MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives

Gabriella Melki[a], Alberto Cano[a], Sebastián Ventura[b,c]

[a]*Department of Computer Science, Virginia Commonwealth University, USA*
[b]*Department of Computer Science and Numerical Analysis, University of Cordoba, Spain*
[c]*Department of Computer Science, King Abdulaziz University, Saudi Arabia Kingdom*

Abstract

Multiple-instance learning (MIL) is a variation of *supervised learning*, where samples are represented by labeled bags, each containing sets of instances. The individual labels of the samples within a bag are unknown, and labels are assigned based on a multi-instance assumption. One of the major complexities associated with this type of learning is the ambiguous relationship between a bag's label and the instances it contains. This paper proposes a novel support vector machine (SVM) multiple-instance formulation and presents an algorithm with a bag-representative selector that trains the SVM based on bag-level information, named MIRSVM. The contribution is able to identify instances that highly impact classification, i.e. bag-representatives, for both positive and negative bags, while finding the optimal class separation hyperplane. Unlike other multi-instance SVM methods, this approach eliminates possible class imbalance issues by allowing both positive and negative bags to have at most one representative, while highlighting the instances that contribute most to the classification model. The experimental study evaluates and compares the performance of this proposal against 10 state-of-the-art multi-instance methods over 15 datasets, and the results are validated through non-parametric statistical analysis. The results indicate that bag-based learners outperform the instance-based and wrapper methods, as well as this proposal's overall superior performance against other multi-instance SVM models.

Keywords: Machine learning, multiple-instance learning, support vector machines, bag-level multi-instance classification, bag-representative selection

1. Introduction

In traditional classification, learning algorithms attempt to correctly label unknown samples by finding patterns that exist between training samples and their class label. *Multi-instance* (MI) learning (or *multiple-instance* learning) is a variation of *supervised learning* that has been recently been gaining interest because of its applicability to many real-world problems such as text categorization [4], image classification and annotation [30, 43], human action recognition [55], and drug activity prediction [18].

The difference between MIL and traditional learning is the nature of the data samples. In the traditional setting, each sample is represented by a single feature vector that has its own class label. In multi-instance learning, a sample is considered to be a *bag* that contains multiple instances, and is associated with a single label. The individual labels of the samples within a bag, referred to as *instances*, are unknown, and labels are assigned based on a multi-instance assumption, or

hypothesis. Introduced by [18], the standard MI assumption states that a bag is labeled as positive if and only if it contains at least one positive instance. More recently, other MI hypotheses and frameworks have been proposed by [22] to encompass a wider range of applications with multi-instance data.

One of the major complexities associated with multi-instance learning is the ambiguity of the relationship between a bag label and instances within the bag [2]. This stems from the standard MIL assumption, where the underlying distribution among instances within positive bags is unknown. All that is known is that at least one instance has a positive label within a positive bag. There have been different attempts to overcome this complexity. One approach involves "flattening" the MIL datasets, meaning samples contained in positive bags each adopt a positive label, allowing the use of classical supervised learning techniques [44]. This approach assumes that positive bags contain a significant number of positive samples, which may not be the case, causing the classifier to mislabel negative instances within the bag, decreasing the power of the resulting MIL model.

To overcome this, a different MIL approach was proposed, where subsets of instances are selected from positive bags for classifier training [38, 56]. One drawback of this type of method is that the resulting training datasets become imbalanced towards positive instances by significantly decreasing the number of negative instances. The performance of these methods deteriorates when more instances are selected as subsets than needed [13, 21, 47]. Our proposal aims to deal with this drawback by minimizing class imbalance. This is achieved by optimally selecting bag-representatives from both classes using Support Vector Machines.

Support Vector Machines (SVMs) represent a set of supervised, linear and nonlinear, classification and regression methods that have theoretical foundations on Vapnik-Chervonenkis (VC) theory [33, 45]. SVM models are similar to other machine learning algorithms and techniques, but research has shown that they usually outperform them in terms of computational efficiency, scalability, and robustness against outliers [35, 40], which makes them a useful data mining tool for various real-world applications.

To address the limitations presented by MIL algorithms, this paper proposes a novel support vector machine formulation with a bag-representative selector, called Multiple-Instance Representative Support Vector Machine (MIRSVM). SVMs are known to perform well when data is limited, therefore combining them with a bag-representative selector aims to remedy class imbalance caused by limited positive bags, without assuming their internal distributions. The algorithm selects bag-representatives iteratively according to the standard MI assumption, ensuring known information, such as the negative bag labels, are fully utilized during the training process. The optimal separating hyperplane between bags is then found with respect to the bag-representatives using a Gaussian kernel and a quadratic programming solver. The optimal set of representatives is found when they have stopped changing from one iteration to the next. The algorithm does not assume any distribution of the instances and is not affected by the number of samples within a bag, making it applicable to a variety of contexts. The key contributions of this work include,

- Reformulating the traditional primal L1-SVM problem to optimize over bags, rather than instances, ensuring all the information contained within each bag is utilized during training, while defining bag representative selector criteria.

- Deriving the dual multi-instance SVM problem, with the Karush-Kuhn-Tucker necessary and sufficient conditions for optimality. The dual is maximized with respect to the Lagrange

multipliers and provides insightful information about the resulting sparse model. The dual formulation is kernelized with a Gaussian radial basis function, which calculates the distances between bag representatives.

