.1 (0.4)**	89.2 (0.5)	90.1 (0.4)
Birds, Dark-eyed junco ♦	94.5 (0.3)	95.8 (0.3)	95.8 (0.2)	95.8 (0.2)	95.5 (0.2)	**96.4 (0.0)**	95.2 (0.3)	**96.4 (0.0)**	**96.4 (0.0)**
Birds, Hammonds flycatcher ♦	88.1 (0.8)	91.1 (0.8)	89.4 (0.4)	90.8 (0.4)	90.1 (0.4)	90.2 (0.5)	92.6 (0.4)	99.5 (0.2)	**99.6 (0.1)**
Birds, Hermit thrush ♦	96.2 (0.2)	96.7 (0.2)	**97.3 (0.1)**	97.2 (0.1)	**97.3 (0.1)**	**97.3 (0.1)**	97.0 (0.1)	**97.3 (0.1)**	**97.3 (0.1)**
Birds, Hermit warbler ♦	87.4 (0.6)	90.4 (0.5)	91.5 (0.3)	91.6 (0.3)	92.1 (0.3)	88.4 (0.1)	93.8 (0.4)	94.1 (0.3)	**95.5 (0.4)**
Birds, Olive-sided flycatcher ♦	84.0 (0.6)	88.7 (0.5)	83.2 (0.2)	84.3 (0.2)	86.6 (0.4)	84.9 (0.3)	91.3 (0.5)	89.6 (0.5)	**93.4 (0.5)**
Birds, Pacificslope flycatcher ♦	72.6 (0.7)	79.6 (0.8)	77.4 (0.5)	77.6 (0.5)	75.5 (0.5)	72.6 (0.3)	**95.4 (0.4)**	91.0 (0.5)	91.8 (0.5)
Birds, Red-breasted nuthatch ♦	85.7 (0.5)	88.4 (0.6)	85.0 (0.2)	85.0 (0.2)	86.1 (0.3)	88.1 (0.3)	90.3 (0.5)	94.7 (0.3)	**96.8 (0.3)**
Birds, Swainsons thrush ♦	86.1 (0.7)	86.7 (0.7)	91.5 (0.3)	91.4 (0.3)	91.7 (0.4)	87.2 (0.2)	93.4 (0.4)	95.4 (0.3)	**96.5 (0.3)**
Birds, Varied thrush ♦	84.6 (0.5)	92.7 (0.6)	88.0 (0.3)	88.1 (0.3)	89.2 (0.4)	83.7 (0.1)	91.4 (0.4)	**99.6 (0.1)**	99.5 (0.1)
Birds, Western tanager ♦	92.7 (0.4)	93.5 (0.5)	94.9 (0.3)	94.7 (0.3)	94.3 (0.3)	91.6 (0.1)	**97.9 (0.2)**	95.8 (0.3)	96.8 (0.3)
Birds, Winter wren ♦	90.3 (0.5)	91.2 (0.4)	92.4 (0.4)	93.4 (0.4)	79.5 (0.2)	82.3 (0.3)	**97.6 (0.3)**	95.8 (0.3)	97.1 (0.3)

Table 3.5. Performance summary of path-encoding on 20 classes of PASCAL VOC 2017 dataset.

Class	AUC	Accuracy	AP	Class	AUC	Accuracy	AP
aero	99.3	98.2	95.0	table	98.0	96.4	67.7
bike	97.9	97.3	91.5	dog	96.4	93.0	80.3
bird	97.7	95.5	88.1	horse	98.5	97.7	93.2
boat	98.1	98.3	89.8	mbike	98.1	97.1	87.2
bottle	92.1	95.9	51.4	person	95.4	88.9	94.9
bus	98.5	97.7	87.5	plant	94.9	95.9	65.6
car	97.3	93.4	92.4	sheep	98.6	98.7	83.7
cat	96.7	96.7	86.0	sofa	96.3	95.9	63.7
chair	92.5	92.7	59.9	train	99.4	98.1	96.0
cow	98.1	98.2	84.5	tv	96.5	96.5	71.1



(a) Terminal node-encoding (b) Path-encoding

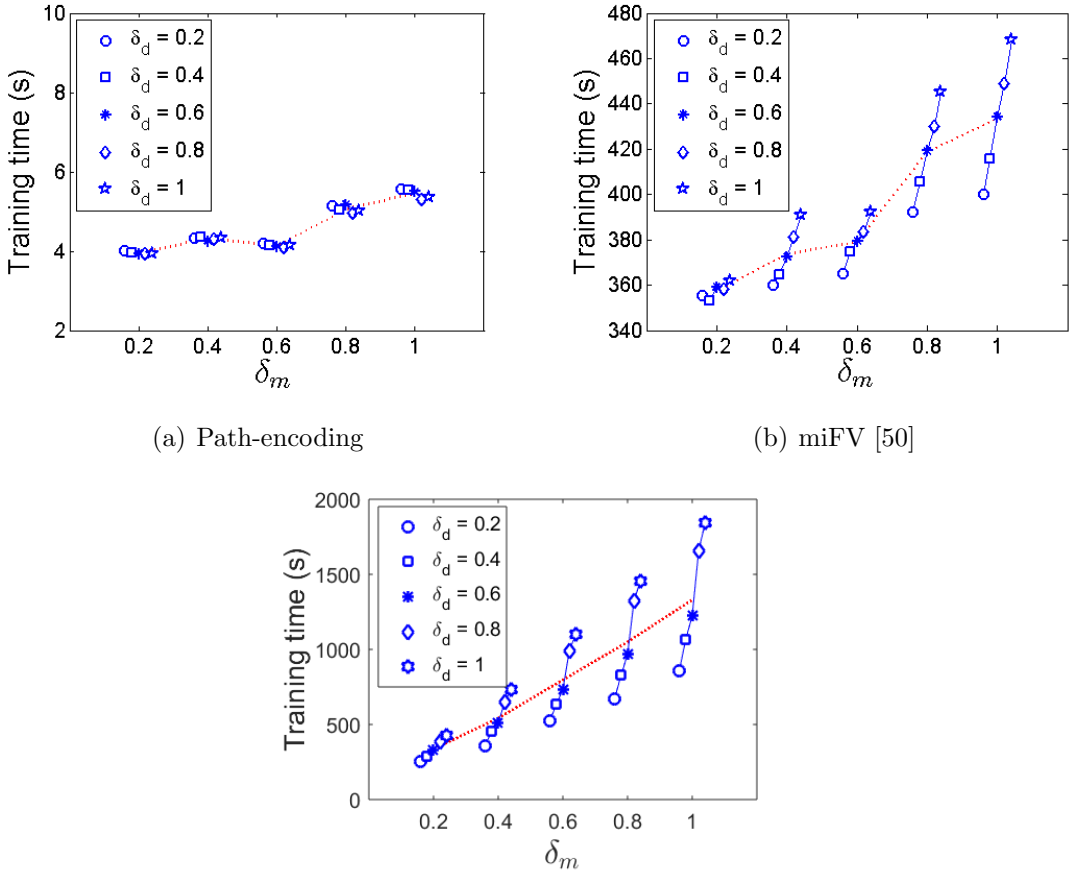
Figure 3.10. Curves of the AUC values of RT-encodings on Elephant dataset obtained for different number of trees and number of maximum tree depths.

3.4.4. Computational Time Analysis

Our experiments use a Windows 10 system with 8 GB RAM, dual core CPU (i5-3470, 3.2 GHz). Although the CPU can handle four threads in parallel, only a single thread is used. Elephant and PASCAL VOC 2007 datasets are used to demonstrate the effect of the number of training bags (m) and the number of features (d) on the computation times empirically. We select the two best performing methods, $\mathrm{D}_{\mathrm{meanmin}}$ [26]

and miFV [50] for comparison. The same set of parameters in Section 3.4.1 is used to evaluate the performance of the two approaches. A recent method, RSIS [55] is also included for comparison by using the following setting. As advised by the authors, four parameter values of RSIS are determined in each experiment by an inner cross validation loop. The levels for number of clusters are $\{5, 10, 15, 20\}$, whereas the temperature for instance selection is chosen from the set $\{0.00001, 0.001, 10\}$. In addition, cost parameter of the SVM classifier is selected from the set $\{1, 10, 100\}$, together with the width parameter γ of the Gaussian kernel from $\{0.000001, 0.0001, 0.1\}$. Because of the computational requirements of miFV and RSIS, comparisons to these methods are performed only on Elephant dataset. In order to discuss potential problems with similarity-based approaches, PASCAL VOC 2007 dataset is used for comparisons to $\mathrm{D}_{\mathrm{meanmin}}$. For each dataset, we randomly selected proportions of the number of bags (δ_m) and the number of features (δ_d) from the set $\{0.2, 0.4, 0.6, 0.8, 1\}$. Here, 10 replications are conducted for each setting combination.

Reported training times for $\mathrm{D}_{\mathrm{meanmin}}$ and path-encoding are the time required to learn the representation for training bags. For $\mathrm{D}_{\mathrm{meanmin}}$, it involves the distance calculations between the training instances and aggregation of these distances to obtain a bag representation. For miFV, it involves the clustering of the training instances and obtaining the Fisher vector representation. For path-encoding, it includes the parameter tuning, training of random tree ensemble, traversal of the trees and aggregation of instance representations. Training time of RSIS comprises of clustering the data in random subspaces to compute instance selection probabilities, and classifier ensemble generation on the probabilistically formed training subsets. We also measure the time for parameter selection in miFV and RSIS, which is included in the training time. Testing time for $\mathrm{D}_{\mathrm{meanmin}}$ contains the distance calculation between test and train instances and aggregation of these distances to obtain a bag representation. miFV maps the test instances to the new space and obtains bag representation in testing. In RSIS, individual instances of a test bag are fed into the trained classifiers in the ensemble and instance positivity scores are calculated to decide bag labels. Path-encoding has the same steps as in the training except the tree training phase.



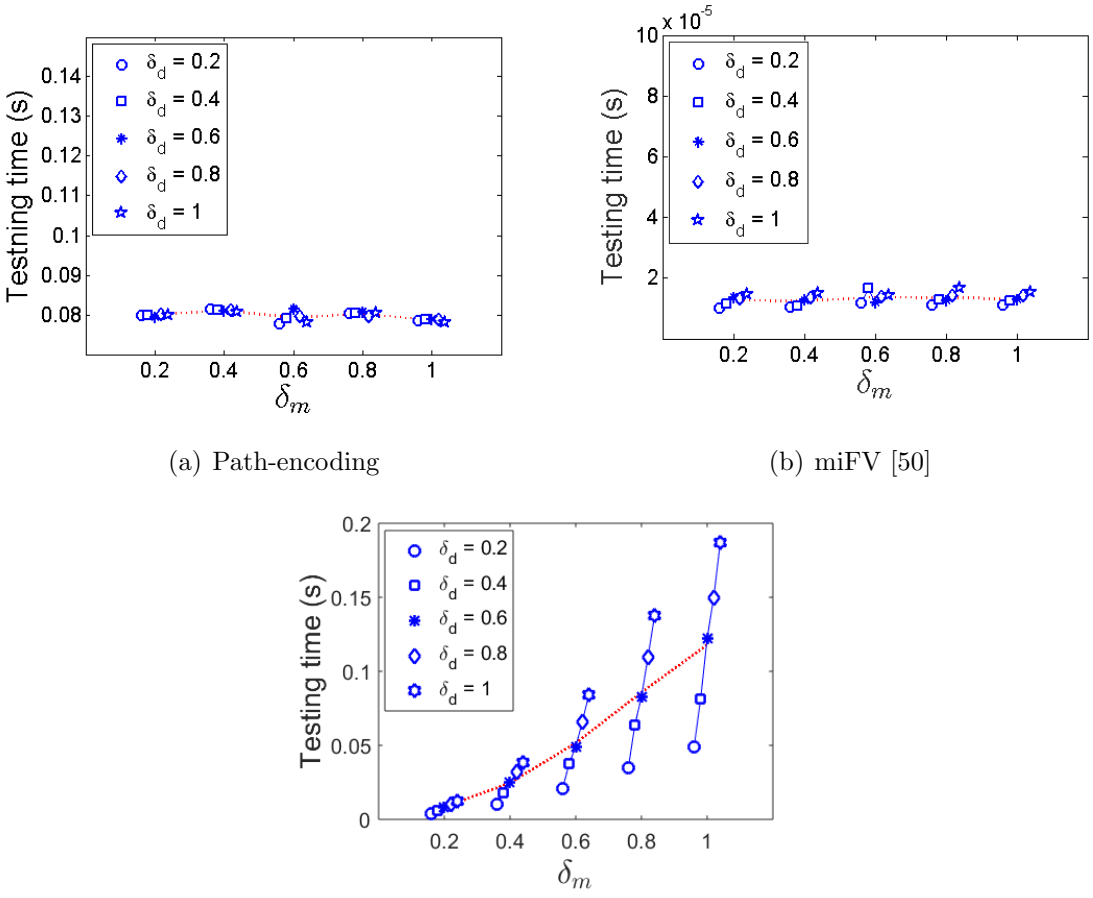
(a) Path-encoding

(b) miFV [50]

(c) RSIS [55]

Figure 3.11. Training times of path-encoding, miFV and RSIS on Elephant dataset with changing values of δ_m and δ_d.

The training and test times of path-encoding, miFV and RSIS on Elephant dataset are illustrated in Figure 3.11 and Figure 3.12, respectively. A linear increase in training time of path-encoding with increasing δ_m values in Figure 3.11(a) is consistent with the complexity discussed in Section 3.2.5. It is insensitive to the changes in the number of features because of the totally random selection of features in tree learning. Consequently, path-encoding is computationally very efficient for large databases with large number of features. The training time of miFV is schematized in Figure 3.11(b). The training time increases linearly with the increase in both δ_m and δ_d. A similar behavior is observed for RSIS in Figure 3.11(c), where the time required for training is highly affected from dataset dimensionality and number of bags. Along with the effect of classifier parameters and the dataset size, ensemble learning also increases



(a) Path-encoding

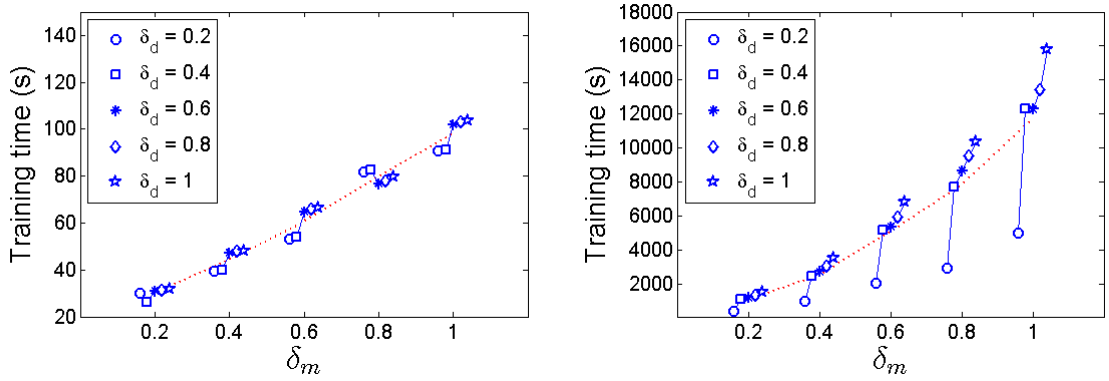
(b) miFV [50]

(c) RSIS [55]

Figure 3.12. Test times of path-encoding, miFV and RSIS on Elephant dataset with changing values of δ_m and δ_d.

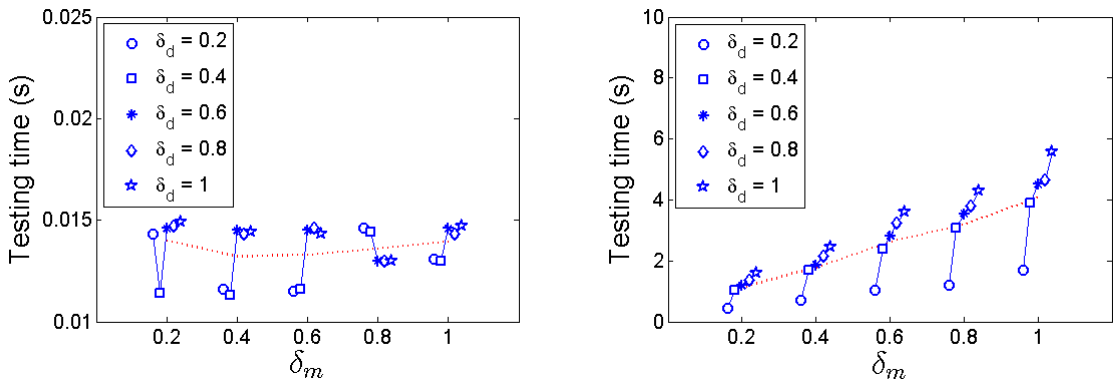
the time requirements of this method. Since longer training times might be due to implementation bias, we only compare trends in the run times.

Time required to obtain the test representation using path-encoding is insensitive to the problem characteristics as shown in Figure 3.12(a). Since testing requires only tree traversal in the ensemble and the tree building parameters are fixed, this result is expected. Testing time trend of miFV is shown in Figure 3.12(b). For each instance in the bag, distances to the centroids are calculated in this encoding algorithm. However, miFV test times are not affected by the dataset size as shown in Figure 3.12(b). This method is very efficient in obtaining the test representation. Finally, Figure 3.12(c) shows the increasing trend of RSIS testing times in terms of δ_m and δ_d. Scoring the



(a) Path encoding (b) $D_{meanmin}$ [26]

Figure 3.13. Training times of path-encoding and $D_{meanmin}$ on PASCAL VOC 2007 dataset with changing values of δ_m and δ_d.



(a) Path-encoding (b) $D_{meanmin}$ [26]

Figure 3.14. Test times of path-encoding and $D_{meanmin}$ on PASCAL VOC 2007 dataset with changing values of δ_m and δ_d.

instances in test bags according to multiple base classifiers introduces scalability issues in contrast with miFV and path-encoding, since these methods evaluate performance on a single bag-level learner in testing.

The representation learning times of path-encoding and $D_{meanmin}$ on PASCAL VOC 2007 dataset are illustrated in Figure 3.13. The behavior of path-encoding is illustrated in Figure 3.13(a), which is quite similar to the one observed for Elephant dataset. On the other hand, $D_{meanmin}$ is severely affected by the increase in number of training bags. This is due to the pairwise distance calculations between the instances

of bags. Observed training time for path-encoding remains acceptable even for $\delta_m = 1$, compared to the training time of D_{meanmin} schematized in Figure 3.13(b). In testing, dissimilarity calculations are still needed for D_{meanmin}, which increases the time for testing linearly with changing δ_m and δ_d as shown in Figure 3.14(b). Moreover, D_{meanmin} requires storage of whole training data to test a new bag, which is problematic while working on big datasets. Figure 3.14(a) demonstrates efficient testing time of path encoding on PASCAL VOC 2007 dataset.

The main advantage of path-encoding is its robustness to the encoding parameter selection. Overall, computational experiments indicate that miFV and path-encoding are computationally preferable. If the gain in computation time is of first priority, terminal node-encoding can be preferred instead of path-encoding as terminal node-encoding generates smaller representations. Moreover, proposed approaches are embarrassingly parallel and computation times can be decreased significantly by exploiting parallelism.

3.5. Conclusions

Use of bag-level representations for MIL has been shown to provide successful results in the literature. This study proposes a robust framework that uses random trees to partition the feature space together with either a terminal node, or a path-based representation. Our encoding implicitly learns generalized Gaussian Mixture Model (GMM) on the instance feature space and this information is transformed into a bag-level summary. Proposed representations provide very fast and competitive results on benchmark datasets from different domains.

Tree-based encoding inherits the desirable properties of decision tree learners. As opposed to earlier proposals, our methods do not impose any distribution assumptions for the instances. Without any preprocessing step such as standardization, instance vectors are partitioned randomly and a simple sparse representation based on the frequency of instances is obtained to represent the bags. This enables the use of standard classification algorithms on the new representation. Proposed approaches are

reasonably different from the methods following standard MIL assumption.

Many existing studies lack a standard experimentation strategy which complicates comparison of the approaches. The experimental results are reported as the averages of randomly selected train/test splits and the same splits are not being used for all benchmarking methods in some studies. Moreover, classification performance is reported on only a few datasets. To promote reproducible research, we create a website that serves our implementations and codes of the competitive methods. We also share the datasets and corresponding cross-validation indices used for comparison. To the best of our knowledge, this study considers the largest database in evaluation of classification approaches in MIL.

To conclude, simple instance feature space partitioning approaches are utilized to learn representative and assumption-free feature vectors for the bags. Although not explored thoroughly, the encoding algorithms can also be used for the classification of multi-class MIL problems. Another interesting research direction is incorporation of the bag label information in bag encoding phase. As a subclass of semi-supervised learning problem, MIL can benefit from the bag labels.

4. A LINEAR PROGRAMMING APPROACH TO MULTIPLE INSTANCE LEARNING

4.1. Introduction

Multiple instance learning (MIL) concerns with classifying objects where each object is represented with a bag containing multiple instances. The main motivation of MIL is to respect the complete internal structure of an object with a collection of multiple instances. Compared to standard supervised learning problems, where each instance has a label, only the bags are labeled. For example, images are generally represented by a collection of patches in computer vision. This way, certain problems regarding the location or scale invariance can be avoided. Moreover, MIL framework is suitable to a diverse domain of applications such as molecule activity prediction [5], image categorization [8], web mining [82] and audio recording classification [10]. In MIL, the label information is provided for bags and instance labels are unknown. Even when instance labels are known, there should be a rule/model providing the bag label information. In any case of (labeled/unlabeled) instances, bag-level summary of the instance distribution is required. To resolve this problem, most of the existing studies make assumptions regarding the instance labels. For example, *standard* MIL assumption prevails in most of the existing MIL approaches. In standard MIL problem, there is at least one positive instance in positive bags and all other instances in given data are negative. Since bag positivity is determined by a few instances, standard MIL methods focus on labeling these potentially positive instances.

Considering the limited structure of standard MIL, a variety of assumptions on relating instance labels with bag labels are introduced in [90] as *generalized MIL*. In generalized MIL, a certain portion of potentially positive instances must be contained in positive bags. Moreover, these positive instances may belong to different data regions of the instance-feature space and are effective on the bag labels. A hierarchy of MI assumptions in generalized MIL problems is proposed by Weidmann as *presence-*

based, *threshold-based*, and *count-based* MI assumptions [20]. A *concept* is a formation defined in the underlying instance-feature space which characterizes some subset of instances. Requirement of at least one instance from each concept forms the *presence-based* assumption in Weidmann's hierarchy. Presence-based assumption is followed by *threshold-based* assumption where certain number of instances from multiple concepts define bag positivity. Finally, *count-based* assumption requires existence of additional upper and lower bounds on number of the instances from multiple concepts. Weidmann's hierarchy [20] is later comprehensively extended in [21]. Beside the standard assumption, they proposed so called *collective* assumption [91] in which each instance equally and independently contributes to the bag label.

A wide range of MIL methods prioritize generalized MIL to embrace different MIL applications by managing multi-instance data [22]. Main point of the discussion in [22] is that MIL methods differ from each other based on how they managed the relationships between instances residing in bags. To tackle generalized MIL problems, we predict bag class labels by aggregation of instance contributions. Instance-level scores are obtained by an appropriate mapping function of feature weights. Then, a bag is represented by simply averaging the instance-level scores, which is analogous to the collective assumption. This kind of approach deals with a variety of MI assumptions by optimizing feature weights to assess contribution of each instance to the bag label. In the cases where only a few instances in positive bags are responsible for class differentiation, equally weighting the instance-level scores may not be capable of representing the bag classes. Therefore, *weighted collective* assumption is proposed in [91] to highlight the predictions associated with the representative instances in positive bags while computing the bag class estimates.

Researchers make use of margin maximization based approaches to solve MIL problem [7, 36, 92]. Generally, inter-bag margin is maximized but the ways of relating instance margin to bag margin differ. More importantly, most of the existing optimization-based methods suffer from scalability problems, which is a major challenge in MIL problems. Considering the limitations of previous approaches, we propose a novel linear programming-based MIL framework [93]. As opposed to margin maxi-

mization based MIL models, we build MIL classifiers using a simplified optimization framework. Our approach models the contributions of instances to the bag labels rather than individually labeling them. The instance level contributions are implicitly mapped into a latent variable to obtain the bag class membership estimates.

Figure 4.1 shows the way of mitigating instance information to obtain a bag-level mapping on an illustrative example from UCSB Breast Cancer dataset [9]. Two cellular images belonging to malignant (positive) class and benign (negative) class are considered as bags. Instances of the bags are sampled as square patches of the images on a grid as exemplified in Figure 4.1. In classification, the aim is to predict the label of a bag given its set of instances. Instance-level estimates between 0 and 1 are calculated by a linear decision function. For each bag, scores of corresponding instances are averaged to assess bag-level class probability estimate. Classification scores of the bags in Figure 4.1 are predicted as 0.76 for the positive bag, and 0.22 for the negative bag by simply averaging the pseudo-class memberships of corresponding instances.

In our proposal, we also process all training instances and their relationships to determine bag classes. It is shown in [94] that there is weak correlation between bag-level and instance-level performance of MIL classifiers. Hence, instance labels are not necessarily to be predicted correctly and true labels of instances are not known in most of the datasets. Besides, instances of a bag may contain both relevant and irrelevant information, which is the drastic structure of MIL compared to supervised learning. In the described example, only the final bag label estimate is sufficient for diagnosis of the disease as shown in Figure 4.1. This way, instances and corresponding bags are related without enforcing any requirements on the binding MIL assumption. Note that certain informative instances from the concept regions are prioritized by using a scoring idea to assess bag-level estimates. Similarly, insignificant instances are ineffective through proper determination of their scores. Bag class labels are determined based on instance level pseudo-membership scores analogical with the collective MI assumption [91].

Resulting classifiers are linear functions in the given feature space, and have low capability of modeling nonlinear decision boundaries. An appropriate transformation

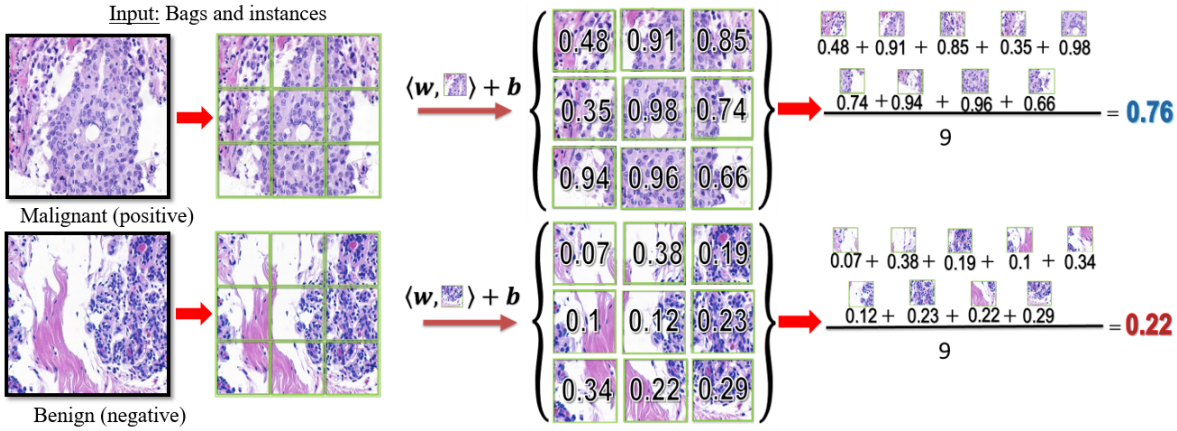


Figure 4.1. An example of bag class membership estimation.

of the original features is needed to apply classifiers to nonlinear data. As mentioned in [56], bags are not independently identically distributed samples of the underlying instance-feature space. Exploiting unsupervised dissimilarities leads to capture the unknown and potentially nonlinear relationships between instances from positive and negative bags. An instance selection method, MILES [8] selects the most important pairwise instance dissimilarities that characterize positive and negative classes. To capture nonlinear relationships among all training instance vectors, we consider an instance dissimilarity based data representation. The new features are the dissimilarities to all training instances which embed bags to a higher dimensional space. Increased number of instances introduces computational difficulties to classifier training especially for large scale datasets. To reduce dimensionality of the new space, we propose a simple and unifying data representation alternative. Using clustering, a low dimensional and summarized representation of the instances is obtained. Instances in training bags are pooled and then clustered to form the new features. Instead of instance dissimilarities, new features of the representation are distances to cluster centers. Dimensionality of the constructed data representation only depends on the number of clusters.

Our mathematical programming approach to MIL takes into consideration rankings of positive and negative bags. Pairwise comparisons of positive and negative bag label estimates form the rankings. The idea is to ensure that each positive bag has a higher label estimation value than the negative bags. After optimization, the

output linear scoring function is used to find instance-level pseudo-class membership estimates. Bag class labels are determined based on their instances' scores without forcing a specific MIL assumption. Most of the instance-level MIL approaches adopt standard MIL assumption. The first MIL paper [5] introduces formal descriptions of both MIL problem and standard MIL assumption whereas [23] presents a survey on standard MIL methods. In addition to the first MIL method axis parallel rectangles (APR) [5] and Citation-kNN [46], a generative method Diverse Density (DD) [13] and its variant EM-DD [30] also solve standard MIL problem. A famous MIL method, MILES [8] performs embedded instance selection iteratively and assumes instances in both positive and negative bags belong to the target concept. Prevention of long computational times in MILES [8] is achieved in MILIS [28] via pruning instances after density estimation and then updating the classifier iteratively.

Aforementioned methods incorporate machine learning algorithms and their performance depend on the adaptation process to given data, such as fine tuning of parameters and data preprocessing. Hence, it is hard to prove that these methods suit up to a wide range of datasets. Mathematical programming approaches are also considered to solve MIL problems. MIL formulations in the literature are extensions of generic SVM model [7, 32, 34–36] where instance level margin maximization is performed for bag classification initially assuming that all instances in positive bags are positive. To compensate the impact of this assumption, a witness selection procedure is employed [7, 35, 36]. For each bag from positive class, an instance is selected as a witness to represent that bag. However, only standard MIL assumption suits this specification. Sparse transductive MIL formulation (stMIL) [32] handles the issue of low witness rates by pulling all the negative instances in the bag closer to the hyperplane. A Concave Convex Procedure (CCCP) is proposed to solve resulting non-convex formulation. However, this method approach poorly performed when the witness rate is high. To resolve this issue, they proposed sparse balanced MIL (sbMIL) formulation where the number of witness instances is controlled. In mi-SVM and MI-SVM formulations [7], two types of constraints are added to the SVM formulation satisfying at least one sample in each positive bag has a label of one in mi-SVM and a witness instance is present for positive bags in MI-SVM. A 1-norm SVM-based formulation [34]

incorporating the assumption "arbitrary convex combination of instances in the positive bags represents each positive bag" is proposed, which is a linear program with bilinear constraints. Their solution approach guarantees convergence to a local optimal solution, rather than the global optimal. MIL problem is formulated as a mixed $0-1$ quadratic programming problem in [35], where the MIL is reduced to instance-level learning and only the standard MIL assumption is taken into account. Their proposed branch and bound heuristic outperformed the direct solutions of the models. In [36], SVM formulations of MIL problem are derived as a hard margin maximization formulation (MIHMSVM) and two soft margin maximization models, MIHLSVM and MIRLSVM with additional bag-level misclassification penalties. A penalty is incurred if all instances in a positive bag are misclassified or at least one instance in the negative bag is misclassified. Exact solution methods like CCCP in [32] are time consuming. Heuristic methods proposed in [35,36] are considerably fast in problems with moderate sized datasets but do not guarantee the quality of final solution [34]. As opposed to quadratic or mixed-integer quadratic programs, we solve models with a linear objective function and constraints. Furthermore, instead of repeatedly solving subproblems, we solve a single linear program, which is solvable in polynomial time. Instances and corresponding bags are related without enforcing any requirements on the binding MIL assumption.

Discriminative methods Citation-kNN [46], mi-SVM [7], MI-SVM [7] and KI-SVM [33] perform instance level learning and permit witness identification. However, these methods require a higher rate of witness instances in positive bags to shift the classification success [55]. Moreover, witness instances selected from positive bags may belong to various regions of the instance-feature space. This alternative characterization is called multiple concept assumption. When positivity of bags is due to multiple concepts that spread into distant regions, relationships between instances must be identified to represent the bags. A typical way of modeling instance relationships is using the dissimilarities between instances. A subset of instances or a representative set selected from the instance-feature space is referred to as prototypes. Dissimilarity based MIL methods [8, 27, 29, 56, 95] exploit dissimilarities to the prototypes to extract useful information with various data representations. MILES [8] and MILD [29] embed

instances to a new space using dissimilarities to a set of instances and therefore heavily depend on the instance selection procedure. MILDS [27] is another bag embedding approach that utilizes an output set of a pairwise clustering algorithm referred to as dominant sets. For each bag, the instance that is farthest from the negative prototypes is selected to obtain positive prototypes. A similar prototype is constructed in Clustering MIL [95], which is a spherical area obtained by clustering all positive instances. MILES [8] and MILD [29] assume that instances from different concepts are independently identically distributed, whereas MILDS [27] and Clustering MIL [95] select only some instances as prototypes thereby waiving instance relationship information. Differently from the aforementioned methods, we learn representations by processing all instances and subsequently model instance contributions to bag labels.

In some MIL applications, bags may have common instances like overlapping passages in text classification [7] or noisy inputs like instances extracted by bounding boxes in image segmentation [96]. A graph kernel is employed in MIGraph [56] to capture underlying manifold structure of data assuming that bags are not independently and identically distributed samples. A recent method, Random Subspace Instance Selection (RSIS) [55] inherits required knowledge by projecting data onto several random subspaces and subsequently clustering instances to calculate instance positivity scores. Then, it averages instance scores to estimate bag labels. This method eliminates the assumptions about instance distributions and robust to various witness rates. Similarly, our optimization procedure aims to discover effective instances for bag classification and outputs an instance-level scoring function.

Previously, the multiple concept structure is captured at bag level by using bag kernels [39], Hausdorff distances between bags in Citation-kNN [46] and bag dissimilarities in MInD [26]. In bag-level MIL approaches, a set of vectors are determined to represent bags and thereafter a bag dissimilarity measure is used to calculate bag-to-bag distances. In particular, MInD [26] compares bags by defining various bag-to-bag dissimilarity measures and demonstrates successful performance on a wide range of MIL datasets. Main advantage of bag level representation is the direct application of standard supervised learning methods. Instead of bag-to-bag dissimilarities, we classify

bags by simply using the instance dissimilarities. Instance-level relationships are considered to benefit from the informative instances in bags since positive and negative bag classes may possess instances that are very similar to each other. Therefore, concept instances are promoted by modeling the instance relationships during optimization.

The remainder of this chapter is organized as follows. Section 4.2 describes the proposed linear optimization based MIL framework. The computational results and discussions are presented in Section 4.3. Finally, the conclusions are given in Section 4.4.

4.2. Linear Programming for Multiple Instance Learning

To formulate MIL problem as a linear programming (LP) model, we define the sets, parameters and decision variables used in the model as follows.

Indices:

$i = 1, 2, \ldots, n$: indices for the instances

$j = 1, 2, \ldots, m$: indices for the bags

Sets:

$J^+ = \{j : l_j = 1\}$: set of positive bags

$J^- = \{j : l_j = -1\}$: set of negative bags

$J = J^+ \cup J^-$: set of all bags

$I^+ = \{i : i \in I_j \wedge j \in J^+\}$: set of instances in positive bags

$I^- = \{i : i \in I_j \wedge j \in J^-\}$: set of instances in negative bags

$I = I^+ \cup I^-$: set of all instances

Parameters:

$\mathbf{x}_i \in \Re^d$, $i = 1, 2, \ldots, n$: instance vectors

l_j, $j = 1, 2, \ldots, m$: bag labels

Decision variables:

$\mathbf{w}$: d-dimensional feature weight vector

b: bias of the linear function

m_i, $i = 1, 2, \ldots, n$: instance pseudo class memberships

β_j, $j = 1, 2, \ldots, m$: bag class memberships

σ_{jl}, $j \in J^+$, $l \in J^-$: bag class membership differences

Our learning approach ranks the bags in a binary classification problem. Namely, a positive bag is ranked before an arbitrary negative bag after classification. Area under the ROC curve (AUC) is the most commonly used measure to evaluate the success of ranking problems. Using a least-squares SVM algorithm, [97] solves AUC maximization problem by comparing positive and negative instance pairs. AUC can be calculated using Wilcoxon-Mann-Whitney (WMW) statistic [98], which can be written for positive and negative bags as

$$W = \frac{\sum_{j \in J^+} \sum_{l \in J^-} 1(\beta_j \geq \beta_l)}{|J^+||J^-|}$$

where $1(\cdot)$ is an indicator function that equals one if its argument is true, and equals zero otherwise.

WMW statistic yields the quantity of positive bags having higher rank compared to the negative bags, which is divided by the number of all possible bag pairs. Our LP model minimizes pairwise positive and negative bag class membership differences, which is also improves the bag ranks [99]. Therefore, comparison of positive and negative bag pairs can also be casted as solving AUC maximization problem.

Instead of labeling each instance individually, determination of class membership scores permits contributions of instances from multiple concepts with different importance degrees to the bag class. Hence, membership values are not assessed by favoring a specific target concept as observed in the standard MIL problem. This property emphasizes the superiority of our approach compared to the margin maximization based methods where standard MIL assumption is deemed [7, 33, 46]. Finally, a linear binary MIL classifier is built by solving the following model:

$$\text{(LP)} \quad \max_{\mathbf{w},b,\boldsymbol{\beta},\mathbf{m},\boldsymbol{\sigma}} \quad \sum_{j\in J^+}\sum_{l\in J^-}\sigma_{jl} \tag{4.1a}$$

$$\text{st} \quad \langle \mathbf{w}, \mathbf{x}_i\rangle + b = m_i \quad \forall i \in I \tag{4.1b}$$

$$\beta_j = \frac{1}{n_j}\sum_{i\in I_j} m_i \quad \forall j \in J \tag{4.1c}$$

$$\beta_j = \beta_l + \sigma_{jl} \quad \forall j \in J^+, \forall l \in J^- \tag{4.1d}$$

$$0 \leq m_i \leq 1 \quad \forall i \in I \tag{4.1e}$$

The values of variables $m_i, \forall i = 1, 2, \ldots, n$ correspond to instance pseudo class memberships which are bounded by Constraint (4.1e). As introduced, $\mathbf{w}$ is the feature weight vector, whereas b is the bias parameter that are optimized to form an instance level separating hyperplane. This hyperplane decides the instance pseudo class memberships in Constraint (4.1b). Constraint (4.1c) forms the bag class memberships $\beta_j, \forall j = 1, \ldots, m$ based on the summation of instance pseudo class memberships for each bag, which is normalized with the size of the corresponding bag, n_j. Constraint (4.1d) characterizes the bag differences for each positive and negative bag pair which are imposed by the slack variables σ_{jl}, $\forall j \in J^+$ and $\forall l \in J^-$. Finally, the objective function (4.1a) maximizes the summation of these slack variables to maximize bag class separation. The resulting model is efficient to solve since it has a linear objective function and constraints. All the instances in training bags constitute to the classifier during optimization. LP solution provides a classifier $\langle \mathbf{w}, \mathbf{x}_i\rangle + b$ which determines instance pseudo-class membership value for an arbitrary d-dimensional instance vector $\mathbf{x}_i$, i.e. $m_i = \langle \mathbf{w}, \mathbf{x}_i\rangle + b$.

For each instance in the dataset, a membership value between 0 and 1 must be decided to map the bag level estimates onto the 0 to 1 interval. We regard this membership value as pseudo class label estimate. If the membership value is less than a threshold, the instance can be assigned to the negative class. Otherwise, the instance is considered to belong to the positive class. The threshold can be selected

based on the highest accuracy level on training bags. We assess the pseudo-membership values of instances to find bag-level estimates, not for instance labeling since the actual instance labels are not known in MIL tasks. Each bag has a class membership value which is obtained related to membership values of its instances. Class membership estimates for bags are determined by averaging pseudo class membership values of its possessed instances as $\beta_j = \frac{1}{n_j}\sum_{i\in I_j} m_i, \quad \forall j \in J$. This representation eliminates single witness instance selection encountered in previous proposals and leads to an optimization problem with continuous variables and linear constraints. To classify a test bag, instance level scores are calculated and then averaged to find bag class label estimates. Such an approach is simple and efficient to implement and optimize and there are no hyperparameters that need to be tuned.

4.3. Experiments and Results

4.3.1. Data Representation

Input data representation plays a crucial role in data mining tasks. Success of the proposed algorithms are varied based on selected feature representation. Many research works have benefited from instance dissimilarity-based representation instead of feature-based representation [100]. In MIL, it is not enough to describe objects with multiple instance vectors, the relationships between these vectors must also be represented. The researchers conducted MIL experiments on various data representations by calculating the dissimilarities to selected prototypes [8,26–29]. As discussed before, methods like MILES [8], MILIS [28] and MILDS [27] perform instance selection during prototype generation to reduce data dimension. Moreover, a summarized information inherited by bag-to-bag dissimilarities as in [26] may extinguish some effective instance dissimilarities for classification. To overcome this issue, same authors propose randomly selecting the instances to form multiple subspaces and then training a classifier in each subspace [48]. In our LP-based MIL framework, we preprocess the input data to allow learning different characteristics of MIL datasets. Solving LP model produces a decision boundary by means of a linear classifier. Most of MIL datasets are formed of complex objects with potentially nonlinear instance relationships. The input data can

be transformed to carry out nonlinear classification in a new, possibly higher dimensional space. A linear classifier is simple to apply and capable of nonlinear separation in the new feature space [100].

Given a set of bags $\mathcal{X} = \{B_1, \ldots, B_m\}$, each bag B_j is composed of n_j many instances. The original instance-feature space is described with d many features. It is practical to transform the original input $\mathcal{X}$ using function $\phi(\mathbf{x}_i)$, which admits to another representation of input data, say $\mathcal{X}'$. For instance, the similarities to prototype instances [8], or a graph kernel [56] transforms the original data to discover its underlying structure. Given $\mathcal{X}$ or $\mathcal{X}'$ with bag labels l_j, $j = 1, \ldots, m$, our MIL task is to predict labels of unseen bags based upon a linear decision function.

Initially, both training set and test set are preprocessed by standardization using the feature means and standard deviations throughout the experiments. Preliminarily, we processed pairwise training instance dissimilarities to learn a MIL classifier. The dissimilarities between instances $\mathbf{x}_i$ and x_k are calculated by using the squared Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_k) = (\mathbf{x}_i - \mathbf{x}_k)^T (\mathbf{x}_i - \mathbf{x}_k)$. In a test bag, distances to all training instances are calculated for each instance of that bag. The dimensionality of the new space equals to total number of instances in training bags, i.e., n and the new representation is referred to as $\mathrm{R}^{\mathrm{instance}}$. When n is large, there are large number of variables in LP model which introduce computational difficulties. Moreover, since the $n \times n$ dimensional instance dissimilarity matrix is large and dense, the resulting mathematical model also has dense columns. Consequently, the solution time is affected from dense columns especially for large datasets. Curse of dimensionality and overfitting due to noisy features in the enlarged representation are categorized as the further problems. Thus, alternative representations can be considered to avoid solution of large models and prevent overfitting on large datasets. To solve LP model on large-scale MIL problems, we offer a simplified version of the first data representation using clustering.

Clustering instances is conducted in MIL setting either to detect the target concept [95] or to obtain a new bag-level data representation [49, 101]. Constructive

clustering-based ensemble (CCE) utilizes data clustering to obtain a new representation. All instances in bags are pooled and then repeatedly clustered. In each repeat, different number of clusters are generated. Representative instances in positive bags which belong to various concepts may reside in too small clusters after k-means clustering in some MIL datasets. Hence, processing only a subset of the clusters (i.e. the largest cluster) as in [95] is not always effective. In our clustering-based data representation, instead of all instances in training bags, cluster centers are selected as prototypes. After clustering the instances using k-means algorithm, instance-to-prototype distances build up the input data. Since dimensionality of the input dissimilarity matrix is decreased by clustering (i.e., there exists κ many clusters), clustering-based data representation is advantageous in datasets with large number of instances. We define the dissimilarity between instance $\mathbf{x}_i$ and cluster center $\mathbf{c}_j$ as $r_{ij}^c = (\mathbf{x}_i - \mathbf{c}_j)^T (\mathbf{x}_i - \mathbf{c}_j)$ where $\mathbf{c}_1, \ldots, \mathbf{c}_\kappa$ are the cluster centers. As a result, each instance is described by a κ-dimensional feature vector. In the final representation, which is denoted by $\mathrm{R}^{\mathrm{cluster}}$, the total number of distance calculations are reduced compared to $\mathrm{R}^{\mathrm{instance}}$ since the selected prototypes are cluster centers instead of all training instances.

The assumption that positively considered instances form a compact cluster in instance-feature space reveals in some MIL problems. For instance, in Newsgroups [56], the number of witnesses is low and these outlying witnesses are dissimilar to each other as discussed in [26, 55]. Assuming generation of enough number of clusters, dissimilarities to the outlying instances form meaningful feature vectors for classification. Since instance label information and binding MI assumption are the two main ambiguities of MIL problems, determination of the informative instance dissimilarities is necessary to remove uncertainty in bag classification. The two alternative representations can be tested on a subset of the given data to understand the underlying structure of the whole data. Simple calculations are performed by selected Euclidean distance metric and no parametrization is required to obtain $\mathrm{R}^{\mathrm{instance}}$ representation. In order to reduce computational time, $\mathrm{R}^{\mathrm{cluster}}$ representation can be exploited.

4.3.2. Experimental Setup and Evaluation Criteria

Initially, we transform the data to zero mean and unit variance. We perform 5 repeats of a stratified ten-fold cross validation to evaluate the classifier performance on each dataset. LP problems are modeled in Gurobi Python interface and solved using Gurobi 7.5 [102]. Input data representations are acquired using scikit-learn [79] library. All the experiments are carried out on a Windows 10 system with dual core CPU (i5-3470, 3.2 GHz) and 12 GB of RAM. In order to perform a fair comparison over state-of-the-art MIL methods, we use the same train/test split indices for each method and experiment. All the scripts, datasets and cross-validation indices are made available on our supporting page [63]. $\mathrm{R}^{\mathrm{instance}}$ representation has no parameters to be predetermined whereas $\mathrm{R}^{\mathrm{cluster}}$ has the input parameter number of clusters κ. The commonly used statistical approach of setting the best number of clusters is cross-validation. We simply identify value of κ using the elbow method based on total within cluster variance and increase the gain in computational time. After learning the representations, LP formulation in Model (4.1) is solved to obtain the bag classifier.

The convergence tolerance for the barrier algorithm is set to 0.01 and default values of the solver are used for the other parameters. Finally, state-of-the art approaches are experimented via their provided MATLAB [85] implementations. We followed the settings proposed by the authors. MInD [26] employs default parameters. The parameters of miFV [50] are PCA energy, number of components and cost parameter of linear SVM. These parameters are selected by an inner ten-fold cross-validation. PCA energy is selected from the set $\{0.8, 0.9, 1\}$ and the number of Gaussian components alternatives are $\{1, 2, 3, 4, 5\}$. The cost parameter levels of the linear SVM classifier are $\{0.05, 1, 10\}$.

Performance of a MIL classifier can be evaluated by AUC [80]. AUC is a more discriminative measure than accuracy [103] since a predetermined decision threshold is necessary to report accuracy. Besides, AUC maximization is related to maximization of positive and negative bag membership differences in LP model. In particular, the probability of a bag from positive class has a higher class membership score than a

bag from the negative class is estimated by AUC given a MIL classifier. AUC also improves classification accuracy by ranking positive bags ahead the negative bags, and is therefore an appropriate evaluation metric for our experiments.

4.3.3. Results

We perform experiments on real world MIL datasets to verify the effectiveness of our approach. MIL datasets are described in Table 3.1 and are categorized based on the application domain. To the best of our knowledge, this is the largest MIL dataset repository with reported results on a proposed MIL framework. Each dataset has different characteristics such as number of bags, number of instances in bags and number of features. In addition, minimum and maximum number of instances in bags, number of positive bags and number of negative bags are also provided in Table 3.1. For some datasets such as Corel [8] and Birds [10], class imbalance occurs at bag-level. Another property of the datasets is discussed in [55] is the low proportion of positive instances in positive bags, as observed in Newsgroups [56]. As a consequence, we tackled MIL problems from different application domains and investigate the utility of our MIL framework across various data characteristics.

To demonstrate the effectiveness and superiority of LP-based approach on real-world datasets, we also experimented the following baseline methods: miFV [50] and dissimilarity-based representations (MInD) [26] with D_{meanmin} representation. We solve LP problem (Model (4.1)) on R^{instance} and R^{cluster} representations of the datasets described in Table 3.1. At first, the significance of the differences are discussed according to the procedure recommended by [87]. A Friedman test [88] is applied to the ranks of the algorithms over all datasets. Since the null hypothesis that all methods have equal AUC performance at the 0.05 level, we proceed with the Nemenyi test [89] to check whether the pairs of classifiers are significantly different from each other. Pairwise differences of the methods are significant if their average ranks differ by at least the critical difference (CD). The resulting CD value for four classifiers at significance level 0.05 is 0.561. By using the rankings of the algorithms on each dataset and the average ranks, a CD diagram [87] shown in Figure 4.2 is obtained. Performances of LP with

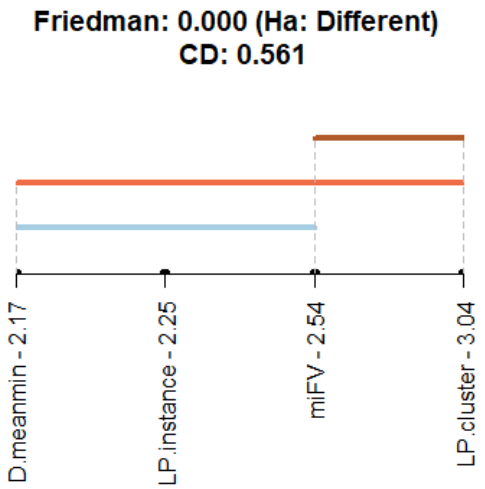
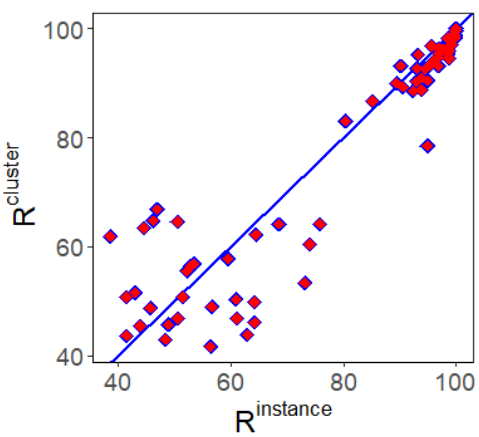


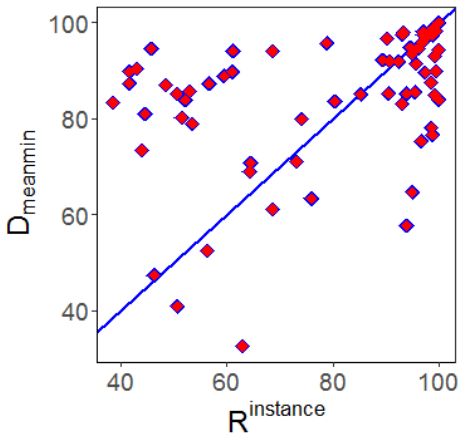
Figure 4.2. The average ranks for MIL methods on 71 datasets based on mean AUC performance. The critical difference at 0.05 is 0.561.

$R^{instance}$, MInD with $D_{meanmin}$ and miFV are not significantly different from each other according to the differences demonstrated in Figure 4.2. miFV and LP with $R^{cluster}$ are not significantly different from each other since their average rank difference is below the CD. Performance of LP model critically differs when either $R^{instance}$ or $R^{cluster}$ representations form the input data.

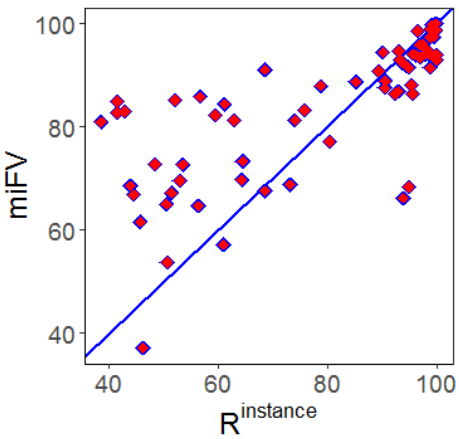
Scatter plots in Figure 4.3 shows the pairwise comparisons of the approaches. Two methods equally perform on a dataset if the corresponding point falls on the line $x = y$. The points falling below the line $x = y$ represent the datasets that are more accurately classified by the method on the x axis. Otherwise if a point is above the line $x = y$, the approach on the y axis is more successful on the corresponding dataset. Figure 4.3(a) shows the scatter plot comparison of LP results on $R^{instance}$ and $R^{cluster}$ representations and performance of $R^{instance}$ is more successful in 48 datasets. As seen in Figures 4.3(b) and 4.3(c), AUC results of LP with $R^{instance}$ are competitive with the other two methods. However, on a group of datasets performances of both $D_{meanmin}$ and miFV are superior, which are the text classification datasets. In real-world MIL applications except for text classification, LP with $R^{instance}$ is the leading method as the ranking results in Figure 4.4 indicates that and its difference with all other methods



(a) AUC comparison of $R^{instance}$ and $R^{cluster}$ representations



(b) AUC comparison of LP with $R^{instance}$ and MInD with $D_{meanmin}$



(c) AUC comparison of LP with $R^{instance}$ and miFV

Figure 4.3. Pairwise AUC comparison of various MIL methods on 71 real-world datasets.

is larger than the CD 0.733. We also compare LP solutions on $R^{instance}$ representation with $D_{meanmin}$ and miFV in detail using the scatter plots without Newsgroups and Web datasets as shown Figure 4.5. On the remaining problem categories, LP with $R^{instance}$ is slightly better than the other approaches as shown in the pairwise comparisons in Figure 4.5.

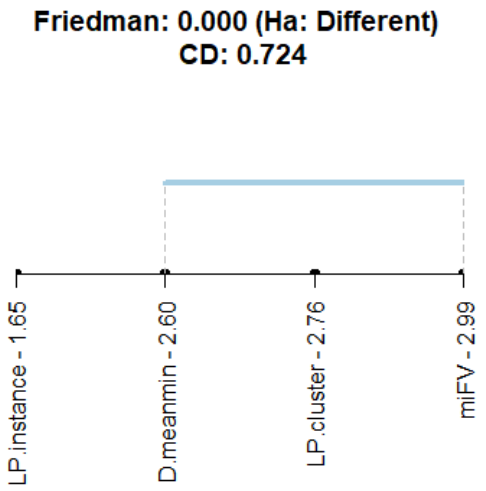
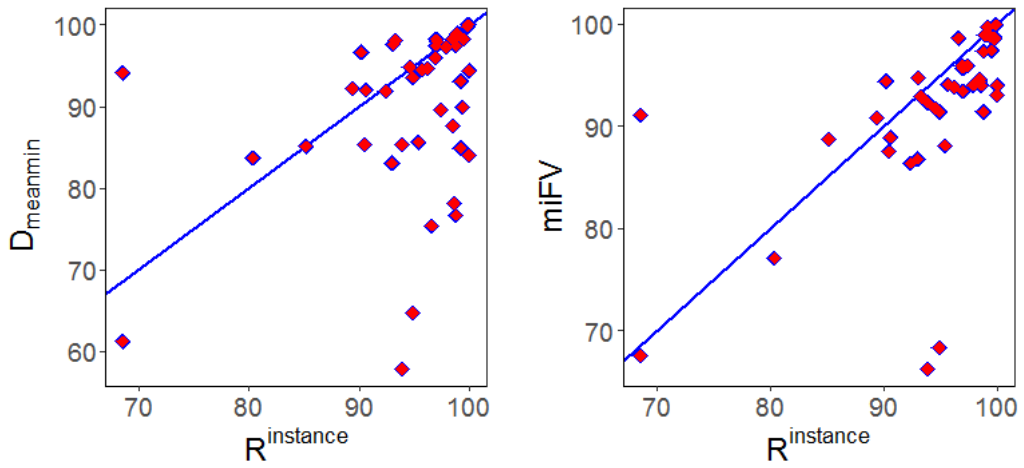


Figure 4.4. The average ranks for MIL methods on 42 datasets based on mean AUC performance. The critical difference at 0.05 is 0.733.



(a) AUC comparison of LP with $R^{instance}$ and MInD with $D_{meanmin}$

(b) AUC comparison of LP with $R^{instance}$ and miFV

Figure 4.5. Pairwise AUC comparison of various MIL methods on biology, image categorization and audio recording classification datasets.

AUC results of all methods on 71 datasets are provided in Table 4.1. LP model has superior performance on Musk 1 and Mutagenesis 2 datasets especially with the $R^{instance}$ representation. The best AUC result on Protein dataset is obtained by LP solution on $R^{cluster}$ representation. Result of LP with $R^{instance}$ representation on Pro-

tein dataset is not provided due to the memory restrictions. In Musk 2, MInD with $D_{meanmin}$ has the best classification performance which is followed by miFV. Best average results for Mutagenesis 1 are obtained by miFV and LP with $R^{cluster}$ is the second best method. In most of the Corel image datasets, LP with $R^{instance}$ representation is the leading method in addition to its best performance on image datasets UCSB Breast Cancer, Elephant, Fox and Tiger. MInD with $D_{meanmin}$ also successful on Corel image datasets. MInD with $D_{meanmin}$ has the best performance on Newsgroups datasets whereas miFV performs better than other methods in Web recommendation datasets. The poor performance of LP formulations on text classification datasets and the proposals for improving the classification success are discussed in Section 4.3.5. Finally, LP with $R^{instance}$ representation is quite successful compared to the other methods in Birds datasets.

4.3.4. Computational Time Analysis

Time complexity of obtaining $R^{instance}$ representation using Euclidean distances to instances in training bags is $\mathcal{O}(n^2d)$ and no parametrization is required. We use k-means clustering algorithm to form the $R^{cluster}$ representation. Time complexity of k-means algorithm is $\mathcal{O}(In\kappa d)$ where κ is the number of clusters and I is the necessary number of iterations until convergence. After determining the κ many cluster centers, it takes $\mathcal{O}(n\kappa d)$ times to have the final $R^{cluster}$ representation. LP problems belong to the complexity class P [104]. We solved LP formulations using barrier solver of Gurobi version 7.5, which means that the solutions are generated in polynomial time. Besides, the testing times after LP solutions are $\mathcal{O}(n)$ for $R^{instance}$ and $\mathcal{O}(\kappa)$ for $R^{instance}$. The execution times are recorded including the data representation and classifier generation times. Specifically, we report training and testing times of data representation learning and the time taken to build a classifier which is the model solution time. We also report representation learning times of the leading methods miFV [50] and MInD [26] with $D_{meanmin}$. Unlike LP-based MIL, both miFV [50] and MInD [26] represents bags using a new bag-level feature vector. Then, bag representation vectors form the input of the linear SVM classifier in polynomial time. LibLinear package [105] is employed in

Table 4.1. AUC and standard error (× 100) results of various MIL methods.

Dataset	**Algorithm AUC (%)**			
	LP			
	$R^{instance}$	$R^{cluster}$	MInD ($D_{meanmin}$)	miFV
Musk 1 ♣	95.7 (0.9)	**96.8 (0.8)**	94.5 (1.2)	94.1 (1.2)
Musk 2 ♣	93.1 (1.0)	92.7 (1.1)	**97.6 (0.8)**	94.7 (1.2)
Mutagenesis 1 ♣	85.2 (1.5)	86.7 (1.3)	85.1 (1.2)	**88.7 (1.2)**
Mutagenesis 2 ♣	**78.8 (3.9)**	78.5 (4.0)	64.7 (5.3)	68.3 (5.0)
Protein ♣	-	**83.9 (1.4)**	52.3 (3.7)	80.0 (1.9)
Elephant ♥	**94.9 (0.5)**	90.5 (1.0)	93.6 (0.9)	91.4 (0.9)
Fox ♥	**68.6 (1.4)**	64.2 (1.5)	61.2 (1.7)	67.5 (1.5)
Tiger ♥	**90.5 (0.9)**	89.3 (1.0)	85.3 (1.1)	87.5 (1.1)
Corel, African ♥	94.5 (0.6)	93.2 (0.7)	**96.7 (0.4)**	94.4 (0.6)
Corel, Antique ♥	89.4 (0.8)	90.0 (0.5)	**92.2 (0.6)**	90.8 (0.6)
Corel, Battleships ♥	93.3 (0.6)	95.2 (0.4)	**98.1 (0.2)**	92.9 (0.6)
Corel, Beach ♥	**99.5 (0.1)**	98.8 (0.2)	98.3 (0.4)	97.4 (0.4)
Corel, Buses ♥	**97.9 (0.2)**	96.3 (0.3)	97.3 (0.4)	94.0 (0.7)
Corel, Cars ♥	94.6 (0.6)	92.6 (0.7)	**94.8 (0.5)**	91.7 (0.7)
Corel, Desserts ♥	**98.8 (0.1)**	95.9 (0.4)	97.4 (0.3)	97.3 (0.4)
Corel, Dinosaurs ♥	**98.5 (0.2)**	95.3 (0.3)	98.3 (0.2)	94.4 (0.5)
Corel, Dogs ♥	**92.4 (0.6)**	88.6 (0.8)	91.9 (0.7)	86.4 (1.2)
Corel, Elephants ♥	97.0 (0.2)	96.4 (0.2)	**98.2 (0.2)**	95.7 (0.4)
Corel, Fashion ♥	98.9 (0.4)	98.1 (0.1)	**99.0 (0.1)**	98.9 (0.2)
Corel, Flowers ♥	**96.2 (0.4)**	93.8 (0.5)	94.7 (0.6)	93.8 (0.6)
Corel, Food ♥	**99.8 (0.0)**	98.3 (0.1)	**99.8 (0.1)**	98.7 (0.1)
Corel, Historical ♥	**99.8 (0.0)**	98.8 (0.1)	**99.8 (0.0)**	98.5 (0.3)
Corel, Horses ♥	90.6 (0.6)	89.3 (0.7)	**92.0 (0.6)**	88.9 (0.8)
Corel, Lizards ♥	97.1 (0.3)	95.7 (0.5)	**98.0 (0.3)**	95.8 (0.5)
Corel, Mountains ♥	99.9 (0.1)	99.7 (0.1)	**100 (0.0)**	99.9 (0.0)
Corel, Skiing ♥	**96.9 (0.3)**	93.1 (0.5)	96.0 (0.3)	95.9 (0.4)
Corel, Sunset ♥	80.4 (1.2)	83.1 (0.9)	**83.7 (1.0)**	77.1 (1.3)
Corel, Waterfalls ♥	97.0 (0.3)	95.4 (0.3)	**97.5 (0.2)**	93.4 (0.5)
UCSB Breast Cancer ♥	**93.0 (2.0)**	90.3 (2.2)	83.1 (2.7)	86.8 (2.5)
Newsgroups 1, alt.atheism ♠	47.0 (2.5)	66.8 (2.8)	**94.1 (1.0)**	91.1 (1.2)
N.g. 2, comp.graphics ♠	61.0 (2.3)	50.4 (3.0)	**89.8 (1.6)**	57.2 (3.2)
N.g. 3, comp.os.ms-windows.misc ♠	44.6 (2.8)	63.4 (2.5)	**81.0 (2.1)**	66.8 (2.2)
N.g. 4, comp.sys.ibm.pc.hardware ♠	53.0 (2.7)	56.5 (3.2)	**85.7 (2.2)**	69.5 (2.4)
N.g. 5, comp.sys.mac.hardware ♠	50.6 (2.2)	64.6 (3.2)	**85.2 (1.6)**	65.0 (2.6)
N.g. 6, comp.windows.x ♠	59.5 (2.6)	57.8 (2.8)	**89.0 (1.7)**	82.2 (2.0)

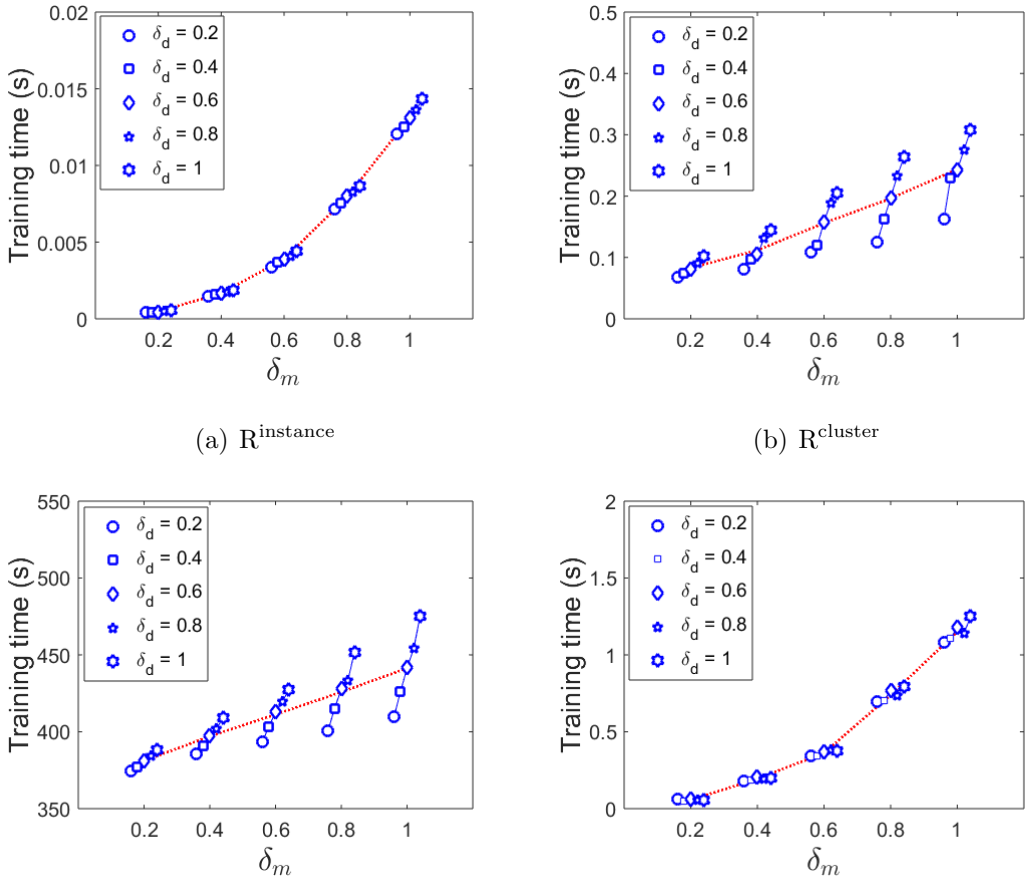
MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

Table 4.1. – AUC and standard error (× 100) results of various MIL methods (cont.).

Dataset	**Algorithm AUC (%)**			
	LP			
	$R^{instance}$	$R^{cluster}$	MInD ($D_{meanmin}$)	miFV
N.g. 7, misc.forsale ♠	53.5 (2.3)	56.9 (3.1)	**79.0 (2.0)**	72.6 (2.5)
N.g. 8, rec.autos ♠	48.5 (2.5)	43.0 (3.3)	**87.0 (1.7)**	72.7 (2.5)
N.g. 9, rec.motorcycles ♠	63.0 (2.8)	43.8 (2.7)	32.6 (3.2)	**81.2 (2.4)**
N.g. 10, rec.sport.baseball ♠	64.3 (2.4)	49.8 (3.0)	**91.4 (1.4)**	86.4 (1.8)
N.g. 11, rec.sport.hockey ♠	49.0 (2.5)	45.8 (3.2)	**95.8 (0.8)**	87.9 (1.5)
N.g. 12, sci.crypt ♠	52.2 (2.6)	55.5 (2.8)	84.0 (1.9)	**85.1 (1.8)**
N.g. 13, sci.electronics ♠	45.8 (2.1)	48.8 (4.0)	**94.6 (1.0)**	61.6 (2.6)
N.g. 14, sci.med ♠	61.2 (2.5)	46.8 (3.2)	**94.2 (0.8)**	84.3 (1.7)
N.g. 15, sci.space ♠	43.0 (2.3)	51.6 (3.1)	90.5 (1.4)	**82.9 (1.9)**
N.g. 16, soc.religion.christian ♠	41.6 (2.7)	43.7 (3.0)	**89.8 (1.4)**	84.9 (1.5)
N.g. 17, talk.politics.guns ♠	41.6 (2.7)	50.8 (2.8)	**87.4 (1.5)**	82.7 (2.0)
N.g. 18, talk.politics.mideast ♠	56.7 (2.5)	49.0 (3.1)	**87.4 (1.7)**	85.8 (1.9)
N.g. 19, talk.politics.misc ♠	51.5 (1.9)	50.8 (2.3)	**80.2 (1.9)**	67.2 (2.9)
N.g. 20, talk.religion.misc ♠	38.6 (2.3)	61.9 (2.7)	**83.4 (2.2)**	80.9 (2.3)
Web 1 ♠	75.9 (3.0)	64.2 (3.2)	63.4 (4.2)	**83.2 (2.3)**
Web 2 ♠	46.3 (4.1)	**64.7 (3.6)**	47.4 (4.2)	37.1 (2.5)
Web 3 ♠	64.5 (4.2)	62.2 (3.9)	70.8 (4.6)	**73.3 (3.6)**
Web 4 ♠	74.1 (3.7)	60.4 (3.8)	79.9 (3.6)	**81.2 (3.4)**
Web 5 ♠	**73.2 (3.5)**	53.4 (4.0)	71.1 (3.7)	68.7 (3.4)
Web 6 ♠	56.4 (4.4)	41.7 (4.4)	52.5 (4.2)	**64.6 (3.6)**
Web 7 ♠	64.3 (2.9)	46.1 (3.2)	69.0 (2.8)	**69.7 (3.4)**
Web 8 ♠	50.7 (3.0)	46.9 (2.4)	40.9 (2.6)	**53.7 (2.4)**
Web 9 ♠	44.0 (3.2)	45.5 (3.0)	73.5 (2.7)	**68.5 (3.1)**
Birds, Brown creeper ♦	**99.4 (0.1)**	98.4 (0.2)	89.9 (0.5)	98.8 (0.2)
Birds, Chestnut-backed chickadee ♦	**93.9 (0.4)**	88.8 (0.7)	85.3 (0.8)	92.3 (0.8)
Birds, Dark-eyed junco ♦	**95.4 (0.6)**	93.4 (0.7)	85.6 (1.3)	88.1 (1.2)
Birds, Hammonds flycatcher ♦	**100.0 (0.0)**	**100 (0.0)**	94.4 (0.7)	94.0 (0.7)
Birds, Hermit thrush ♦	**93.9 (1.4)**	90.9 (1.0)	57.8 (4.4)	66.2 (3.1)
Birds, Hermit warbler ♦	**98.6 (0.2)**	98.2 (0.2)	78.1 (1.5)	94.0 (0.6)
Birds, Olive-sided flycatcher ♦	**97.4 (0.2)**	96.2 (0.3)	89.6 (0.6)	95.9 (0.4)
Birds, Pacificslope flycatcher ♦	96.6 (0.3)	94.5 (0.4)	75.4 (1.0)	**98.6 (0.2)**
Birds, Red-breasted nuthatch ♦	**98.5 (0.2)**	94.7 (0.4)	87.6 (0.7)	94.6 (0.5)
Birds, Swainsons thrush ♦	**98.8 (0.2)**	94.5 (0.4)	76.7 (1.7)	91.4 (1.0)
Birds, Varied thrush ♦	**100.0 (0.0)**	99.6 (0.1)	84.0 (1.2)	93.0 (0.7)
Birds, Western tanager ♦	**99.2 (0.1)**	97.0 (0.3)	84.9 (1.8)	98.9 (0.2)
Birds, Winter wren ♦	99.2 (0.1)	98.5 (0.2)	93.1 (0.7)	**99.7 (0.1)**

miFV [50] to build a linear SVM classifier and corresponding time complexity is $\mathcal{O}(n)$, whereas MInD [26] uses LiBSVM [106] implementation where the linear SVM classifier learning time scales between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$. Prediction time of a test bag takes $\mathcal{O}(h)$ times where h is the dimensionality of the obtained bag representation. Note that the testing times of LP solutions and SVM classifiers of miFV [50] and MInD [26] are negligible since only a few vector multiplications and arithmetic operations are performed.

In order to observe the time complexity, pseudo-synthetic datasets have various properties such as number of bags and number of features are generated. All the methods are experimented on pseudo-synthetic datasets that originate from Elephant dataset. Proportion of bags δ_m and proportion of features δ_d are selected from the set $\{0.2, 0.4, 0.6, 0.8, 1\}$. We repeat 10 replications of each setting combination and plot the average results. Figure 4.6 shows representation learning times of LP-MIL, miFV [50] and $\mathrm{D}_{\mathrm{meanmin}}$ [26] on the training set. $\mathrm{D}_{\mathrm{meanmin}}$ [26] and $\mathrm{R}^{\mathrm{cluster}}$ increases linearly in terms of the increase in number of features and number of bags. In $\mathrm{R}^{\mathrm{instance}}$ representation and miFV [50], a cubic growth is followed as the number of bags increases. It can be seen from Figure 4.7 that testing times of miFV [50] and $\mathrm{R}^{\mathrm{cluster}}$ representation are robust to the changes in the data size properties. Effect of distance calculations degrade representation learning times both on training and test sets when number of bags and number of features are increased in $\mathrm{R}^{\mathrm{instance}}$ representation and $\mathrm{D}_{\mathrm{meanmin}}$ [26]. The performance of LP-based MIL especially depends on the model solution time. Once the LP model is built, the elapsed time during optimization is the classifier building time. Figure 4.8 shows the changes in model solution times for $\mathrm{R}^{\mathrm{instance}}$ and $\mathrm{R}^{\mathrm{cluster}}$ representations. Since dimensionality of $\mathrm{R}^{\mathrm{instance}}$ is proportional to number of the training instances, LP solution times can be challenging in datasets with large number of bags or instances as demonstrated in Figure 4.8(a). $\mathrm{R}^{\mathrm{cluster}}$ representation is simple and generally low-dimensional compared to $\mathrm{R}^{\mathrm{instance}}$. Moreover, linear increase in the solution time curve in Figure 4.8(b) when solving LP formulation on $\mathrm{R}^{\mathrm{cluster}}$ representation with increasing number of bags promotes this representation on large datasets.



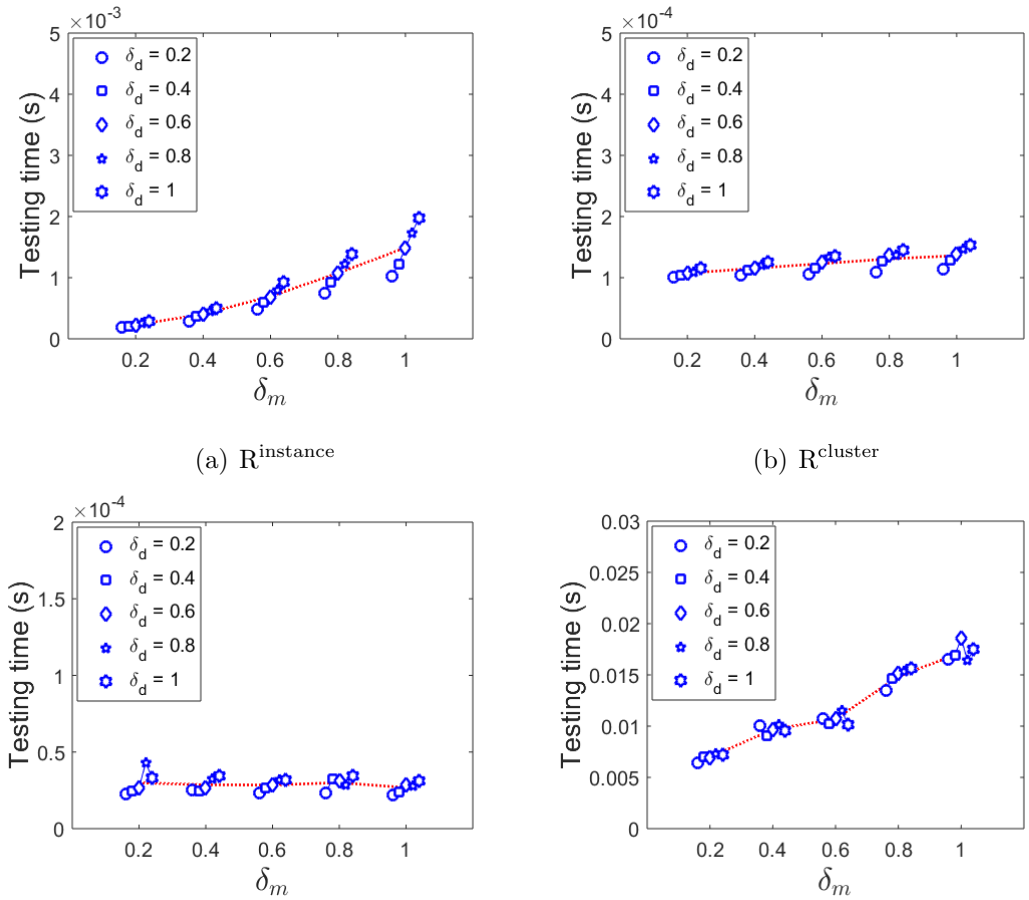
(a) $R^{instance}$ (b) $R^{cluster}$

(c) miFV (d) $D_{meanmin}$

Figure 4.6. Training times of LP-MIL, miFV and $D_{meanmin}$ on Elephant dataset with changing values of δ_m and δ_d.

4.3.5. Weighting Instance Memberships

In the proposed LP model, all instances in a bag contribute equally to the bag membership. Since instances can be related to bag labels in many different ways depending on the application [21], we propose an alternative way of encoding the bag class memberships. Besides, some real world datasets might have low witness rates [55]. For instance, there are multiple positive concepts and proportion of positive instances in positive bags is low in Newsgroups datasets [56]. Namely, the influence of witness instances to bag positivity is larger in positive bags. Our LP models poorly perform on Newsgroups datasets [56] due to the equally weighted instance contributions to the bag label, as considered in collective assumption of MIL. To support adaptability to this



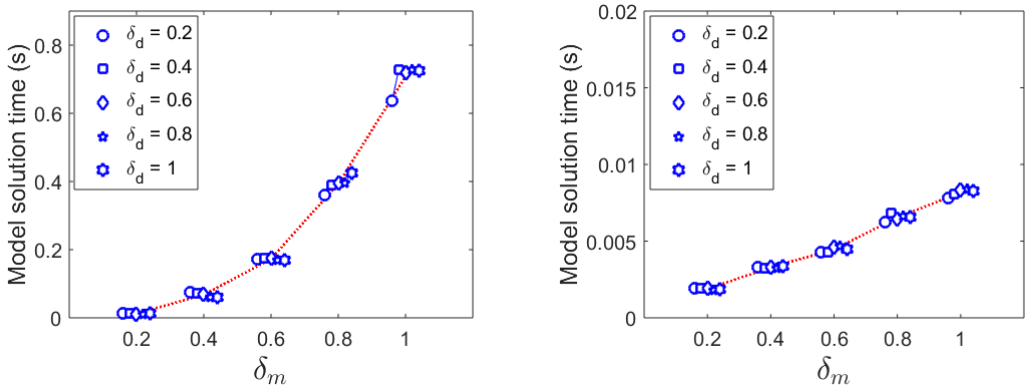
(a) $R^{instance}$ (b) $R^{cluster}$

(c) miFV (d) $D_{meanmin}$

Figure 4.7. Testing times of LP-MIL, miFV and $D_{meanmin}$ on Elephant dataset with changing values of δ_m and δ_d.

special type of MIL problem and other analogous MIL applications, we define additional weight coefficients. Our instance weighting scheme resembles to label learning under *weighted collective* assumption of MIL [21].

There are several ways of determining the membership of a bag such as taking the minimum, maximum or the average of the membership values of its instances. In [36], a subset of instances from positive bags and all instances from negative bags are separated by a hyperplane. MI-SVM [7] selects the most positive instance in positive bags and the least negative instance in negative bags. This idea is in alignment with the standard MI assumption in which positive instances belong to a single target concept. Taking the minimum and the maximum instance memberships are two extreme decision



(a) $\mathrm{R}^{\mathrm{instance}}$ + LP (b) $\mathrm{R}^{\mathrm{cluster}}$ + LP

Figure 4.8. Solution time of LP on representations $\mathrm{R}^{\mathrm{instance}}$ and $\mathrm{R}^{\mathrm{cluster}}$ of Elephant dataset with changing values of δ_m and δ_d.

alternatives to determine bag class estimate. In our proposals, membership estimates for bags are determined by averaging membership values of its possessed instances as $\beta_j = \frac{1}{n_j} \sum_{i \in I_j} m_i \quad \forall j \in J$.

Instance pseudo class membership value of each instance is weighted by the summation of distances between this instance and all other instances in its owner bag. The contribution of instances to the bag label is proportional with their distances to the other instances in the corresponding bag. Let $d(\mathbf{x}_i, \mathbf{x}_k)$ be the Euclidean distance between instance i and instance k. The weights of instances in a bag are normalized with the summation of all pairwise distances to sum up to one. Finally, the instance weights are calculated using the following equation

$$u_i = \frac{\sum_{k \in I_j} d(\mathbf{x}_i, \mathbf{x}_k)}{\sum_{i \in I_j} \sum_{k \in I_j} d(\mathbf{x}_i, \mathbf{x}_k)}, \qquad \forall i \in I_j, \forall j \in J. \tag{4.2}$$

The normalized weight u_i adjusts the contribution of instance i to the class membership estimate of its owner bag. Therefore, the alternative statement of Constraint (4.1c) can be written as $\beta_j = \sum_{i \in I_j} u_i m_i, \quad \forall j \in J$ where u_i is the normalized weight associated with instance i.

As an illustration, instances from 50 positive and 50 negative bags are selected from a compact cluster in a 2D feature space as shown in Figure 4.9. Each bag has 10 instances. Moreover, some instances from positive bags are selected from a perimeter which is located away from the compact cluster of all other instances. This example illustrates the situation in Newsgroups datasets [56] where instances in positive and negative bags share high level of similarity and a few instances in positive bags are identified as outliers. As mentioned earlier, these outlier instances are effective for classification. To tackle this MIL problem, we solved our LP model by weighting instance pseudo class memberships. The average of AUC values after 5 repeats of ten-fold cross validation runs is 89.5 when data is represented by using instance dissimilarities. For the non-weighted case, the resulting AUC value was 74.8.

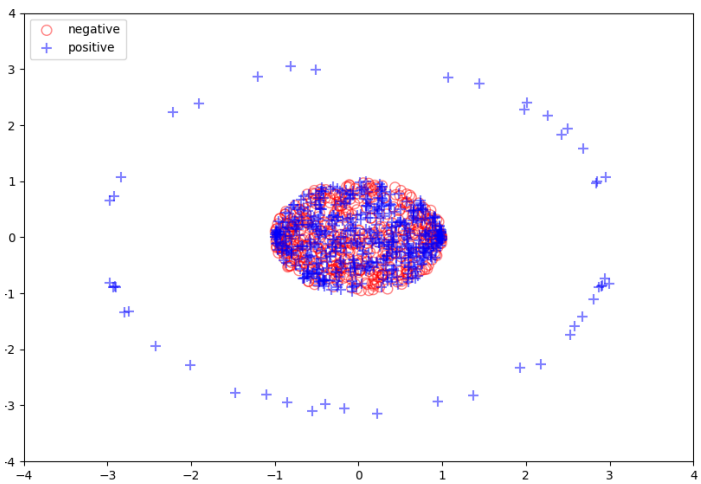


Figure 4.9. Scatter plot of instances from positive (blue) and negative (red) bags in 2D. Instances from positive and negative bags are similar to each other except for the instances located at the outer perimeter.

Table 4.2 shows the results of LP model including application of the weight function on Newsgroups datasets [56]. The best AUC results are marked with asterix. Table 4.2 demonstrates the improved classification success by non-equally weighting the instances in this MIL problem.

Table 4.2. AUC results of LP model on Newsgroups [56] datasets with and without instance membership weighting

Dataset	**Avg. of algorithm AUC (%)**			
	Unweighted		Weighted	
	LP + $R^{instance}$	LP + $R^{cluster}$	LP + $R^{instance}$	LP + $R^{cluster}$
Newsgroups 1, alt.atheism	47.0 (2.5)	66.8 (2.8)	73.2 (2.3)	89.5 (1.3)*
N.g. 2, comp.graphics	61.0 (2.3)	50.4 (3.0)	80.2 (1.7)*	49.9 (3.2)
N.g. 3, comp.os.ms-windows.misc	44.6 (2.8)	63.4 (2.5)	68.6 (2.1)	70.6 (2.8)*
N.g. 4, comp.sys.ibm.pc.hardware	53.0 (2.7)	56.5 (3.2)	71.0 (2.7)	72.2 (2.6)*
N.g. 5, comp.sys.mac.hardware	50.6 (2.2)	64.6 (3.2)	78.8 (1.9)*	69.0 (2.9)
N.g. 6, comp.windows.x	59.5 (2.6)	57.8 (2.8)	79.1 (2.3)*	69.6 (3.0)
N.g. 7, misc.forsale	53.5 (2.3)	56.9 (3.1)	66.5 (2.3)*	57.2 (3.3)
N.g. 8, rec.autos	48.5 (2.5)	43.0 (3.3)	72.5 (2.3)*	49.8 (3.7)
N.g. 9, rec.motorcycles	63.0 (2.8)	43.8 (2.7)	79.6 (2.2)	85.9 (1.7)*
N.g. 10, rec.sport.baseball	64.3 (2.4)	49.8 (3.0)	87.7 (1.9)*	52.2 (3.2)
N.g. 11, rec.sport.hockey	49.0 (2.5)	45.8 (3.2)	77.1 (1.9)*	42.0 (3.2)
N.g. 12, sci.crypt	52.2 (2.6)	55.5 (2.8)	69.3 (2.0)*	66.8 (2.5)
N.g. 13, sci.electronics	45.8 (2.1)	48.8 (4.0)	85.0 (1.6)*	64.2 (3.6)
N.g. 14, sci.med	61.2 (2.5)	46.8 (3.2)	76.5 (2.3)	84.2 (1.6)*
N.g. 15, sci.space	43.0 (2.3)	51.6 (3.1)	71.8 (2.1)	72.7 (3.0)*
N.g. 16, soc.religion.christian	41.6 (2.7)	43.7 (3.0)	75.0 (2.1)	80.6 (1.8)*
N.g. 17, talk.politics.guns	41.6 (2.7)	50.8 (2.8)	61.4 (2.4)*	58.0 (2.5)
N.g. 18, talk.politics.mideast	56.7 (2.5)	49.0 (3.1)	74.9 (2.0)*	42.6 (3.2)
N.g. 19, talk.politics.misc	51.5 (1.9)	50.8 (2.3)	67.0 (2.0)*	61.9 (2.5)
N.g. 20, talk.religion.misc	38.6 (2.3)	61.9 (2.7)	60.2 (2.7)	80.7 (2.5)*

4.4. Conclusions

We propose a multiple instance learning framework including a new mathematical model of multiple instance classification and enhanced data representations. We efficiently solve the MIL problem without imposing strict assumptions on object descriptions. Our approach embeds instance relationships via inputting various data representations and determines class memberships of the objects. To the best of our knowledge, this is the first linear programming based classification approach in MIL. We compare our learning procedure with state-of-the-art MIL methods on a wide range of machine learning datasets to highlight the classification success on different applica-

tion domains. Unlike the previous mathematical models of MIL, we do not force regular margin maximization. This leads to avoiding quadratic optimization, which is computationally more difficult than linear programming. Moreover, a common initialization setting of previous models is that all the instances in positive bags are positive and all the instances in negative bags are negative. This strong assumption is not required in our approach since we only calculate pseudo-class memberships of instances regardless of the class label of their owner bag. We also exploit different data representations to improve success of the linear classifier. Instance dissimilarity spaces are constructed to represent the input data to perform nonlinear separation. In datasets with large number of instances, it is computationally demanding to form the new instance-feature space. In order to reduce amount of distance calculations between pairs of instances, we employed data clustering. Instead of instance dissimilarities, distances to the centers of generated clusters are the new features. Finally, the internal structure of Newsgroups datasets is not captured successfully by LP-based MIL. Therefore, an instance weighting scheme is employed for each bag during calculation of bag-level class membership estimates.

In this chapter, linear programs are solved to perform MIL classification. Proposed mathematical models are efficient to solve on different input data representations. Processing the instance-level relationships and forming the bag label estimates using the instance-level scores deliver promising classification success on diversified real world MIL applications. As an extension, MIL can be used in large scale data mining applications requiring decentralized data storage. To decrease the solution times and considering the restrictions on data availability in such applications, subsets of the original data can be used to form a MIL classifier. Inspections on the potential loss in classification accuracy due to not being able to process whole data may give rise to a reformulation of the proposed model. A commonly seen property in optimization-based data mining approaches is overfitting. Both data representation and classifier generation processes may reinforce this situation. Potential overfitting problems on some MIL datasets can be recovered by using an ensemble formed by repeatedly solving mathematical models on different subsamples of the data.

5. MULTIPLE INSTANCE CLASSIFICATION VIA QUADRATIC PROGRAMMING

5.1. Introduction

Most data mining approaches focus on solving classification problems using machine learning and pattern recognition techniques. Classification tasks require input samples with given outputs, known as the class labels. In multiple instance learning (MIL), instances are grouped into bags and a class label is known for each bag, whereas the instance labels are not fully provided. The data representation and learning setup of MIL are in alignment with many real world applications. Current research areas of MIL include image classification, drug activity prediction, text mining and many others [107]. In these applications, global descriptions of the objects are decomposed into multiple parts. When objects are represented by multiple parts, only some parts may be relevant for classification. In addition, it is expensive and time consuming to collect true labels of parts individually. MIL paradigm provides an opportunity to solve classification problem under these circumstances.

For instance, consider sample images from Corel image classification dataset [8] in Figure 5.1. Under MIL scenario, images correspond to bags and patches sampled from the images correspond to the instances. In this example, images are classified either as positive or negative based on the presence of a horse on its patches as shown in Figure 5.1. Only some patches of an image are informative for classification and it is sufficient to label the whole image instead of the individual instances.

Unknown instance labels and uncertainty on the bag formations contribute to the difficulty of MIL problem. In Figure 5.2, a regular input data with 12 instances and 3 features is used to form a MIL data with 3 bags following the standard MIL assumption. Although it is embraced by many methods, standard assumption is considered to be restrictive for some MIL applications. For example, consider a document retrieval

Figure 5.1. An illustration of MIL setting for image classification. Images on the left with located horses inside the red rectangles are classified as positive whereas the other images form the negative class.

application, where the bags are articles and multiple sections extracted from them are the instances. The aim is to detect whether an article is about a specific subject (e.g. finance) or not. A section including the predetermined words and word combinations makes this section a positive instance. However, articles that are not relevant may also contain these words in a particular section (e.g. including financial terms in the introduction). Thus, standard MIL assumption is not well suited to this problem.

Generalized MIL [20–22] is formalized to describe MIL scenarios other than the standard MIL under various constraints [20]. Alternative ways of defining positive bags considering the bags as a whole are introduced in [21]. Collective MIL assumption [21] reveals a holistic MIL problem where each instance in a bag equally contributes to the bag label. The idea is to derive a bag-level classifier from an instance-level decision function by averaging the learning results in underlying instance-feature space.

Since there is an ambiguity on validity of the deemed MIL assumption in a MIL problem, prospective MIL methods are needed to be compatible with alternative MIL assumptions arising in different MIL scenarios. We regard collective MIL assumption in our proposed MIL method unlike the previous methods because of its modeling capability of the standard assumption and coverage on other MIL assumptions by

Instance Class	Instance ID	Feature 1	Feature 2	Feature 3
1	1	42	-198	-109
1	2	42	-191	-142
0	3	42	-191	-142
0	4	42	-198	-110
0	5	40	-39	30
0	6	36	-198	-166
0	7	43	69	30
0	8	36	-198	-166
0	9	24	80	29
0	10	38	-29	27
0	11	17	77	30
0	12	17	77	31

Regular

Bag Class	Bag ID	Feature 1	Feature 2	Feature 3
1	1	42	-198	-109
	1	42	-191	-142
	1	42	-191	-142
	1	42	-198	-110
0	2	40	-39	30
	2	36	-198	-166
	2	43	69	30
	2	36	-198	-166
0	3	24	80	29
	3	38	-29	27
	3	17	77	30
	3	17	77	31

Multiple Instance

Figure 5.2. Multiple instance data representation of one positive bag and 2 negative bags with 3 features.

means of the smooth average of instance-level decisions [108].

Previously, various data-mining and machine learning algorithms have been devised to solve the MIL problem. These approaches are heuristic algorithms and optimality of their solutions cannot be guaranteed. In this study, we focus on optimization-based approaches to solve MIL problem, and we refer the reader to comprehensive surveys [22, 107] for other categories of MIL methods.

Support vector machine (SVM) classification is a state-of-the-art discriminative approach for solving traditional supervised learning problems. Classic SVM formulations are quadratic or linear programs that maximize margin and minimize the classification error [62]. SVM classification is extended to MIL setting previously [4, 7, 33–36]. Table 5.1 describes and compares the Multiple Instance Support Vector Machine (MISVM) models in the literature. The level of the formulations indicates whether the misclassification penalties are incurred for bags or not. The assumptions are qualified as weak if only the standard MIL assumption holds. Otherwise, if there are additional restrictions reflected to the mathematical model, assumption status is entitled as strong.

In MISVM models, an instance is selected from a positive bag as a witness to represent that bag. Figure 5.3 illustrates standard SVM classification in instance space and bag-level separation. To classify bags, a witness instance is selected from a positive bag as shown in Figure 5.3. Witness instances are considered to be responsible from bag positivity and must be correctly classified by the MISVM solution.

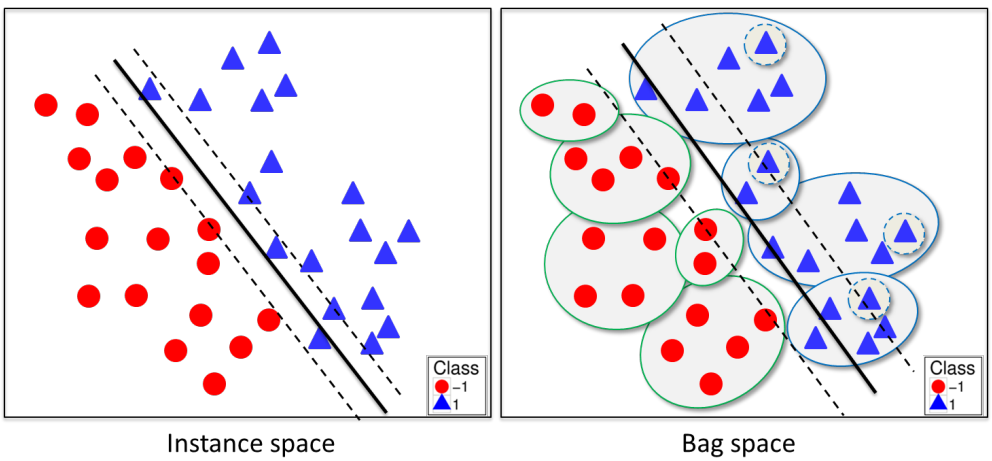


Figure 5.3. An illustration of witness selection in MISVM models. Red circles indicate instances in negative bags. In positive bags, instances are represented with blue triangles and witness instances are enclosed in dashed circles.

In mi-SVM and MI-SVM formulations [7], two types of constraints are added to the SVM formulation satisfying at least one sample in each positive bag has a label of one in mi-SVM and a witness instance is present for positive bags in MI-SVM. MissSVM [4] is formulated upon MI-SVM [7] with additional constraints on the positive bags. Minimizing the misclassification error at either extreme, an instance of a positive bag is either positively or negatively labeled. Another method KI-SVM [33] selects witnesses from positive bags as key instances. Since their witness selection scheme incurs exponential number of constraints, the authors employ a cutting plane algorithm for model solution.

Sparse transductive MIL formulation (stMIL) [32] has an additional constraint that pulls all the negative instances in the bag closer to the hyperplane. The authors propose a concave convex procedure (CCCP) to solve the non-convex formulation. An ℓ_1-norm SVM-based formulation [34] incorporating the assumption "arbitrary convex

combination of instances in the positive bags represents each positive bag" is proposed which is a linear program with bilinear constraints. Their solution approach is a multiple instance classification algorithm with a nonlinear kernel (MICA), which holds the bilinear terms constant while solving the linear programs iteratively. MICA guarantees convergence to a local optimal solution.

MIL problem is formulated as a mixed $0-1$ quadratic programming problem in [35], where MIL is reduced to instance-level learning and only the standard MIL assumption is taken into account. A branch and bound heuristic, which gives tight upper bounds is presented. However, the formulation disregards the bag information, therefore the classification performance is degraded.

In [36], SVM formulations of MIL problem are derived as a hard margin maximization formulation, MIHMSVM, and a soft margin maximization formulation, MIHLSVM with additional bag-level misclassification penalties. A penalty is incurred if all instances in a positive bag are misclassified or at least one instance in the negative bag is misclassified. The resulting formulations performing witness selection are mixed integer quadratic programs (MIQPs), which are known to be NP-hard problems [35].

Most of the aforementioned MISVM models are analyzed in a recent survey [37]. It is emphasized in [37] that local convergence of the heuristic solution approaches for solving non-convex MISVM formulations leads to a sacrifice from the classification performance. The authors also discuss scalability of MISVM methods: Increased number of instances and bags affect model dimensionality and therefore increase both the hyperparameter selection and model solution times.

Classical SVM formulations for supervised classification form large-scale quadratic programs that have long training times. Instead of standard solvers, SVM formulations are solved using specifically devised SVM solvers [109]. However, when SVMs are tailored for MIL, these solvers can only be used solving subproblems of various heuristic solution algorithms [4,7,33–35,110]. As we have noted, extra constraints regarding MIL setting are involved in the models and complicate the solution process. We, therefore,

propose a simplified quadratic programming (QP) formulation for MIL classification, which can be directly solved to optimality using any commercial QP solver.

In addition to aforementioned drawbacks in [37], MISVM methods in Table 5.1 rely on standard MIL assumption and may fail to model generalized MIL problems. The binary variables associated with witness selection and nonlinear constraints due to the maximum operator complicate the MISVM models. Hence, they are unlikely to be solved to optimality in reasonable amount of time. Convergence of heuristic algorithms to local minima has unfavorable influence on the solution quality and hence on the correctness of the resulting classifier.

Table 5.1. The comparison of MIL formulations.

Formulation	Model type	Level	Assumptions	Solution approach	Main reference
mi-SVM	MIQP	instance	weak	mi-SVM opt. heuristic	[7]
MI-SVM	MIQP	bag	weak	MI-SVM opt. heuristic	[7]
stMIL	NC-MINLP	instance	strong	CCCP	[32]
MissSVM	NLP	instance	strong	CCCP	[4]
ℓ_1-norm SVM-MIL	NLP	instance	strong	MICA	[34]
KI-SVM	MIQP	instance	strong	Cutting plane algorithm	[33]
Max-Margin MIL	0-1 MIQP	instance	weak	Branch and bound	[35]
MIHMSVM	MIQP	instance	strong	Three-phase heuristic	[36]
MIHLSVM	MIQP	bag	strong	Exact	[36]

Motivated by the above discussions on SVM-based MIL methods, we propose a novel Quadratic Programming-based Multiple Instance Learning (QP-MIL) framework [111]. Our proposal is based on the idea of determining a single threshold value for discriminating positive and negative bag classes. We model MIL problem as a QP problem using the input data representation.

An optimal solution of our QP formulation returns an instance-level scoring function together with a bag-level classification threshold. For an unlabeled bag, instance-level scores are averaged to assess the bag-level score. Finally, class label of the bag is determined according to the predetermined threshold value. Rather than selecting bag representatives as in standard MIL, QP-MIL processes the set of entire instances to out-

put a simple bag classifier. This setting is also applicable to various MIL applications which are referred to as generalized MIL.

A common initialization property of the MISVM formulations is that each instance shares the same label with its owner bag. However, only a few instances in positive bags are informative for classification. Elimination of additional assumptions on relating instance labels to bag labels releases compact formulations to solve MIL problem, which are easy to optimize by standard QP solvers. Instead of utilizing an iterative heuristic procedure, we are able to report exact solutions of each problem instance. Thus, repetition of the performed classification task is possible and the resulting classifier is reproducible in this way.

Our study explores the utility of QP-MIL compared to the previous state-of-the-art MIL approaches. Leading methods in MIL literature are various machine learning-based approaches. We select several MIL algorithms as baseline methods to demonstrate success of the MIL classifiers. Inspired by existing mathematical models in the MIL literature, we carry out another comparison of QP-MIL considering SVM-based MIL, in terms of model building and classifier testing. A mixed integer quadratic programming (MIQP) formulation proposed in [36] enables robust representation of outliers to perform a generalizable classification. Since direct solution of this formulation requires considerable solution time especially for larger datasets, we also provide a generalized Benders decomposition (GBD) [112] algorithm for comparison.

The remainder of the chapter is organized as follows: Section 5.2 describes the proposed QP-MIL framework. Section 5.3 provides insights resulting from the numerical comparisons of QP-MIL with MIHLSVM and presents the classification success and computational efficacy of QP-MIL with the experiments on a wide-range of MIL datasets. Conclusions and future extensions are discussed in Section 5.4.

5.2. Quadratic Programming for Multiple Instance Learning

5.2.1. A Novel QP Formulation for MIL

The sets, parameters and decision variables used in the model are given as follows.

Indices:

$i = 1, 2, \ldots, n$: index for the instances

$j = 1, 2, \ldots, m$: index for the bags

Sets:

$J^+ = \{j : l_j = 1\}$: set of positive bags

$J^- = \{j : l_j = -1\}$: set of negative bags

$J = J^+ \cup J^-$: set of all bags

$I^+ = \{i : i \in I_j \wedge j \in J^+\}$: set of instances in positive bags

$I^- = \{i : i \in I_j \wedge j \in J^-\}$: set of instances in negative bags

$I = I^+ \cup I^-$: set of all instances

Parameters:

$\mathbf{x}_i \in \Re^d$, $i = 1, 2, ..., n$: instance vectors

l_j: bag labels

Decision variables:

$\mathbf{w}$: d-dimensional feature weight vector

m_i, $i = 1, 2, \ldots, n$: instance pseudo class memberships

β_j, $j = 1, 2, \ldots, m$: bag class memberships

δ_j^+, δ_j^-: slack variables for the positive and negative bag deviations

τ: decision threshold for bag classification

A bag classification rule along with a decision threshold can be found by solving the following optimization model:

$$\text{(QP)} \quad \min_{\mathbf{w},\boldsymbol{\beta},\mathbf{m},\tau,\boldsymbol{\delta}^+,\boldsymbol{\delta}^-} \quad \frac{1}{2}||\mathbf{w}||^2 - C\left(\frac{1}{m^+}\sum_{j\in J^+}\delta_j^+ + \frac{1}{m^-}\sum_{j\in J^-}\delta_j^-\right) \tag{5.1a}$$

$$\text{s.t.} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle = m_i \quad \forall i \in I \tag{5.1b}$$

$$\beta_j = \frac{1}{n_j}\sum_{i\in I_j} m_i \quad \forall j \in J \tag{5.1c}$$

$$\beta_j \geq \tau + \delta_j^+ \quad \forall j \in J^+ \tag{5.1d}$$

$$\beta_j \leq \tau - \delta_j^- - \varepsilon \quad \forall j \in J^- \tag{5.1e}$$

$$0 \leq m_i \leq 1 \quad \forall i \in I \tag{5.1f}$$

$$0 \leq \delta_j^+ \leq 1 \quad \forall j \in J^+ \tag{5.1g}$$

$$0 \leq \delta_j^- \leq 1 \quad \forall j \in J^- \tag{5.1h}$$

$$0 \leq \tau \leq 1 \tag{5.1i}$$

The quadratic objective function (5.1a) performs maximization of bag class membership margin together with a regularization of feature weights. The second term of the objective function (5.1a) maximizes the margin of bag class estimates formed by the threshold variable. In order to handle potential problems due to class imbalances, summations of the nonzero slack variables δ_j^+, $\forall j \in J^+$ and δ_j^-, $\forall j \in J^-$ in the objective function (5.1a) are normalized with the number of positive bags m^+, and the number of negative bags m^-, respectively.

Constraint (5.1b) determines instance pseudo class memberships $m_i, \forall i = 1, \ldots, n$ using the coefficient vector $\mathbf{w}$ entry of which corresponds to the weight assigned to a feature of the input data. Constraint (5.1c) maps bag-level class estimates $\beta_j, \forall j = 1, \ldots, m$ onto the $[0, 1]$ interval by averaging instance-level scores, which are forced to be between 0 and 1 by Constraint (5.1f). Constraints (5.1d) and (5.1e) are the threshold determination constraints ensuring that absolute difference between class membership estimate β_j and the threshold τ are maximized in the objective function for both positive and negative bags. Constraint (5.1i) restricts the decision threshold

τ to be between 0 and 1. We set ε in Equation (5.1e) to a small positive value (10^{-6}) so that class membership value of a negative bag is strictly below the threshold τ.

Regularization processes are introduced to supervised learning problems for recovering the important features and for satisfying model generalizability. The discriminative features are designated by optimizing the weight coefficients $\mathbf{w}$. In the first term of the objective function (5.1a), standard ℓ_2-norm of the weight coefficients $\mathbf{w}$ are minimized. Therefore, effect of redundant or uninformative features can also be controlled. The hyperparameter C in the objective function (5.1a) tunes the trade-off between regularization of $\mathbf{w}$ and maximization of bag class membership estimate margin.

QP-MIL models the contributions of all instances in a bag to the bag label collectively. Averages of pseudo-class membership estimates for instances determine the class membership estimates for the bags. We conduct a thresholding scheme to determine the class labels regarding the bag-level estimates. A bag is positively labeled if its class membership value is above decision threshold τ, and negatively labeled otherwise. An optimal value of τ is adaptively identified in QP-MIL during the optimization process. This threshold is also applicable to the test bags. After solving the QP formulation in (5.1) on the training set, instance scores are calculated by Equation (5.1b) for each instance in a test bag and simply averaged in Equation (5.1c) to compute the bag-level score. If the output is below the optimal value of τ, the classifier produces a negative label, else a positive label.

The resulting bag-level classifier can be defined as

$$g(B_j) = \begin{cases} 1 & \text{if} \quad \beta_j \geq \tau, \\ -1 & \text{otherwise} \end{cases}$$

where

$$\beta_j = \frac{1}{n_j} \sum_{i \in I_j} m_i$$

and

$$m_i = \langle \mathbf{w}, \mathbf{x}_i \rangle \quad \forall i \in I_j.$$

Our proposed MIL framework is independent of the underlying MIL assumptions. We seek to model bag structures by taking into account the reflection of instance scores to the bag labels. Since all instances contribute to the bag-level scoring, this paradigm resembles the collective MIL assumption [21]. It is shown in [108] that if an instance level separation can be performed in an embedding space $\mathcal{H}$ with a classifier f in a standard MIL problem, then the bags can also be separated in another embedding space $\mathcal{H}'$, which has a higher dimensionality than $\mathcal{H}$, by scoring each bag with the average of its instance-level estimates as $g(B_j) = \frac{1}{n_j} \sum_{i \in I_j} f(\mathbf{x}_i)$. Therefore, various MIL assumptions can be handled with a proper data representation and collective modeling of the bag structures.

In order to perform class separation by correct classification, having class membership values above the threshold for positive bags and below the threshold for negative bags is desirable. Therefore, we maximize summation of the absolute differences between bag class membership estimates. This paradigm defines the margin between positive and negative class membership estimates, as well. Thus, optimal value of decision threshold τ leaves the maximum margin between bag class membership estimates. Figure 5.4 illustrates a possible solution to the QP model (5.1). The selected value for decision threshold τ is 0.55 and the class memberships estimates for three positive and three negative bags are consistent with this threshold.

5.2.2. A Previous MIQP Formulation: MIHLSVM [36]

Multiple Instance Hinge Loss Support Vector Machines (MIHLSVM) [36] extends traditional SVM for MIL. Unlike earlier SVM-based approaches to MIL, MIHLSVM

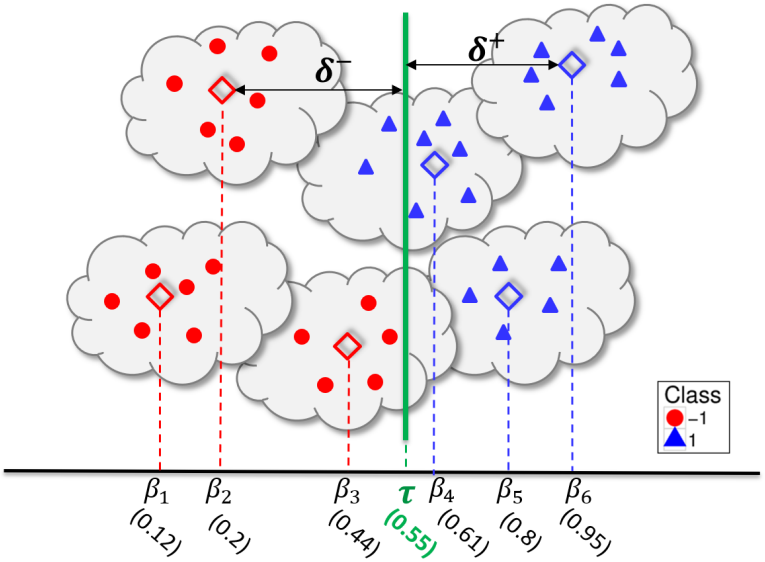


Figure 5.4. An illustration of a solution to QP model (5.1). Instance level scores are symbolized with red circles and blue triangles, for negative and positive bags, respectively. The vertical green line indicates the decision threshold and each dashed line maps bag class membership of a bag. For a positive and a negative bag, class membership margins are indicated with horizontal arrows.

defines bag-level hinge loss to penalize bag misclassifications. The proposed model handles the situation of nonlinearly separable classes and the resulting formulation is a MIQP. The authors propose direct solution of MIHLSVM in [36] and do not present a heuristic algorithm as those in other MISVM studies [4, 7, 33–35]. Still, it is difficult to get an exact solution to a MIHLSVM problem instance. MIHLSVM formulation is considered to be investigated both in modeling and solution aspects for comparison purposes. We devise a generalized Benders decomposition scheme for MIHLSVM in Section 5.2.3. We present our comparisons in Section 5.3.4.1 by solving both the original and the decomposed version of this problem.

A MIQP formulation of the described problem is given in [36] as below

$$\text{(MIHLSVM)} \quad \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^+,\boldsymbol{\xi}^-,\boldsymbol{\eta},\boldsymbol{z}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\left(\sum_{j\in J^-}\xi_j^- + \sum_{j\in J^+}\xi_j^+\right) \tag{5.2a}$$

$$\text{st} \quad -(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i \quad \forall i \in I^- \tag{5.2b}$$

$$\langle \mathbf{w}, \mathbf{x}_i\rangle + b \geq 1 - \xi_i \quad \forall i \in I^+ \tag{5.2c}$$

$$\sum_{i\in I_j}\eta_i = 1 \quad \forall j \in J^+ \tag{5.2d}$$

$$\xi_i \leq \xi_j^- \quad \forall j \in J^-, \forall i \in I_j \tag{5.2e}$$

$$\xi_j^+ = \sum_{i\in I_j} z_i \quad \forall j \in J^+ \tag{5.2f}$$

$$z_i \geq \xi_i - M(1-\eta_i) \quad \forall i \in I^+ \tag{5.2g}$$

$$z_i \leq \xi_i \quad \forall i \in I^+ \tag{5.2h}$$

$$z_i \leq M\eta_i \quad \forall i \in I^+ \tag{5.2i}$$

$$z_i \geq 0 \quad \forall i \in I^+ \tag{5.2j}$$

$$\xi_i \geq 0 \quad \forall i \in I \tag{5.2k}$$

$$\eta_i \in \{0,1\} \quad \forall i \in I^+. \tag{5.2l}$$

In addition to maximization of the margin between bag classes, the objective function (5.2a) also minimizes bag misclassifications where a selected constant C controls the trade-off between two objectives. Constraints (5.2b) and (5.2c) are margin constraints with the slack variables for penalization of misclassification using slack variables ξ_i for misclassified instances. The weight vector $\mathbf{w}$ and the offset parameter b defines the instance-level separating hyperplane. Constraint (5.2d) forces a positive bag to have a positive instance as a witness. Negative bag misclassifications are represented by constraint (5.2e) using slack variables $\xi_j^-, \forall j \in J^-$. It is assumed that a negative bag is misclassified if all of its instances are misclassified. Constraint (5.2l) imposes binary restrictions on witness variables and nonnegativity restrictions on slack variables are introduced by constraint (5.2k). Constraints (5.2g)–(5.2i) with the auxiliary variables $z_i \geq 0, \forall i \in I^+$, determine the positive bag misclassification of a bag as

the misclassification of its selected witness instance.

After solving MIQP formulation, the following classifier can be used for bag classification

$$\operatorname{sgn}\left(\max_{i\in I_j}\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right)\right), \qquad j \in J. \tag{5.3}$$

5.2.3. A Generalized Benders Decomposition Approach to MIHLSVM

Benders decomposition algorithm [113] provides the opportunity of solving large-scale problems especially with binary or integer variables, efficiently. The mechanism of the algorithm is based on solving a master problem for a subset of variables including the complicating variables and then solving the well-known structured subproblem for the remaining subset of variables where the values of complicating variables are fixed. By using LP duality in the subproblem, the Benders cut is obtained and added to the master problem. To decompose nonlinear convex programming problems, Generalized Benders Decomposition (GBD) [112] is developed. Its main difference from original Benders decomposition is that the subproblem is a nonlinear program. Instead of LP duality, the Benders cut is obtained by using nonlinear duality after solving the subproblem.

The mathematical formulation of the standard SVM classifier has a quadratic objective function with linear constraints and continuous variables. MIHLSVM has additional binary variable set η and its objective function is also quadratic with some linear constraints. Hence, the formulation is a nonlinear mixed integer programming formulation and GBD is a suitable method for this type of mathematical models. Once we know the set of binary variables η, the remaining problem is a convex optimization problem. To obtain the generalized Benders cut to be added to relaxed problem, the following subproblem (SP) is solved:

$$
\begin{aligned}
(\text{SP}) \qquad \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^+,\boldsymbol{\xi}^-,\mathbf{z}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C(\sum_{j\in J^-}\xi_j^- + \sum_{j\in J^+}\xi_j^+) && \text{(5.4a)}\\
\text{st} \quad & -(\langle \mathbf{w},\mathbf{x}_i\rangle + b) \geq 1-\xi_i \quad \forall i\in I^- && \text{(5.4b)}\\
& \langle \mathbf{w},\mathbf{x}_i\rangle + b \geq 1-\xi_i \quad \forall i\in I^+ && \text{(5.4c)}\\
& \xi_j^+ = \sum_{i\in I_j} z_i \quad \forall j\in J^+ && \text{(5.4d)}\\
& z_i \geq \xi_i - M(1-\eta_i) \quad \forall i\in I^+ && \text{(5.4e)}\\
& z_i \leq \xi_i \quad \forall i\in I^+ && \text{(5.4f)}\\
& z_i \leq M\eta_i \quad \forall i\in I^+ && \text{(5.4g)}\\
& \xi_i \leq \xi_j^- \quad \forall j\in J^-, \forall i\in I_j && \text{(5.4h)}\\
& z_i \geq 0 \quad \forall i\in I^+ && \text{(5.4i)}\\
& \xi_i \geq 0 \quad \forall i\in I^-. && \text{(5.4j)}
\end{aligned}
$$

The Lagrangean dual can be written as follows:

$$
\begin{aligned}
\Phi(\boldsymbol{\eta}) = \max_{\substack{\boldsymbol{\theta}\geq 0,\\ \boldsymbol{\rho}\geq 0,\boldsymbol{\mu}\geq 0,\boldsymbol{\nu}\geq 0,\\ \boldsymbol{\pi}\geq 0,\boldsymbol{\lambda}\geq 0}} \Bigg[\min_{\substack{\mathbf{w},b,\boldsymbol{\xi},\\ \boldsymbol{\xi}^+,\boldsymbol{\xi}^-,\mathbf{z}}} \Bigg[& \frac{1}{2}||\mathbf{w}||^2 + C\left(\sum_{j\in J^-}\xi_j^- + \sum_{j\in J^+}\xi_j^+\right)\\
& + \sum_{i\in I^-}\theta_i\left(1+\langle \mathbf{w},\mathbf{x}_i\rangle + b - \xi_i\right) + \sum_{i\in I^+}\rho_i\left(1-\langle \mathbf{w},\mathbf{x}_i\rangle - b - \xi_i\right)\\
& + \sum_{i\in I^+}\mu_i\left(-M(1-\eta_i)+\xi_i - z_i\right) + \sum_{i\in I^+}\nu_i\left(z_i - M\eta_i\right)\\
& + \sum_{i\in I^+}\pi_i\left(z_i-\xi_i\right) + \sum_{j\in J^-}\sum_{i\in I_j}\lambda_{ij}\left(\xi_i-\xi_j^-\right)\Bigg]\Bigg].
\end{aligned}
\tag{5.5}
$$

Note that it is unnecessary to add Benders feasibility cuts since SP always has a feasible solution during iterations. Next, the following generalized Benders optimality cut is

obtained.

$$y \geq \Phi(\boldsymbol{\eta}^k) - \sum_{i \in I^+} \left(-Mu_i^k + Mv_i^k\right)\left(\eta_i - \eta_i^k\right) \tag{5.6}$$

After determination of the Benders cut, the following relaxed MIHLSVM model is formulated by adding K many Benders cuts as the following Master Problem (MP):

$$\begin{aligned}
(\text{MP}) \quad & \min_{v,\eta} \quad v && && (5.7a)\\
& \text{st} \quad v \geq \Phi(\eta^k) - \sum_{i \in I^+} \left(-Mu_i^k + Mv_i^k\right)\left(\eta_i - \eta_i^k\right) && \forall k = 1, ..., K && (5.7b)\\
& \sum_{i \in I_j} \eta_i = 1 && \forall j \in J^+ && (5.7c)\\
& \eta_i \in \{0, 1\} && \forall i \in I^+. && (5.7d)
\end{aligned}$$

We know that the MIHLSVM formulation given in (5.2) is a mixed integer quadratic program, and therefore, can be solved directly by commercial MIQP solvers. The efficiency of these two approaches along with QP-MIL are compared in Section 5.3.4.1 to verify the modeling and solution quality of the proposed MIL framework.

5.3. Experiments and Results

5.3.1. Data Representation

In MIL, a specific data region representing the positive instance class is named as a *concept*. The concept instances are informative for class discrimination. Based on this idea, representative sets can be derived in many ways as prototypes to capture the informative instance relationships. Several MIL methods benefit from the dissimilarities to selected prototypes to represent the bags [8, 26–29]. Moreover, a number of similar algorithms [49, 95] utilize clustering to learn a target concept in MIL problems.

Inspired by success of aforementioned methods, we attempt to perform MIL classification in a newly represented feature space. QP model (5.1) produces a linear classifier and success of this classifier is limited only to linearly separable data.

In QP-MIL, the relationships between instances can be implicitly modeled by preprocessing the input data. Instead of building a classifier in the original instance feature space, we attempt to represent the instances using dissimilarities to the selected prototypes. Aim of the representation is building a linear classifier, which is capable of class separation in a different space. We pool instances in bags and then group them by k-means clustering algorithm into an appropriate number of clusters. Then, the cluster centers are taken as the prototypes. The new features are simply constructed by calculating the Euclidean distances of each instance to these cluster centers. This way, protoypes are derived as a summarized representation of the original data and the linear classifier becomes applicable to the new features.

5.3.2. Multiple Instance Datasets

We evaluate our approach in image classification, molecular activity prediction, text categorization and audio classification tasks. The datasets are categorized in Table 3.1 based on their application domain and the dataset characteristics are also provided. Datasets differ in terms of number of bags, instances and features. The first category includes famous drug activity prediction tasks on Musks and Mutagenesis' datasets and a protein identification task. Image classification datasets constitute the second category containing the Corel image datasets, UCSB breast cancer dataset and other smaller sized benchmarks Elephant, Fox and Tiger. Positive class is considered as the target images and the remaining images determine the negative bag class.

Another dataset category covers web mining tasks on Newsgroups and Web recommendation datasets. In Newsgroups, blog posts are categorized into 20 groups based on their subjects where a bag is formed by a collection of multiple posts (i.e. the instances). In the positive class, the terms about a specific subject appears in a number of posts, and the bags with posts about other subjects constitute the negative bags.

In Web recommendation, a web page in the user history is a bag and the web pages linked to that web page are the instances. Recommendations of a specific user form the positive class and the bags constituted by the remaining eight users are negatively labeled.

The last category is the bird song recordings from 13 different classes of birds, where a recording is bag and segments of recording are the instances. The target bird class is considered as positive, whereas the bags from the other classes are labeled as negative. We follow an effective experimentation strategy on a wide range of MIL datasets. Cross validation folds are generated by splitting the original dataset into the training set and the test set. We utilize the same splitting indices across both our proposed and the state-of-the-art methods from the literature to perform a comprehensive comparison. All the datasets and cross validation indices are available online at [63].

5.3.3. Experimental Setup and Evaluation Criteria

Our experiments use a Windows 10 PC with 16 GB RAM, dual core CPU (Intel Core i7-7700HQ 2.8 GHz). For each dataset, a stratified cross validation scheme is conducted to assess the generalizability of the classifiers. Initially, we scale each feature to zero mean and unit variance. We obtain data representations in QP-MIL via the implementation in Python that uses scikit-learn [79] library. We model QP formulations using Gurobi Python interface and solve using barrier QP solver of Gurobi 8.0 [102]. The default parameters are accepted for the barrier algorithm except for the convergence tolerance, which is set to 0.01.

QP-MIL has two parameters: number of clusters, κ in data representation and cost parameter C of QP model (5.1). In k-means clustering, necessarily enough number of clusters, κ is determined by using elbow approach [114]. Briefly, within cluster variance after k-means clustering is plotted along with increasing values of κ and the position of the elbow is identified to assign the corresponding value to κ. We run a nested cross-validation with an inner cross-validation loop to choose hyperparameter C from the set $\{0.01, 0.1, 1, 10, 100, 1000\}$. All of the instances of MIHLSVM and

GBD-MIHLSVM formulations are also executed using Gurobi 8.0 [102].

The baseline MIL approaches selected for comparison are MILES [8], MInD [26] with bag dissimilarity representation $D_{meanmin}$ and miFV [50]. MILES iteratively measures similarities of bags to the training instances, and builds a linear SVM classifier along with ℓ_1-norm regularization at the same time. MInD defines a bag-level feature representation by using the bag-to-bag dissimilarity measure $D_{meanmin}$. miFV benefits from Fisher vectorial coding to map each bag to a single vector. Both MInD and miFV build a linear SVM classifier to classify bag vectors.

We execute MILES [8] and MInD [26] using the MIL toolbox [84], and use a MATLAB [85] implementation to run miFV [50]. We accept the default parameters in the original paper for MILES [8]. We use the parameter setting proposed in [26] for MInD [26]. Following the authors' advice, we employ an inner ten-fold cross-validation to select the three parameters of miFV [50], which are enumerated as PCA energy, number of components and cost parameter of linear SVM. PCA energy attains values from the set $\{0.8, 0.9, 1\}$. The alternatives for the number of Gaussian components is selected from $\{1, 2, 3, 4, 5\}$. The cost parameter of the linear SVM classifier are $\{0.05, 1, 10\}$.

A receiver operation characteristics (ROC) curve visualizes the trade-off between percentage of true positive predictions and percentage of false positive predictions. The x and y axes in ROC curve plot range from 0 to 1. Area under the ROC curve (AUC) can also be used to compare classifiers and asserted to be a reliable metric for classification [115]. Larger AUC values indicate a better classifier. A random classification produces an AUC value close to 0.5, whereas a perfect classification leads to an AUC value of 1.

Another measure for classifier performance in MIL problems is classification accuracy. For a specific decision threshold value, such as the value of τ in QP-MIL after optimization, the bag classes are predicted and the accuracy of the classifier is computed. The class imbalance problem is seen in MIL tasks such as Corel, Web

recommendation and Birds benchmarks. The value of τ is optimized on the training bags, and suffers from misleading accuracy when the bag classes are imbalanced. AUC is more effective under class imbalance since all possible thresholds are evaluated to report the classifier performance. Additionally, given the consistent performance of AUC on MIL datasets [81], we qualify AUC as a primary comparison metric in our study.

5.3.4. Results

5.3.4.1. Comparison of QP-MIL with MIHLSVM. In this section, we present a comparison between QP-MIL and MIHLSVM formulation given in Section 5.2 in terms of computational efficiency and other indicators related to classification performance of the derived solutions. In addition to MIHLSVM model, GBD-MIHLSVM approach devised in Section 5.2.3 is considered for comparison. The clustering-based data representation described in Section 5.3.1 is considered as the input of all compared formulations.

Table 5.2 presents the overview of problem sizes on four moderate sized MIL datasets. All datasets are modelled using QP-MIL formulation in (5.1) and MIHLSVM formulation in (5.2). For each dataset in Table 5.2, ten separate models of QP-MIL and MIHLSVM are built, where ten different partitioning of the original dataset form the input in each model. The averages of problem dimension properties for ten models are reported in Table 5.2. Formulations in (5.1) and (5.2) have quadratic objective functions and number of the quadratic terms are equal for both. Since we solve the formulations on a cluster center-based data representation, the number of quadratic terms is equal to the dimensionality of this representation.

Problem size of MIHLSVM differs significantly from QP-MIL in terms of number of constraints and continuous variables. In addition, there are extra binary variables in MIHLSVM problem instances in proportion with the number of instances in positive bags due to the witness selection. Motivated by the large problem sizes of MIHLSVM even on moderated sized MIL datasets, we propose a generalized Benders decomposi-

Table 5.2. Model size summary of QP-MIL and MIHLSVM on problem instances of 4 datasets. 10 models of each formulation are built for each dataset, and the average values are reported.

				QP-MIL				MIHLSVM			
# / Dataset	Bags	Instances	Features	Constraints	Cont. variables	Binary variables	Quad. terms	Constraints	Cont. variables	Binary variables	Quad. terms
Elephant	200	1391	230	1611.9	1881.4	0	267.5	4055.4	2952.3	1251.9	267.5
Fox	200	1302	230	1548.0	1827.5	0	277.5	3720.6	2834.5	1188.0	277.5
Musk 1	92	476	166	594.0	816.0	0	220.0	1314.0	1160.6	428.4	220.0
Musk 2	102	6598	166	6121.8	6381.3	0	257.5	13777.2	12226.7	5938.2	257.5

tion algorithm discussed in Section 5.2.3 to observe the effect of solving smaller sized problems in MIL experiments.

In Table 5.3, we compare the performance of QP-MIL with the MIHLSVM. In addition, experiments on the GBD approach to MIHLSVM described in Section 5.2.3 are reported in Table 5.3. MIHLSVM is an MIQP and can be directly solved by standard MIQP solvers. We solve the MIHLSVM formulation in (5.2) and set the cost parameter C in the objective function (5.2a) to 1. It is plausible to tune up the appropriate value for C by a cross-validation procedure. However, the computation time of parameter selection in MIHLSVM is a limitation [36].

We are unable to report overall results for MIHLSVM since each cross-validation fold lasts longer than one day for relatively small datasets such as Elephant and Fox. Therefore, we do not carry out a cross-validation loop, and manifest only the model solution time for $C = 1$. In contrast with the described procedure in Section 5.2, we do not embed parameter selection into QP-MIL during comparisons of this section and the predetermined value of C is 1. The results in Table 5.3 are based on one repeat of a ten-fold cross validation.

All methods are executed within a time limit of 1800 seconds. First column is the number of problem instance from each dataset that is solved to optimality until the time limit is reached. The mean percentage optimality gap [(upper bound- lower bound)/upper bound] is reported for each algorithm and the corresponding average model solution time in seconds is also presented. To observe generalizability of the

learner, we evaluate obtained solutions on the test bags. Average accuracy and AUC values over ten experiments are reported for all three approaches.

Table 5.3. Comparison of QP-MIL, MIHLSVM and MIHLSVM-GBD on problem instances of 4 datasets. 10 models of each formulation are built for each dataset, and the average values are reported.

	QP-MIL				
Dataset	Solved	Gap	Time	AUC	Accuracy
Elephant	10	0	1.6	93.7	83.5
Fox	10	0	1.5	70.3	65.0
Musk 1	10	0	0.3	96.5	89.0
Musk 2	10	0	3.0	94.7	88.3
	MIHLSVM				
Dataset	Solved	Gap	Time	AUC	Accuracy
Elephant	0	97.6	1800	87.3	63.5
Fox	0	98.6	1800	64.9	55.0
Musk 1	1	37.1	1721.4	89.3	71.7
Musk 2	0	91.0	1800	90.8	74.4
	MIHLSVM-GBD				
Dataset	Solved	Gap	Time	AUC	Accuracy
Elephant	0	100	1800	87.3	77.5
Fox	0	100	1800	62.6	57.0
Musk 1	0	100	1800	90.5	81.3
Musk 2	0	100	1800	94.3	86.3

Computational study demonstrates that QP-MIL is significantly more efficient and provides accurate solutions compared to the MIHLSVM formulation. All instances of QP-MIL can be solved exactly without a sacrifice in classification success as demonstrated by AUC and accuracy results in Table 5.3. Being the largest dataset in this comparison, Musk 2 requires an average solution time of 3 seconds to solve QP model (5.1) to optimality. On the other hand, only one MIHLSVM instance of Musk 1 dataset can be solved to optimality within the time limit. Except for Musk 1, Gurobi is unable to reduce the optimality gap below 90%.

Moreover, GBD approach does not provide gain in computation time compared to directly solving MIHLSVM formulation. However, acquired solutions of GBD-MIHLSVM implementation led to an increase in accuracy. Although the corresponding optimality gaps are lower than GBD-MIHLSVM, solution quality of MIHLSVM is degraded in terms of accuracy for all four datasets, concurrently with AUC values for Musk 1 and Musk 2. This observation is a strong evidence of the overfitting problem, which is frequently observed in classification tasks. Namely, the output classifier is successful on training set but fails to produce generalizable results on testing data. For the sake of fairness, we do not include MIHLSVM and GBD-MIHLSVM in the overall comparison results in Section 5.3.4.2 due to the requirements of a higher runtime even for small/moderate sized datasets.

5.3.4.2. Comparison to Baseline Methods. Table 5.4 summarizes the performance of our proposed QP-MIL approach with MILES [8], MInD [26] with bag dissimilarity representation $D_{meanmin}$ and miFV [50] on four different MIL application categories. Their descriptions and implementation details are provided in Section 5.3.3.

AUC and accuracy results of MIL classifiers Table 5.4 are the averages of a ten-fold cross validation repeated for five times. The best result for each dataset is in boldface. In molecular activity prediction, the highest AUC results are obtained by QP-MIL in Musk 1, and by $D_{meanmin}$ in Musk 2. Fisher vector based bag representation suits on Mutagenesis 1 dataset, where second best AUC and accuracy results are obtained by QP-MIL and miFV, respectively. In Protein, the leading method is MILES, which is followed by QP-MIL.

QP-MIL has the best image classification success in Elephant, Tiger and USCB Breast cancer datasets. The implicit instance selection mechanism of MILES is effective on Fox dataset and QP-MIL follows MILES on this dataset. In Corel image datasets, $D_{meanmin}$ has the highest average performance, and QP-MIL performs very close to $D_{meanmin}$. Results of QP-MIL and $D_{meanmin}$ are very close to each other on the average on 20 Newsgroups datasets. In Web recommendation, performance of QP-MIL falls

behind miFV and $D_{meanmin}$. QP-MIL has the highest AUC and accuracy results in almost all Birds datasets.

The average testing results based on problem categories are reported in Table 5.5. For each problem category, results of the best method are in boldface, whereas the second best results are shown in italic. Average AUC and accuracy results in Table 5.5 demonstrate that QP-MIL is competitive with other algorithms across all application categories and provides the best classification results on some datasets. QP-MIL achieves the best or the second best average AUC and accuracy performance on molecular activity prediction datasets.

Image classification results in Table 5.5 reveal that QP-MIL is broadly comparable with the competitors in all benchmarks. In text categorization, performance of QP-MIL is competitive in Newsgroups datasets and miFV is the leading method in Web recommendation datasets. QP-MIL yields the best average AUC and accuracy results in audio recording classification as verified by the reported results on Birds competition. Finally, QP-MIL has the best overall average AUC and accuracy results.

Both miFV and $D_{meanmin}$ are bag-level methods and they are mostly tuned for computer vision and bioinformatics applications of MIL. However, QP-MIL is not tailored for a certain MIL application and overall results of this section confirm generalizability of our approach to various application domains. Without forcing the standard MIL assumption, QP-MIL matches or outperforms the state-of-the-art algorithms on a broad range of applications.

Table 5.4. AUC and accuracy results of four MIL methods with standard errors (× 100).

Dataset	AUC				Accuracy			
	MILES	$D_{meanmin}$	miFV	QP-MIL	MILES	$D_{meanmin}$	miFV	QP-MIL
Musk 1 ♣	93.5 (1.0)	94.5 (1.2)	94.1 (1.2)	**96.8 (0.8)**	79.6 (2.0)	84.1 (1.8)	85.2 (1.6)	**88.4 (1.4)**
Musk 2 ♣	96.7 (0.7)	**97.6 (0.8)**	94.7 (1.2)	94.5 (1.0)	86.3 (1.1)	**92.3 (1.1)**	87.7 (1.6)	87.8 (1.5)
Mutagenesis 1 ♣	78.0 (1.5)	85.1 (1.2)	**88.7 (1.2)**	85.5 (1.5)	79.8 (1.2)	77.4 (1.3)	**82.6 (1.2)**	77.6 (1.5)
Mutagenesis 2 ♣	62.7 (5.0)	64.7 (5.3)	68.3 (5.0)	**78.5 (3.8)**	70.9 (2.7)	70.4 (2.0)	**76.6 (2.2)**	71.2 (3.0)
Protein ♣	**95.3 (1.1)**	52.3 (3.7)	80.0 (1.9)	85.1 (1.6)	**94.9 (0.6)**	87.1 (0.3)	85.4 (0.8)	88.3 (0.9)
Elephant ♥	92.7 (0.7)	93.6 (0.9)	91.4 (0.9)	**94.1 (0.7)**	82.7 (1.0)	**86.2 (1.0)**	82.9 (1.1)	85.0 (0.9)
Fox ♥	**73.8 (1.6)**	61.2 (1.7)	67.5 (1.5)	67.7 (1.4)	**64.9 (1.4)**	57.8 (1.2)	62.3 (1.3)	63.4 (1.2)
Tiger ♥	86.8 (1.0)	85.3 (1.1)	87.5 (1.1)	**90.1 (1.0)**	79.2 (1.2)	77.7 (1.2)	80.4 (1.3)	**81.8 (1.2)**
Corel, African ♥	95.7 (0.5)	**96.7 (0.4)**	94.4 (0.6)	95.3 (0.5)	96.6 (0.2)	**97.3 (0.1)**	96.4 (0.1)	95.8 (0.2)
Corel, Antique ♥	87.3 (0.7)	**92.2 (0.6)**	90.8 (0.6)	87.1 (0.7)	93.8 (0.2)	**95.4 (0.1)**	95.0 (0.1)	92.7 (0.3)
Corel, Battleships ♥	94.9 (0.5)	**98.1 (0.2)**	92.9 (0.6)	95.8 (0.3)	96.1 (0.2)	**97.1 (0.1)**	96.2 (0.1)	96.0 (0.2)
Corel, Beach ♥	**99.3 (0.1)**	98.3 (0.4)	97.4 (0.4)	**99.3 (0.1)**	98.1 (0.1)	97.8 (0.1)	97.7 (0.1)	**98.4 (0.1)**
Corel, Buses ♥	**97.5 (0.4)**	97.3 (0.4)	94.0 (0.7)	96.0 (0.4)	**98.1 (0.1)**	97.7 (0.1)	97.2 (0.2)	96.8 (0.2)
Corel, Cars ♥	91.9 (0.7)	**94.8 (0.5)**	91.7 (0.7)	91.2 (0.7)	95.4 (0.2)	**97.3 (0.1)**	96.5 (0.1)	95.2 (0.2)
Corel, Desserts ♥	93.9 (0.7)	**97.4 (0.3)**	97.3 (0.4)	96.9 (0.3)	96.5 (0.1)	**97.7 (0.1)**	97.4 (0.1)	96.9 (0.2)
Corel, Dinosaurs ♥	97.5 (0.3)	**98.3 (0.2)**	94.4 (0.5)	96.4 (0.4)	97.5 (0.1)	**97.9 (0.1)**	96.6 (0.1)	96.3 (0.2)
Corel, Dogs ♥	87.4 (1.1)	**91.9 (0.7)**	86.4 (1.2)	86.3 (0.9)	94.8 (0.2)	**96.3 (0.1)**	95.6 (0.1)	93.4 (0.3)
Corel, Elephants ♥	95.7 (0.4)	**98.2 (0.2)**	95.7 (0.4)	96.8 (0.3)	96.0 (0.1)	**96.9 (0.1)**	96.5 (0.1)	96.2 (0.1)
Corel, Fashion ♥	99.0 (0.1)	**99.0 (0.1)**	98.9 (0.2)	98.6 (0.2)	98.3 (0.1)	**98.6 (0.1)**	98.1 (0.1)	98.1 (0.1)
Corel, Flowers ♥	94.3 (0.6)	**94.7 (0.6)**	93.8 (0.6)	93.8 (0.5)	96.1 (0.2)	**97.0 (0.1)**	96.4 (0.2)	95.5 (0.2)
Corel, Food ♥	99.4 (0.1)	**99.8 (0.1)**	98.7 (0.1)	99.2 (0.1)	98.6 (0.1)	**98.9 (0.1)**	97.9 (0.2)	97.9 (0.1)
Corel, Historical ♥	99.3 (0.1)	**99.8 (0.0)**	98.5 (0.3)	99.4 (0.1)	98.7 (0.1)	**98.9 (0.1)**	97.8 (0.1)	98.6 (0.1)
Corel, Horses ♥	89.6 (0.7)	**92.0 (0.6)**	88.9 (0.8)	86.0 (0.8)	94.6 (0.2)	**96.2 (0.1)**	95.9 (0.1)	93.1 (0.2)

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

Table 5.4. – AUC and accuracy results of four MIL methods with standard errors (× 100) (cont.).

Dataset	AUC				Accuracy			
	MILES	$D_{meanmin}$	miFV	QP-MIL	MILES	$D_{meanmin}$	miFV	QP-MIL
Corel, Lizards ♥	97.0 (0.4)	**98.0 (0.3)**	95.8 (0.5)	97.3 (0.3)	**97.8 (0.1)**	**97.8 (0.1)**	97.0 (0.1)	97.2 (0.2)
Corel, Mountains ♥	99.9 (0.0)	**100 (0.0)**	99.9 (0.0)	99.8 (0.0)	99.3 (0.1)	**99.5 (0.1)**	**99.5 (0.1)**	99.1 (0.1)
Corel, Skiing ♥	94.7 (0.4)	**96.0 (0.3)**	95.9 (0.4)	**96.0 (0.3)**	**96.4 (0.1)**	**96.4 (0.1)**	96.3 (0.1)	96.0 (0.1)
Corel, Sunset ♥	76.3 (1.2)	**83.7 (1.0)**	77.1 (1.3)	73.9 (1.2)	92.4 (0.2)	**95.2 (0.1)**	95.1 (0.1)	90.5 (0.4)
Corel, Waterfalls ♥	94.5 (0.5)	**97.5 (0.2)**	93.4 (0.5)	95.5 (0.3)	96.0 (0.2)	**97.0 (0.2)**	95.8 (0.1)	95.5 (0.2)
UCSB Breast Cancer ♥	83.3 (2.6)	83.1 (2.7)	86.8 (2.5)	**88.8 (2.2)**	75.8 (2.2)	72.2 (2.3)	79.6 (2.4)	**81.7 (2.0)**
Newsgroups 1, alt.atheism ♠	41.6 (2.3)	**94.1 (1.0)**	91.1 (1.2)	93.3 (1.2)	41.8 (1.8)	**85.6 (1.5)**	81.2 (1.4)	82.0 (1.6)
N.g. 2, comp.graphics ♠	52.8 (2.2)	**89.8 (1.6)**	57.2 (3.2)	83.1 (1.8)	52.2 (2.0)	**79.0 (1.4)**	53.4 (1.2)	75.0 (1.8)
N.g. 3, comp.os.ms-windows.misc ♠	47.9 (2.7)	**81.0 (2.1)**	66.8 (2.2)	77.8 (2.1)	46.6 (2.2)	54.0 (0.9)	55.2 (1.7)	**71.2 (1.7)**
N.g. 4, comp.sys.ibm.pc.hardware ♠	68.2 (2.4)	**85.7 (2.2)**	69.5 (2.4)	82.0 (1.9)	62.4 (2.5)	**75.4 (1.6)**	65.0 (2.1)	74.4 (1.9)
N.g. 5, comp.sys.mac.hardware ♠	58.9 (2.8)	**85.2 (1.6)**	65.0 (2.6)	81.2 (1.6)	56.4 (2.4)	**79.6 (1.2)**	59.6 (1.6)	73.4 (1.6)
N.g. 6, comp.windows.x ♠	61.0 (2.4)	**89.0 (1.7)**	82.2 (2.0)	84.9 (2.2)	56.8 (2.0)	66.8 (1.5)	75.0 (1.9)	**76.4 (2.0)**
N.g. 7, misc.forsale ♠	51.1 (2.4)	79.0 (2.0)	72.6 (2.5)	**81.6 (1.8)**	53.0 (2.3)	53.2 (1.5)	60.0 (1.8)	**70.0 (2.1)**
N.g. 8, rec.autos ♠	49.8 (2.5)	**87.0 (1.7)**	72.7 (2.5)	81.8 (1.9)	50.2 (1.7)	**77.0 (1.6)**	63.0 (1.9)	69.0 (2.0)
N.g. 9, rec.motorcycles ♠	52.2 (2.6)	32.6 (3.2)	81.2 (2.4)	**85.0 (1.9)**	51.8 (1.9)	51.0 (0.5)	74.0 (2.1)	**74.8 (1.9)**
N.g. 10, rec.sport.baseball ♠	51.0 (2.7)	**91.4 (1.4)**	86.4 (1.8)	88.0 (1.8)	50.6 (1.9)	**80.0 (1.6)**	77.2 (1.5)	75.8 (1.7)
N.g. 11, rec.sport.hockey ♠	37.4 (2.2)	**95.8 (0.8)**	87.9 (1.5)	92.3 (1.4)	42.0 (2.1)	**85.8 (1.5)**	77.0 (1.5)	82.6 (1.8)
N.g. 12, sci.crypt ♠	46.2 (2.7)	84.0 (1.9)	85.1 (1.8)	**86.5 (2.0)**	47.2 (2.0)	62.4 (1.6)	**77.6 (2.1)**	75.6 (2.1)
N.g. 13, sci.electronics ♠	48.7 (2.2)	94.6 (1.0)	61.6 (2.6)	**94.9 (0.8)**	46.6 (1.7)	**89.0 (1.3)**	57.4 (1.6)	87.4 (1.2)
N.g. 14, sci.med ♠	49.8 (2.4)	**94.2 (0.8)**	84.3 (1.7)	88.8 (1.5)	52.0 (1.9)	**80.0 (1.4)**	73.2 (1.5)	78.6 (1.4)
N.g. 15, sci.space ♠	43.6 (2.5)	90.5 (1.4)	82.9 (1.9)	**92.1 (1.1)**	45.2 (2.1)	78.4 (1.5)	77.4 (1.9)	**83.6 (1.4)**
N.g. 16, soc.religion.christian ♠	46.1 (2.4)	**89.8 (1.4)**	84.9 (1.5)	82.2 (2.0)	45.0 (1.9)	**83.8 (1.4)**	76.0 (1.4)	70.2 (1.7)
N.g. 17, talk.politics.guns ♠	49.8 (2.5)	**87.4 (1.5)**	82.7 (2.0)	79.6 (1.9)	48.4 (2.1)	**77.8 (1.6)**	73.4 (1.7)	66.0 (1.9)
N.g. 18, talk.politics.mideast ♠	54.6 (3.0)	**87.4 (1.7)**	85.8 (1.9)	85.2 (1.5)	53.8 (2.1)	78.2 (1.3)	**79.0 (1.5)**	76.6 (1.7)

Table 5.4. – AUC and accuracy results of four MIL methods with standard errors (× 100) (cont.).

Dataset	AUC				Accuracy			
	MILES	$D_{meanmin}$	miFV	QP-MIL	MILES	$D_{meanmin}$	miFV	QP-MIL
N.g. 19, talk.politics.misc ♠	55.0 (2.3)	80.2 (1.9)	67.2 (2.9)	**81.4 (2.0)**	51.8 (2.1)	68.6 (1.7)	60.0 (2.2)	**72.2 (1.8)**
N.g. 20, talk.religion.misc ♠	56.0 (2.7)	**83.4 (2.2)**	80.9 (2.3)	81.3 (2.3)	52.8 (2.3)	62.4 (1.3)	69.2 (2.2)	**70.8 (1.9)**
Web 1 ♠	73.2 (3.0)	63.4 (4.2)	**83.2 (2.3)**	65.5 (3.4)	**75.5 (1.5)**	69.9 (0.9)	74.9 (1.3)	68.1 (2.0)
Web 2 ♠	**54.4 (3.9)**	47.4 (4.2)	37.1 (2.5)	53.7 (4.4)	75.0 (1.1)	**75.3 (0.9)**	73.4 (1.2)	68.1 (2.3)
Web 3 ♠	67.1 (4.4)	70.8 (4.6)	**73.3 (3.6)**	66.1 (4.1)	**85.1 (1.4)**	81.6 (1.0)	82.1 (1.1)	74.9 (2.1)
Web 4 ♠	74.3 (3.5)	79.9 (3.6)	**81.2 (3.4)**	62.7 (3.3)	76.5 (1.8)	75.8 (0.9)	**82.5 (1.7)**	69.6 (1.6)
Web 5 ♠	**74.3 (3.4)**	71.1 (3.7)	68.7 (3.4)	55.2 (3.8)	**82.1 (1.3)**	79.7 (1.1)	79.8 (1.1)	78.0 (1.8)
Web 6 ♠	55.0 (3.4)	52.5 (4.2)	64.6 (3.6)	**65.0 (3.6)**	**81.5 (1.0)**	78.9 (0.8)	74.7 (1.3)	75.1 (1.7)
Web 7 ♠	62.5 (2.6)	69.0 (2.8)	**69.7 (3.4)**	54.5 (3.2)	54.3 (1.9)	63.1 (2.3)	**65.1 (2.8)**	51.3 (2.7)
Web 8 ♠	51.1 (3.1)	40.9 (2.6)	**53.7 (2.4)**	53.0 (3.0)	50.1 (2.3)	50.3 (1.4)	**53.3 (2.1)**	51.7 (2.7)
Web 9 ♠	68.7 (2.3)	**73.5 (2.7)**	68.5 (3.1)	50.3 (3.2)	61.8 (2.1)	**69.3 (2.2)**	65.9 (2.0)	49.8 (2.5)
Birds, Brown creeper ♦	97.4 (0.3)	89.9 (0.5)	98.8 (0.2)	**99.0 (0.1)**	92.4 (0.5)	81.8 (0.7)	95.1 (0.5)	**95.8 (0.4)**
Birds, Chestnut-backed chickadee ♦	80.1 (1.3)	85.3 (0.8)	**92.3 (0.8)**	91.7 (0.5)	80.6 (0.8)	88.4 (0.5)	**91.1 (0.4)**	86.9 (0.5)
Birds, Dark-eyed junco ♦	89.1 (1.2)	85.6 (1.3)	88.1 (1.2)	**93.2 (0.6)**	**95.8 (0.3)**	**95.8 (0.2)**	95.2 (0.3)	94.6 (0.3)
Birds, Hammonds flycatcher ♦	93.9 (0.8)	94.4 (0.7)	94.0 (0.7)	**100.0 (0.0)**	91.1 (0.8)	90.8 (0.4)	92.6 (0.4)	**99.6 (0.1)**
Birds, Hermit thrush ♦	68.2 (3.0)	57.8 (4.4)	66.2 (3.1)	**90.3 (1.1)**	96.7 (0.2)	**97.2 (0.1)**	97.0 (0.1)	95.6 (0.3)
Birds, Hermit warbler ♦	90.4 (1.3)	78.1 (1.5)	94.0 (0.6)	**98.4 (0.3)**	90.4 (0.5)	91.6 (0.3)	93.8 (0.4)	**95.2 (0.5)**
Birds, Olive-sided flycatcher ♦	92.0 (0.5)	89.6 (0.6)	95.9 (0.4)	**96.7 (0.3)**	88.7 (0.5)	84.3 (0.2)	91.3 (0.5)	**91.4 (0.5)**
Birds, Pacificslope flycatcher ♦	84.8 (0.8)	75.4 (1.0)	**98.6 (0.2)**	94.3 (0.4)	79.6 (0.8)	77.6 (0.5)	**95.4 (0.4)**	86.4 (0.6)
Birds, Red-breasted nuthatch ♦	90.7 (0.7)	87.6 (0.7)	94.6 (0.5)	**97.1 (0.3)**	88.4 (0.6)	85.0 (0.2)	90.3 (0.5)	**92.7 (0.5)**
Birds, Swainsons thrush ♦	80.4 (1.7)	76.7 (1.7)	91.4 (1.0)	**97.6 (0.3)**	86.7 (0.7)	91.4 (0.3)	93.4 (0.4)	**94.2 (0.5)**
Birds, Varied thrush ♦	95.1 (0.6)	84.0 (1.2)	93.0 (0.7)	**99.7 (0.2)**	92.7 (0.6)	88.1 (0.3)	91.4 (0.4)	**98.5 (0.2)**
Birds, Western tanager ♦	89.4 (1.6)	84.9 (1.8)	**98.9 (0.2)**	97.3 (0.3)	93.5 (0.5)	94.7 (0.3)	**97.9 (0.2)**	94.6 (0.4)
Birds, Winter wren ♦	94.6 (0.4)	93.1 (0.7)	**99.7 (0.1)**	98.8 (0.1)	91.2 (0.4)	93.4 (0.4)	**97.6 (0.3)**	94.7 (0.4)

Table 5.5. Average AUC and accuracy results of four MIL methods based on problem categories.

Dataset	**AUC**				**Accuracy**			
	MILES	$D_{meanmin}$	miFV	QP-MIL	MILES	$D_{meanmin}$	miFV	QP-MIL
Musk ♣	95.1	**96.1**	94.4	*95.7*	83.0	**88.2**	*86.5*	**88.1**
Mutagenesis ♣	70.4	74.9	*78.5*	**82.0**	*75.4*	73.9	**79.6**	74.4
Protein ♣	**95.3**	52.3	80.0	*85.1*	**94.9**	87.1	85.4	*88.3*
Elephant, Fox, Tiger ♥	**84.4**	80.0	82.1	*84.0*	*75.6*	73.9	75.2	**76.7**
Corel ♥	*94.3*	**96.2**	93.8	94.0	96.6	**97.3**	*96.7*	96.0
UCSB Breast Cancer ♥	83.3	83.1	*86.8*	**88.8**	75.8	72.2	*79.6*	**81.7**
Newsgroups ♠	51.1	**85.1**	*77.4*	**85.2**	50.3	*73.4*	69.2	**75.3**
Web recommendation ♠	*64.5*	63.2	**66.7**	58.4	71.3	*71.5*	**72.4**	65.2
Birds ♦	88.2	83.3	*92.7*	**96.5**	*89.8*	89.2	**94.0**	**93.9**
Avg.	80.7	*84.4*	81.1	**86.5**	76.9	*83.4*	83.3	**83.9**

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

Table 5.6 shows the time taken up by experiments of QP-MIL on 71 datasets. Again, reported results are the averages after 5 repeats of a ten-fold cross validation. We divide the total time spent by QP-MIL into three main parts: representation learning (RL) time, inner cross-validation (CV) time and model solution time. At first, we obtain clustering-based data representation. We determine the required number of clusters on the training instances and use the resulting cluster centers to represent the training bags. Compared to the computational time on the training instances, RL time for the test bags is negligible. Therefore, we only report the RL time consumed on the training set. As described in Section 5.3.3, we report classification results after a nested cross-validation procedure. The time spent for inner cross-validation loop is the CV time. After parameter selection, we solve QP model and record execution time of barrier algorithm as the model solution time.

Table 5.6 reveals that QP models are solved efficiently regardless of the dataset dimensionality. Due to the repeated solution of the QP model within each inner fold, significant amount of time is spent on parameter selection. However, RL times are considerably longer compared to CV times in Web datasets since large number of features complicates the dissimilarity calculations in data representation phase. In Mutagenesis datasets, predetermined value of the threshold controlling parameter ε may cause infeasibility in QP models. If infeasibility is detected, we solve an auxiliary optimization problem to deal with this situation. Specifically, by keeping the original constraints of (5.1), we convert ε into a decision variable and maximize its value. This way, a suitable value of ε is derived. Then, QP model (5.1) is solved after stating the selected ε value. This process increases both the CV time and model solution time on these datasets as seen in Table 5.6. QP-MIL provides an efficient learning approach concerning different MIL application categories. In the light of parameter sensitivity discussions in Section 5.3.4.3, QP-MIL can be implemented without parameter selection to gain from the execution time.

Table 5.6. Average time results of QP-MIL.

Dataset	Instances	Features	Bags	RL time	CV time	Solution time
Musk 1 ♣	476	166	92	5.3	15.4	0.3
Musk 2 ♣	6598	166	102	59.4	187.0	2.7
Mutagenesis 1 ♣	10486	7	188	47.8	844.1	27.4
Mutagenesis 2 ♣	2132	7	42	25.7	570.8	20.0
Protein ♣	26611	8	193	125.2	782.7	13.2
Elephant ♥	1391	230	200	17.8	77.5	1.5
Fox ♥	1302	230	200	19.5	77.1	1.6
Tiger ♥	1220	230	200	16.1	69.9	1.6
Corel, African ♥	7947	9	2000	34.2	294.6	5.0
Corel, Antique ♥	7947	9	2000	36.6	346.2	7.4
Corel, Battleships ♥	7947	9	2000	34.7	339.3	6.1
Corel, Beach ♥	7947	9	2000	30.8	321.0	5.8
Corel, Buses ♥	7947	9	2000	31.7	325.8	5.8
Corel, Cars ♥	7947	9	2000	34.1	350.1	6.1
Corel, Desserts ♥	7947	9	2000	37.0	345.9	5.7
Corel, Dinosaurs ♥	7947	9	2000	35.5	329.8	6.0
Corel, Dogs ♥	7947	9	2000	35.7	338.2	6.8
Corel, Elephants ♥	7947	9	2000	32.6	341.0	6.0
Corel, Fashion ♥	7947	9	2000	37.1	350.3	6.3
Corel, Flowers ♥	7947	9	2000	41.7	336.2	6.2
Corel, Food ♥	7947	9	2000	39.8	333.7	5.8
Corel, Historical ♥	7947	9	2000	42.5	330.8	6.1
Corel, Horses ♥	7947	9	2000	37.9	330.2	6.8
Corel, Lizards ♥	7947	9	2000	32.4	321.0	5.7
Corel, Mountains ♥	7947	9	2000	34.8	370.1	6.0
Corel, Skiing ♥	7947	9	2000	40.0	316.1	5.6
Corel, Sunset ♥	7947	9	2000	41.9	362.0	12.0
Corel, Waterfalls ♥	7947	9	2000	28.4	348.6	6.6
UCSB Breast Cancer ♥	2002	708	58	33.6	31.0	0.5
Newsgroups 1, alt.atheism ♠	5443	200	100	41.2	129.6	1.8
N.g. 2, comp.graphics ♠	3094	200	100	57.1	303.1	4.5
N.g. 3, comp.os.ms-windows.misc ♠	5175	200	100	79.4	172.0	2.7
N.g. 4, comp.sys.ibm.pc.hardware ♠	4827	200	100	85.3	179.8	2.7
N.g. 5, comp.sys.mac.hardware ♠	4473	200	100	83.5	239.1	2.9
N.g. 6, comp.windows.x ♠	3110	200	100	45.2	217.6	2.8
N.g. 7, misc.forsale ♠	5306	200	100	75.7	162.9	2.4
N.g. 8, rec.autos ♠	3458	200	100	53.7	282.2	3.0
N.g. 9, rec.motorcycles ♠	4730	200	100	47.3	128.1	1.9
N.g. 10, rec.sport.baseball ♠	3358	200	100	49.4	265.0	4.0

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

Table 5.6. – Average time results of QP-MIL (cont.).

Dataset	Instances	Features	Bags	RL time	CV time	Solution time
N.g. 11, rec.sport.hockey ♠	1982	200	100	32.8	176.0	3.8
N.g. 12, sci.crypt ♠	4284	200	100	30.4	97.5	1.3
N.g. 13, sci.electronics ♠	3192	200	100	65.0	380.6	6.4
N.g. 14, sci.med ♠	3045	200	100	26.1	143.9	2.0
N.g. 15, sci.space ♠	3655	200	100	32.1	146.6	2.3
N.g. 16, soc.religion.christian ♠	4677	200	100	35.9	112.1	1.6
N.g. 17, talk.politics.guns ♠	3558	200	100	27.0	107.6	1.6
N.g. 18, talk.politics.mideast ♠	3376	200	100	40.9	181.3	2.3
N.g. 19, talk.politics.misc ♠	4788	200	100	39.3	108.7	1.5
N.g. 20, talk.religion.misc ♠	4606	200	100	36.6	113.9	1.6
Web 1 ♠	2212	5863	75	143.3	29.1	0.4
Web 2 ♠	2212	6519	75	144.8	26.1	0.4
Web 3 ♠	2212	6306	75	154.7	31.5	0.4
Web 4 ♠	2291	6059	75	142.4	26.9	0.4
Web 5 ♠	2546	6407	75	158.7	33.4	0.5
Web 6 ♠	2462	6417	75	156.7	26.2	0.4
Web 7 ♠	2400	6450	75	151.6	27.1	0.4
Web 8 ♠	2183	5999	75	137.2	23.1	0.4
Web 9 ♠	2321	6279	75	149.6	28.2	0.4
Birds, Brown creeper ♦	10232	38	548	43.3	211.8	3.6
Birds, Chestnut-backed chickadee ♦	10232	38	548	43.8	212.7	3.6
Birds, Dark-eyed junco ♦	10232	38	548	39.3	241.7	4.3
Birds, Hammonds flycatcher ♦	10232	38	548	40.9	220.2	4.3
Birds, Hermit thrush ♦	10232	38	548	48.9	228.9	3.9
Birds, Hermit warbler ♦	10232	38	548	47.6	229.7	4.0
Birds, Olive-sided flycatcher ♦	10232	38	548	46.4	232.4	4.1
Birds, Pacificslope flycatcher ♦	10232	38	548	47.4	232.6	3.9
Birds, Red-breasted nuthatch ♦	10232	38	548	46.9	225.1	3.9
Birds, Swainsons thrush ♦	10232	38	548	43.5	234.9	3.9
Birds, Varied thrush ♦	10232	38	548	49.5	237.2	4.0
Birds, Western tanager ♦	10232	38	548	48.5	241.1	4.3
Birds, Winter wren ♦	10232	38	548	44.7	239.6	4.1

5.3.4.3. Parameter Sensitivity. In this section, we conduct experiments on four real-world datasets to examine the sensitivity of QP-MIL to C setting. Six different values of C are tested with 50 replicates of the experiments. We select the tuning set of C as $\{0.01, 0.1, 1, 10, 100, 1000\}$. We execute data representation and model solving as described in Section 5.3.3 except for the inner cross validation. For each level of C, we solve QP model (5.1) and record the classification results for the test bags. Figure 5.5 presents the behavior of the QP-MIL classifier on four datasets. For each dataset, boxplots show the AUC values for different levels of C. For Musk 2, value of C does not have a significant effect on the AUC performance. Corresponding boxplots in Figure 5.5 show that smaller C values yield slightly better AUC results in Elephant dataset. Finally, analysis with the boxplots in Figure 5.5 demonstrates that changing value of C does not significantly affect the AUC performance for other datasets.

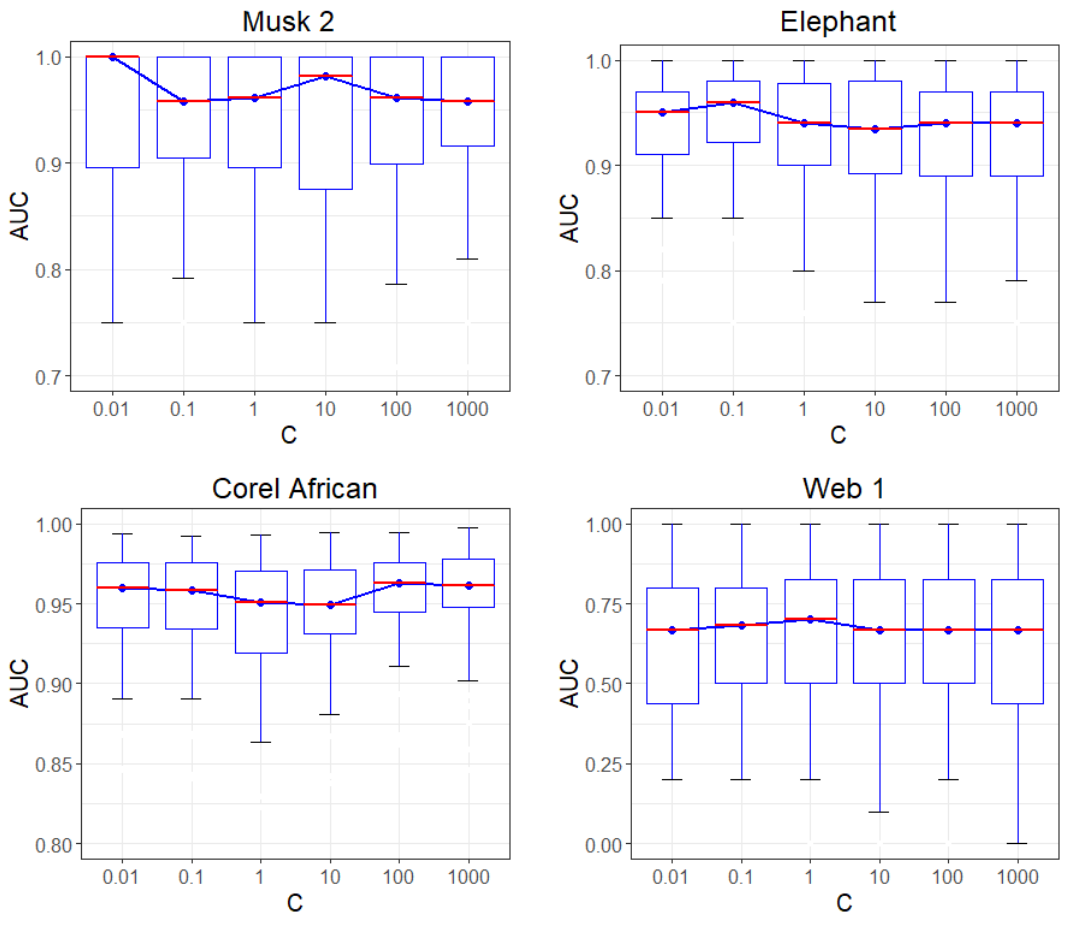


Figure 5.5. Sensitivity of the QP-MIL to different values for C on 4 real-world datasets.

The reported results of the comparisons with baseline approaches are provided after a cross-validation procedure in Section 5.3.4.2. The trade-off between maximization of bag class membership margin and sparsity of the weighting vector can be considered as a practically dispensable criterion for learning. Since most of the computation time is consumed by parameter selection as reported in Table 5.3, value of C can be fixed initially for run-time considerations. Setting a higher value of C introduces potential risk of overfitting, and therefore may reduce generalization to unknown objects. As shown in the boxplots of Figure 5.5, small C values yield higher AUC values in both Musk 2 and Elephant. Therefore, if the parameter selection phase is skipped, we suggest to use small values of C to obtain satisfactory results.

5.4. Conclusions

In this chapter, we propose an optimization-based method, QP-MIL, to solve multiple instance classification problem, where a bag of instances are classified instead of single instances. Our algorithm is based on an quadratic programming (QP) formulation, which performs classification without imposing additional constraints on relating instance labels to the bag labels. Solving QP problem produces a decision function, which computes a bag class membership score by aggregating instance-level scores. Instance-level scores are obtained by a linear function of feature values. This way, all instances contribute to the bag label and their contributions are modeled by specifying the feature weights. The optimization process outputs a bag-level decision threshold to classify new bags together with the decision function. Distances of bag class memberships to the threshold value are maximized and the sparseness of feature weight vector is controlled by a cost parameter.

We have tested our approach on a wide range of datasets from various categories such as drug activity prediction, image categorization, text mining and audio recording classification. In order to support further research on this area, we serve the used datasets, codes and configurations on our supporting page [63]. We compared the performance of our approach to state-of-the-art machine learning based approaches. To model instance relationships, cluster centers are selected as prototypes and input

features are the instance-to-prototype distances. For each dataset, generated problem instances can be easily solved to optimality in seconds. Our experiments on 71 datasets indicate that QP-MIL is competitive with the recent successful heuristic algorithms, and provides the best classification results on a variety of datasets.

Since this study focuses on optimization-based MIL, we also performed comparisons with a recent method MIHLSVM in terms of problem size and computation time. MIHLSVM solves mixed integer quadratic programs to learn a bag classifier. Our comparisons between QP-MIL and MIHLSVM indicate that MIHLSVM problem instances have difficulties to scale to large datasets. We applied generalized Benders decomposition (GBD) to MIHLSVM to inspect the possible reductions in solution time. Our computational results show that neither direct solution of MIHLSVM nor GBD approach is able to retrieve satisfactory solutions to MIL problem within a reasonable amount of time. Finally, we examined the effect of the cost parameter and illustrated that the classification performance does not excessively depend on adjustment of the cost parameter. Our MIL approach offers an efficient solution to MIL problem in terms of classification accuracy and model solution time, and can be extended to large real-world challenges as a future work.

6. CONCLUSION AND FUTURE RESEARCH

In this thesis, we propose learning frameworks specialized to solve real world learning problems that can be generalized to classify bags of multiple instances. Unavailability of instance label information in MIL setting prevents the application of regular supervised learning. To resolve this problem, researchers devise methods focusing on certain assumptions regarding the instance labels. However, it is not a trivial task to determine which assumption holds for a new type of MIL problem. A bag-level representation based on instance characteristics does not require assumptions about the instance labels and is shown to be successful in MIL tasks. These approaches mainly encode bag vectors using bag-of-features type of representations.

In Chapter 3, we propose bag encoding strategies that partition the instance feature space and represent the bags using the frequency of instances residing at each partition. Proposed tree-based encoding algorithm implicitly learns a generalized Gaussian Mixture Model (GMM) on the instance feature space, and transforms this information into a bag-level summary. We show that bag representation using tree ensembles provides fast, accurate and robust representations. Our experiments on a large database of MIL problems show that tree-based encoding is highly scalable and its performance is competitive with recent successful MIL approaches.

Previous optimization based MIL approaches encode the standard MIL assumption and form models with computational difficulties. To handle the potential problems with the standard assumption, Chapter 4 proposes a linear programming (LP) framework to learn instance level contributions to the bag labels. Each instance of a bag is mapped to a pseudo-class membership estimate and these estimates are aggregated to obtain the bag-level class membership in an optimization framework. A simple linear mapping enables handling various MIL assumptions with adjusting instance contributions. Our experiments with instance-dissimilarity based data representations verify effectiveness of the proposed MIL framework. Proposed LP formulation requires no parameters to tune and can be solved efficiently in polynomial time.

Existing MIL models in the literature make use of certain assumptions regarding the instance labels and provide mixed integer quadratic programs, which introduce computational difficulties. In Chapter 5, we present a novel quadratic programming (QP)-based approach to classify bags. Solution of our QP formulation links the instance-level contributions to the bag label estimates, and outputs a linear bag classifier along with a decision threshold. Our approach imposes no additional constraints on relating instance labels to bag labels and can be adapted to learning applications with different MIL assumptions. Unlike existing specialized heuristic approaches to solve previous MIL formulations, our QP models can be directly solved to optimality using any commercial QP solver. Our computational experiments show that proposed QP formulation is efficient in terms of solution time, overcoming a main drawback of previous optimization algorithms for MIL.

In our study, a wide range of real world datasets from different application domains form the largest experimented repository to compare MIL algorithms. For each experiment, the results presented are averages computed after a ten-fold cross-validation repeated five times. The same cross validation folds are used in all experiments to allow fair comparability. Our experiments demonstrate that resulting performance of the proposed methods are competitive with the state-of-the-art algorithms. Table 6.1 shows the average classification results of the proposed methods and six competitors tested on various problem categories. The results on all individual datasets are available on the website [116]. Highest result in each problem category is in boldface and the second best results are in italic. Classification of bags after random tree ensemble-based encoding formed by decision paths turned out to be the best overall approach for MIL. The second best method is QP-MIL, which points out a good candidate benchmark to the studies employing mathematical programming approaches to solve MIL problems. LP-MIL and BoW representation with k-means clustering provide competitive results with the state-of-the-art MIL methods miFV [50], MILES [8] and MInD [26] with D_{meanmin} dissimilarity measure.

Table 6.1. AUC performance comparison of various MIL methods.

Dataset	**Algorithm AUC (%)**										
							Bag encoding		LP-MIL		
	APR	Citation-kNN	CCE	MILES	$D_{meanmin}$	miFV	k-means two class	Path	$R^{instance}$	$R^{cluster}$	QP-MIL
Musk ♣	78.9	86.1	83.1	95.1	**96.1**	94.4	83.0	94.4	94.4	94.7	*95.7*
Mutagenesis ♣	47.9	75.7	78.0	70.3	74.9	78.5	**88.8**	*88.5*	82.0	82.6	82.0
Protein ♣	50.9	55.2	64.3	**95.3**	52.3	80.0	87.3	*94.6*	-	83.9	85.1
Elephant, Fox, Tiger ♥	63.2	73.4	76.9	84.4	80.0	82.1	79.9	**85.5**	*84.7*	81.3	84.0
Corel ♥	63.8	84.8	88.7	94.3	*96.2*	93.8	89.9	**96.5**	95.6	94.4	94.0
UCSB Breast Cancer ♥	56.9	70.6	64.4	83.3	83.1	86.8	84.8	88.0	**93.0**	*90.3*	88.8
Newsgroups ♠	50.0	71.1	73.1	51.1	*85.1*	77.4	64.0	80.8	51.3	52.9	**85.2**
Web recommendation ♠	56.6	57.9	63.7	64.5	63.2	*66.7*	**74.7**	64.5	61.0	53.9	58.4
Birds ♦	59.6	75.3	83.5	88.2	83.3	92.7	96.3	**98.1**	*97.8*	95.8	96.5
Avg.	57.9	74.5	78.5	80.8	84.4	81.1	84.3	**87.4**	78.0	76.7	*86.5*

MIL application categories: ♣ molecular activity prediction, ♥ image annotation, ♠ text classification, ♦ audio recording classification.

A large number of existing approaches to MIL comprise of machine learning algorithms, which lead to heuristic solution methods. Most of them are commonly used to solve the standard MIL problems. A portion of the heuristic methods attempt to solve MIL problem with lower restrictions and benefit from bag-level class information. Motivated by the success of previous bag representation algorithms, we represent each bag with constructed simple and sparse vectors. Subsequently, we train a bag-level classier on the new feature space. Success of bag classification highly depends on the way of bag representation because of encoding the required information for class separation. In bag-encoding using random tree ensembles, bag representation vectors contain rich information on instances and preserve the bag structure. Random tree ensembles can be efficiently trained on large scaled data and as an extension, ensemble learning phase can be easily parallelized for further acceleration. Besides, dimensionality reduction by parallelization to improve learning performance on big data is tackled recently for random forest classifiers [117], but not for random tree ensembles.

A higher level of stability on the solutions to MIL problem can be obtained by solving optimization models. Namely, every time we repeat the experiment for a MIL dataset, we will obtain the same results by solving the corresponding problem instance to optimality. SVM-based MIL methods solve margin maximization formulations to minimize the generalization error bound [118]. However, direct solutions to proposed SVM-MIL formulations are impractical and a variety of heuristic approaches are presented. Particularly, these algorithms either converge to local optimal solutions, or are incapable of returning optimal solutions within given time limits. Our proposed LP and QP formulations have an advantage over the previous proposals since they are efficiently solvable to optimality even for datasets with large number of bags and instances. Still, it is more challenging to retrieve optimal solutions by commercial LP and QP solvers for larger benchmarks, such as PASCAL VOC 2007 [69]. As a future work, we can improve the scalability of solution procedure by solving mathematical models on subsets of the original data. The idea is to train multiple models on repeated bootstrap samples, and then report the average results of multiple classifiers. This approach is known as bagging in machine learning literature, in which a collection of diversified solutions provides higher accuracy with reduced variance [119].

Optimization-based algorithms provide practical and tractable tools for the area of data mining applications under MIL framework. Mathematical models of MIL problem can also be based on graph theoretical learning models. Using relationships between instances or bags, we can construct a graph, in which each data item is a node and relationships between the objects is represented by an edge between corresponding nodes. Several MIL studies benefit from graph representation of bags [56–60]. After learning graph-based bag representations, SVM classification is performed in [56], whereas non-convex optimization problems are established in [57]. Traditional learning approaches are adapted after deriving multi-graph representations of bags [58, 59]. In [60], multi-graph representation is obtained for bag classification by iteratively optimizing to selection of discriminative subgraph features. Success of bag classification highly sensitive to graph construction. In a well-studied area of graph-theoretical data mining, spectral clustering [120], popular ways of generating graph-based representation are a k-nearest-neighbor graph and a fully connected graph. The aim is to model nonlinear relationships between the instances as graph partitioning in spectral clustering. Similarly, a graph is constructed by unipartite generalized matching for semi-supervised learning (SSL) [121]. SSL problem is modeled as a constrained maximum-cut problem in [122]. As a future research direction, one may model and solve optimization problems of MIL on purposely-designed graphs. Leveraging the previous effort on graph-theoretical data mining, MIL problem can be formulated as extensions of classic graph theory problems such as maximum-cut and matching.

In MIL, there is an uncertainty on instance labels, which complicates the utilization of traditional supervised algorithms for MIL. Stochastic programming models can be derived to deal with such difficulties. In [123], multiple kernel learning problem with noisy labels is considered. The authors defined a binary random variable for each instance, which indicates whether the class estimate of a noisy instance is correct or not. These variables are utilized to form a chance constraint of the stochastic programming model, which is a relaxed version of a deterministic constraint. Hence, another possible future research direction is to reformulate the MIL optimization model as a stochastic program by introducing chance constraints to model uncertainties on the instance labels.

REFERENCES

1. Weston, J., C. Leslie, E. Ie, D. Zhou, A. Elisseeff and W. S. Noble, "Semi-supervised protein classification using cluster kernels", *Bioinformatics*, Vol. 21, No. 15, pp. 3241–3247, 2005.

2. Zhou, Z.-H., D.-C. Zhan and Q. Yang, "Semi-supervised learning with very few labeled training examples", *Proceedings of the National Conference on Artificial Intelligence*, Vol. 22, p. 675, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

3. Hoi, S. C., R. Jin, J. Zhu and M. R. Lyu, "Semi-supervised SVM batch mode active learning for image retrieval", *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on*, pp. 1–7, IEEE, 2008.

4. Zhou, Z.-H. and J.-M. Xu, "On the relation between multi-instance learning and semi-supervised learning", *Proceedings of the 24th international Conference on Machine Learning*, pp. 1167–1174, ACM, 2007.

5. Dietterich, T. G., R. H. Lathrop and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles", *Artificial Intelligence*, Vol. 89, No. 1, pp. 31–71, 1997.

6. Andrews, S., I. Tsochantaridis and T. Hofmann, "Support vector machines for multiple-instance learning", *Advances in Neural Information Processing Systems*, pp. 577–584, 2003.

7. Andrews, S., I. Tsochantaridis and T. Hofmann, "Support vector machines for multiple-instance learning", *Advances in Neural Information Processing Systems 15*, pp. 561–568, MIT Press, 2003.

8. Chen, Y., J. Bi and J. Z. Wang, "MILES: Multiple-instance learning via embedded

instance selection", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 28, No. 12, pp. 1931–1947, 2006.

9. Kandemir, M., C. Zhang and F. A. Hamprecht, "Empowering multiple instance histopathology cancer diagnosis by cell graphs", *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*, pp. 228–235, Springer, 2014.

10. Briggs, F., B. Lakshminarayanan, L. Neal, X. Z. Fern, R. Raich, S. J. Hadley, A. S. Hadley and M. G. Betts, "Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach", *The Journal of the Acoustical Society of America*, Vol. 131, No. 6, pp. 4640–4650, 2012.

11. Tao, Q., S. Scott, N. Vinodchandran and T. T. Osugi, "SVM-based generalized multiple-instance learning via approximate box counting", *Proceedings of the 21st International Conference on Machine Learning*, p. 101, ACM, 2004.

12. Murray, J. F., G. F. Hughes and K. Kreutz-Delgado, "Machine learning methods for predicting failures in hard drives: A multiple-instance application", *Journal of Machine Learning Research*, Vol. 6, No. May, pp. 783–816, 2005.

13. Maron, O. and T. Lozano-Pérez, "A framework for multiple-instance learning", *Advances in Neural Information Processing Systems*, pp. 570–576, 1998.

14. Mandel, M. I. and D. P. Ellis, "Multiple-Instance Learning for Music Information Retrieval.", *ISMIR*, pp. 577–582, 2008.

15. Zafra, A., C. Romero and S. Ventura, "Multiple instance learning for classifying students in learning management systems", *Expert Systems with Applications*, Vol. 38, No. 12, pp. 15020–15031, 2011.

16. Quellec, G., G. Cazuguel, B. Cochener and M. Lamard, "Multiple-instance learning for medical image and video analysis", *IEEE Reviews in Biomedical Engineering*, Vol. 10, pp. 213–234, 2017.

17. Lai, K.-T., F. X. Yu, M.-S. Chen and S.-F. Chang, "Video event detection by inferring temporal instance labels", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2243–2250, 2014.

18. Zhang, D., D. Meng and J. Han, "Co-saliency detection via a self-paced multiple-instance learning framework", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 39, No. 5, pp. 865–878, 2017.

19. Delaitre, V., J. Sivic and I. Laptev, "Learning person-object interactions for action recognition in still images", *Advances in Neural Information Processing Systems*, pp. 1503–1511, 2011.

20. Weidmann, N., E. Frank and B. Pfahringer, "A two-level learning method for generalized multi-instance problems", *European Conference on Machine Learning (ECML), 2003*, pp. 468–479, Springer, 2003.

21. Foulds, J. and E. Frank, "A review of multi-instance learning assumptions", *The Knowledge Engineering Review*, Vol. 25, No. 01, pp. 1–25, 2010.

22. Amores, J., "Multiple instance classification: Review, taxonomy and comparative study", *Artificial Intelligence*, Vol. 201, pp. 81–105, 2013.

23. Zhou, Z.-H., "Multi-instance learning: A survey", *Department of Computer Science & Technology, Nanjing University, Tech. Rep*, 2004.

24. Alpaydın, E., V. Cheplygina, M. Loog and D. M. Tax, "Single-vs. multiple-instance classification", *Pattern Recognition*, Vol. 48, No. 9, pp. 2831–2838, 2015.

25. Chen, Y. and J. Z. Wang, "Image categorization by learning and reasoning with regions", *The Journal of Machine Learning Research*, Vol. 5, pp. 913–939, 2004.

26. Cheplygina, V., D. M. Tax and M. Loog, "Multiple instance learning with bag dissimilarities", *Pattern Recognition*, Vol. 48, No. 1, pp. 264–275, 2015.

27. Erdem, A. and E. Erdem, "Multiple-instance learning with instance selection via dominant sets", *Similarity-Based Pattern Recognition*, pp. 177–191, Springer, 2011.

28. Fu, Z., A. Robles-Kelly and J. Zhou, "MILIS: Multiple instance learning with instance selection", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 33, No. 5, pp. 958–977, 2011.

29. Li, W.-J. and D.-Y. Yeung, "MILD: Multiple-instance learning via disambiguation", *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 22, No. 1, pp. 76–89, 2010.

30. Zhang, Q. and S. A. Goldman, "EM-DD: An improved multiple-instance learning technique", *Advances in Neural Information Processing Systems*, pp. 1073–1080, 2001.

31. Kim, M. and F. Torre, "Gaussian processes multiple instance learning", *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pp. 535–542, 2010.

32. Bunescu, R. C. and R. J. Mooney, "Multiple instance learning for sparse positive bags", *Proceedings of the 24th International Conference on Machine Learning*, pp. 105–112, ACM, 2007.

33. Li, Y.-F., J. Kwok, I. Tsang and Z.-H. Zhou, "A convex method for locating regions of interest with multi-instance learning", *Machine Learning and Knowledge Discovery in Databases*, pp. 15–30, 2009.

34. Mangasarian, O. L. and E. W. Wild, "Multiple instance classification via successive linear programming", *Journal of Optimization Theory and Applications*, Vol. 137, No. 3, pp. 555–568, 2008.

35. Kundakcioglu, O. E., O. Seref and P. M. Pardalos, "Multiple instance learning

via margin maximization", *Applied Numerical Mathematics*, Vol. 60, No. 4, pp. 358–369, 2010.

36. Poursaeidi, M. H. and O. E. Kundakcioglu, "Robust support vector machines for multiple instance learning", *Annals of Operations Research*, Vol. 216, No. 1, pp. 205–227, 2014.

37. Doran, G. and S. Ray, "A theoretical and empirical analysis of support vector machine methods for multiple-instance classification", *Machine Learning*, Vol. 97, No. 1-2, pp. 79–102, 2014.

38. Li, M., J. T. Kwok and B.-L. Lu, "Online multiple instance learning with no regret", *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1395–1401, IEEE, 2010.

39. Gärtner, T., P. A. Flach, A. Kowalczyk and A. J. Smola, "Multi-Instance Kernels.", *Proceedings of the 19th International Conference on Machine Learning (ICML)*, Vol. 2, pp. 179–186, 2002.

40. Cheung, P.-M. and J. T. Kwok, "A regularization framework for multiple-instance learning", *Proceedings of the 23rd International Conference on Machine Learning*, pp. 193–200, ACM, 2006.

41. Blockeel, H., D. Page and A. Srinivasan, "Multi-instance tree learning", *Proceedings of the 22nd International Conference on Machine Learning*, pp. 57–64, ACM, 2005.

42. Auer, P. and R. Ortner, *A boosting approach to multiple instance learning*, pp. 63–74, Springer, 2004.

43. Zhang, C., J. C. Platt and P. A. Viola, "Multiple instance boosting for object detection", *Advances in Neural Information Processing Systems*, pp. 1417–1424, 2005.

44. Wen, L., Z. Cai, M. Yang, Z. Lei, D. Yi and S. Z. Li, "Online multiple instance joint model for visual tracking", *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*, pp. 319–324, IEEE, 2012.

45. Leistner, C., A. Saffari and H. Bischof, "MIForests: Multiple-instance learning with randomized trees", *Computer Vision–ECCV 2010*, pp. 29–42, 2010.

46. Wang, J. and J.-D. Zucker, "Solving the multiple-instance problem: A lazy learning approach", *In Proc. 17th International Conf. on Machine Learning*, pp. 1119–1125, 2000.

47. Yuan, L., L. Zhao and H. Xu, "Multi-instance learning via instance-based and bag-based representation transformations", *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 2771–2775, IEEE, 2015.

48. Cheplygina, V., D. M. Tax and M. Loog, "Dissimilarity-based Ensembles for Multiple Instance Learning", *IEEE Transactions on Neural Networks and Learning Systems*, 2015.

49. Zhou, Z.-H. and M.-L. Zhang, "Solving multi-instance problems with classifier ensemble based on constructive clustering", *Knowledge and Information Systems*, Vol. 11, No. 2, pp. 155–170, 2007.

50. Wei, X.-S., J. Wu and Z.-H. Zhou, "Scalable algorithms for multi-instance learning", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28, No. 4, pp. 975–987, 2017.

51. Wang, X., B. Wang, X. Bai, W. Liu and Z. Tu, "Max-margin multiple-instance dictionary learning", *Proceedings of the 30th International Conference on Machine Learning*, pp. 846–854, 2013.

52. Huo, J., Y. Gao, W. Yang and H. Yin, "Abnormal event detection via multi-instance dictionary learning", *Intelligent Data Engineering and Automated*

Learning-IDEAL 2012, pp. 76–83, Springer, 2012.

53. Song, X., L. Jiao, S. Yang, X. Zhang and F. Shang, "Sparse coding and classifier ensemble based multi-instance learning for image categorization", *Signal Processing*, Vol. 93, No. 1, pp. 1–11, 2013.

54. Shrivastava, A., J. K. Pillai, V. M. Patel and R. Chellappa, "Dictionary-based multiple instance learning", *Image Processing (ICIP), 2014 IEEE International Conference on*, pp. 160–164, IEEE, 2014.

55. Carbonneau, M.-A., E. Granger, A. J. Raymond and G. Gagnon, "Robust multiple-instance learning ensembles using random subspace instance selection", *Pattern Recognition*, Vol. 58, pp. 83–99, 2016.

56. Zhou, Z.-H., Y.-Y. Sun and Y.-F. Li, "Multi-instance learning by treating instances as non-iid samples", *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1249–1256, ACM, 2009.

57. Zhang, D., Y. Liu, L. Si, J. Zhang and R. D. Lawrence, "Multiple instance learning on structured data", *Advances in Neural Information Processing Systems*, pp. 145–153, 2011.

58. Wu, J., X. Zhu, C. Zhang and S. Y. Philip, "Bag constrained structure pattern mining for multi-graph classification", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 10, pp. 2382–2396, 2014.

59. Wu, J., S. Pan, X. Zhu and Z. Cai, "Boosting for multi-graph classification", *IEEE Transactions on Cybernetics*, Vol. 45, No. 3, pp. 416–429, 2015.

60. Wu, J., S. Pan, X. Zhu, C. Zhang and X. Wu, "Positive and unlabeled multi-graph learning", *IEEE Transactions on Cybernetics*, Vol. 47, No. 4, pp. 818–829, 2017.

61. Shrivastava, A., V. M. Patel, J. K. Pillai and R. Chellappa, "Generalized dictio-

naries for multiple instance learning", *International Journal of Computer Vision*, Vol. 114, No. 2-3, pp. 288–305, 2015.

62. Carrizosa, E. and D. R. Morales, "Supervised classification and mathematical optimization", *Computers & Operations Research*, Vol. 40, No. 1, pp. 150–165, 2013.

63. Kucukasci, E. S. and M. G. Baydogan, *Bag-level Representations for Multiple Instance Learning*, 2018, `http://ww3.ticaret.edu.tr/eskucukasci/multiple-instance-learning/`, accessed at September 2018.

64. Breiman, L., J. Friedman, C. J. Stone and R. A. Olshen, *Classification and regression trees*, CRC Press, 1984.

65. Quinlan, J. R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

66. Breiman, L., "Random forests", *Machine Learning*, Vol. 45, No. 1, pp. 5–32, 2001.

67. Schölkopf, B., C. Burges and V. Vapnik, "Extracting Support Data for a Given Task", *Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park*, Citeseer, 1995.

68. Baydogan, M. G., G. Runger and E. Tuv, "A Bag-of-Features Framework to Classify Time Series", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 35, No. 11, pp. 2796–2802, 2013.

69. Everingham, M., L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge", *International Journal of Computer Vision*, Vol. 88, No. 2, pp. 303–338, 2010.

70. Coates, A. and A. Y. Ng, "Learning Feature Representations with K-Means", *Neural Networks: Tricks of the Trade: Second Edition*, pp. 561–580, Springer Berlin Heidelberg, 2012.

71. Küçükaşcı, E. Ş. and M. G. Baydoğan, "Bag encoding strategies in multiple instance learning problems", *Information Sciences*, Vol. 467, pp. 559–578, 2018.

72. Vens, C. and F. Costa, "Random forest based feature induction", *2011 IEEE 11th International Conference on Data Mining*, pp. 744–753, IEEE, 2011.

73. Moosmann, F., E. Nowak and F. Jurie, "Randomized clustering forests for image classification", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 30, No. 9, pp. 1632–1646, 2008.

74. Criminisi, A., J. Shotton, E. Konukoglu *et al.*, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning", *Foundations and Trends® in Computer Graphics and Vision*, Vol. 7, No. 2–3, pp. 81–227, 2012.

75. Qiao, M., L. Liu, J. Yu, C. Xu and D. Tao, "Diversified dictionaries for multi-instance learning", *Pattern Recognition*, Vol. 64, pp. 407–416, 2017.

76. Haußmann, M., F. A. Hamprecht and M. Kandemir, "Variational Bayesian Multiple Instance Learning with Gaussian Processes", *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

77. Salzberg, S. L., "On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach", *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, pp. 317–328, Sep 1997.

78. R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2018.

79. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, Vol. 12, No. Oct, pp. 2825–2830, 2011.

80. Majnik, M. and Z. Bosnić, "ROC analysis of classifiers in machine learning: A survey", *Intelligent Data Analysis*, Vol. 17, No. 3, pp. 531–558, 2013.

81. Tax, D. M. and R. P. Duin, "Learning curves for the analysis of multiple instance classifiers", *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 724–733, Springer, 2008.

82. Zhou, Z.-H., K. Jiang and M. Li, "Multi-instance learning based web mining", *Applied Intelligence*, Vol. 22, No. 2, pp. 135–147, 2005.

83. Duin, R., *PRtools. Version 4.1.5.*, 2009.

84. Tax, C. V., D.M.J., *MIL, A Matlab Toolbox for Multiple Instance Learning*, Jun 2015, version 1.1.0.

85. The Mathworks, I., *MATLAB version 8.5.0.197613 (R2015a)*, Natick, Massachusetts, 2015.

86. Srinivasan, A., S. Muggleton and R. King, "Comparing the use of background knowledge by inductive logic programming systems", *Proceedings of the 5th International Workshop on Inductive Logic Programming*, pp. 199–230, Department of Computer Science, Katholieke Universiteit Leuven, 1995.

87. Demšar, J., "Statistical comparisons of classifiers over multiple data sets", *J. Mach. Learn. Res.*, Vol. 7, pp. 1–30, 2006.

88. Friedman, M., "A Comparison of Alternative Tests of Significance for the Problem of m Rankings", *The Annals of Mathematical Statistics*, Vol. 11, No. 1, pp. pp. 86–92, 1940.

89. Nemenyi, P., *Distribution-free Multiple Comparisons*, Princeton University, 1963.

90. Scott, S., J. Zhang and J. Brown, "On generalized multiple-instance learning", *International Journal of Computational Intelligence and Applications*, Vol. 5,

No. 01, pp. 21–35, 2005.

91. Xu, X., *Statistical learning in multiple instance problems*, Ph.D. Thesis, The University of Waikato, 2003.

92. Gehler, P. V. and O. Chapelle, "Deterministic annealing for multiple-instance learning", *International Conference on Artificial Intelligence and Statistics*, pp. 123–130, 2007.

93. Küçükaşcı, E. Ş., M. G. Baydoğan and Z. C. Taşkın, "A Linear Programming Approach to Multiple Instance Learning", *Technical Report*, 2018.

94. Vanwinckelen, G., D. Fierens, H. Blockeel *et al.*, "Instance-level accuracy versus bag-level accuracy in multi-instance learning", *Data Mining and Knowledge Discovery*, Vol. 30, No. 2, pp. 313–341, 2016.

95. Tax, D. M., E. Hendriks, M. F. Valstar and M. Pantic, "The detection of concept frames using clustering multi-instance learning", *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 2917–2920, IEEE, 2010.

96. Wu, J., Y. Zhao, J.-Y. Zhu, S. Luo and Z. Tu, "Milcut: A sweeping line multiple instance learning paradigm for interactive image segmentation", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 256–263, 2014.

97. Holst, A. *et al.*, "Efficient AUC maximization with regularized least-squares", *Tenth Scandinavian Conference on Artificial Intelligence: SCAI 2008*, Vol. 173, p. 12, IOS Press, 2008.

98. Mann, H. B. and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other", *The Annals of Mathematical Statistics*, pp. 50–60, 1947.

99. Ataman, K., W. Streetr and Y. Zhang, "Learning to rank by maximizing AUC with linear programming", *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pp. 123–129, IEEE, 2006.

100. Duin, R. P. *et al.*, *The dissimilarity representation for pattern recognition: foundations and applications*, Vol. 64, World Scientific, 2005.

101. Li, Z., G.-H. Geng, J. Feng, J.-y. Peng, C. Wen and J.-l. Liang, "Multiple instance learning based on positive instance selection and bag structure construction", *Pattern Recognition Letters*, Vol. 40, pp. 19–26, 2014.

102. Gurobi Optimization, I., *Gurobi Optimizer Reference Manual*, 2018.

103. Ling, C. X., J. Huang and H. Zhang, "AUC: a better measure than accuracy in comparing learning algorithms", *Advances in Artificial Intelligence*, pp. 329–341, Springer, 2003.

104. Nelder, J. A. and R. Mead, "A simplex method for function minimization", *The Computer Journal*, Vol. 7, No. 4, pp. 308–313, 1965.

105. Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "LIBLINEAR: A library for large linear classification", *Journal of Machine Learning Research*, Vol. 9, No. Aug, pp. 1871–1874, 2008.

106. Chang, C.-C. and C.-J. Lin, "LIBSVM: a library for support vector machines", *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 2, No. 3, p. 27, 2011.

107. Carbonneau, M.-A., V. Cheplygina, E. Granger and G. Gagnon, "Multiple instance learning: A survey of problem characteristics and applications", *Pattern Recognition*, 2017.

108. Fu, Z., G. Lu, K. M. Ting and D. Zhang, "Learning sparse kernel classifiers

for multi-instance classification", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 24, No. 9, pp. 1377–1389, 2013.

109. Fischetti, M., "Fast training of support vector machines with Gaussian kernel", *Discrete Optimization*, Vol. 22, pp. 183–194, 2016.

110. Şeref, O., W. A. Chaovalitwongse and J. P. Brooks, "Relaxing support vectors for classification", *Annals of Operations Research*, Vol. 216, No. 1, pp. 229–255, 2014.

111. Küçükaşcı, E. Ş., M. G. Baydoğan and Z. C. Taşkın, "Multiple Instance Classification via Quadratic Programming", *Technical Report*, 2018.

112. Geoffrion, A. M., "Generalized Benders decomposition", *Journal of Optimization Theory and Applications*, Vol. 10, No. 4, pp. 237–260, 1972.

113. Taşkın, Z. C., "Benders decomposition", *Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Malden (MA)*, 2010.

114. Ketchen Jr, D. J. and C. L. Shook, "The application of cluster analysis in strategic management research: an analysis and critique", *Strategic Management Journal*, pp. 441–458, 1996.

115. Huang, J. and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 3, pp. 299–310, 2005.

116. Kucukasci, E. S. and M. G. Baydogan, *Multiple Instance Learning Repository*, 2018, `http://www.multipleinstancelearning.com/`, accessed at September 2018.

117. Chen, J., K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng and K. Li, "A parallel random forest algorithm for big data in a spark cloud computing environment", *IEEE*

Transactions on Parallel & Distributed Systems, Vol. 1, pp. 1–1, 2017.

118. Vapnik, V., *Statistical learning theory. 1998*, Vol. 3, Wiley, New York, 1998.

119. Breiman, L., "Bagging predictors", *Machine Learning*, Vol. 24, No. 2, pp. 123–140, 1996.

120. Ng, A. Y., M. I. Jordan and Y. Weiss, "On spectral clustering: Analysis and an algorithm", *Advances in neural information processing systems*, pp. 849–856, 2002.

121. Jebara, T., J. Wang and S.-F. Chang, "Graph construction and b-matching for semi-supervised learning", *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 441–448, ACM, 2009.

122. Wang, J., T. Jebara and S.-F. Chang, "Semi-supervised learning using greedy max-cut", *Journal of Machine Learning Research*, Vol. 14, No. Mar, pp. 771–800, 2013.

123. Yang, T., M. Mahdavi, R. Jin, L. Zhang and Y. Zhou, "Multiple kernel learning from noisy labels by stochastic programming", *arXiv preprint arXiv:1206.4629*, 2012.