- Devising a unique bag-representative selection method that makes no presumptions about the underlying distributions of the instances within each bag, while maintaining the default MIL assumption. This approach eliminates the issue of class imbalance caused by techniques such as flattening or subsetting positive instances from each bag. The key feature of MIRSVM is its ability to identify instances (support vectors) within positive and negative bags that highly impact the model.

This work is organized as follows. First, the notation used throughout the paper is provided, the MIL problem is formalized, and recent multiple-instance learning methods, traditional support vector machines, and state-of-the-art multi-instance SVM methods are reviewed in Section 2. The primal L1-SVM is formulated with respect to optimizing over bags, the dual formulation is then derived, along with the conditions for optimality, and the proposed algorithm, MIRSVM, is listed and described in Section 3. Section 4 presents the experimental environment, including results from various metrics, run-time, and non-parametric statistical analysis over 15 benchmark MI datasets compared with 10 state-of-the-art algorithms. Finally, Section 5 presents the conclusions of this contribution.

2. Background

This section defines the notation that will be used throughout the paper, and reviews related works on multi-instance learning, traditional support vector machines, and multi-instance support vector machines.

2.1. Notation

Let $\mathcal{D}$ be a training dataset of n bags. Let $\boldsymbol{Y} \in \mathcal{D}$ be a vector of n labels corresponding to each bag, having a domain of $\boldsymbol{Y} \in \{-1, +1\}^n$. Let $\boldsymbol{X} \in \mathcal{D}$ be a matrix consisting of d input variables and m instances, having a domain of $\boldsymbol{X} \in \mathbb{R}^{m \times d}$. Let $\mathcal{B}$ be the set of bags which contain a number of instances, sometimes of different size, and usually non-overlapping, such as $\boldsymbol{B}_I = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\boldsymbol{B}_I|}\}$ for index set $I \subseteq \{1, \ldots, n\}$, where $|\boldsymbol{B}_I|$ is the number of instances in bag I. Using the training dataset $\mathcal{D} = \{(\boldsymbol{B}_1, Y_1), \ldots, (\boldsymbol{B}_n, Y_n)\}$, the goal is to learn a model $f : \boldsymbol{X} \times Y$, that assigns a label, $Y_I \in \{-1, +1\}$, for each input bag $\boldsymbol{B}_I$. The model will then be used to predict the labels of new, unlabeled, and unseen input bags. Table 1 provides a summary of the notation used in this paper.

2.2. Multiple-Instance Learning

In traditional binary classification problems, the goal is to learn a model that maps input samples to labels, $f : \mathbb{R}^{m \times d} \to Y^m \in \{-1, +1\}$. Multiple instance learning generalizes this framework and makes weaker assumptions about sample labeling. In the MIL case, samples are called *bags* and each bag contains one or more input instances. Each bag is assigned a label, unlike traditional classification problems, where individual samples are assigned a label. Input instances, $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, are grouped into bags with unique identifiers, $\mathcal{B} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_n \,|\, \boldsymbol{B}_I = \{\boldsymbol{x}_i \,|\, \forall i \in I\}\}$,

Table 1: Summary of notation used throughout the paper

Definition	Notation
Number of Bags	n
Number of Instances	m
Number of Input Attributes	d
Set of Bags	$\mathcal{B} = \{\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n\}$
Bag Index Set	$I \in \mathbb{Z}_+^n$
Input Space	$\boldsymbol{X} \in \mathbb{R}^{m \times d}$
Bag Labels	$\boldsymbol{Y} \in \{-1, 1\}^n$
Input Instance i from Bag I	$\boldsymbol{x}_i = (x_1, \ldots, x_d),\ i \in I$
Individual Instance Label i	$y_i \in \{-1, +1\}$
Bag I	$\boldsymbol{B}_I = \{\boldsymbol{x}_i \,\vert\, \forall i \in I\}$
Full Multi-Instance Training Dataset	$\mathcal{D} = \{(\boldsymbol{B}_1, Y_1), \ldots, (\boldsymbol{B}_n, Y_n)\}$
Full Single-Instance Training Dataset	$\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

$\forall I \in \{1, \ldots, n\}$ and assigned a label, Y_I. An example representation of an MI dataset is shown in Figure 1.

In MIL, the goal is to train a classifier that predicts the label of an unseen bag, $f(\boldsymbol{B}_{n+1}) \rightarrow Y_{n+1}$ [3]. In order to build a classifier without any knowledge of the individual training instance labels, [18] proposed the *standard MI* (SMI) hypothesis based on the domain of drug-activity prediction, shown in Equation 1, which states that a bag is labeled positive if and only if at least one of the instances in the bag is positive, and is labeled negative otherwise.

$$Y_I = \begin{cases} +1 & \text{if } \exists\, y_i = +1,\ \forall i \in I \\ -1 & \text{otherwise.} \end{cases} \tag{1}$$

This implies that individual instance labels y_i exist, but are not known during training. Equation 1 can also be rewritten as Equation 2 for simplicity:

$$Y_I = \max_{\forall i \in I}\ y_i. \tag{2}$$

In addition to the SMI assumption, alternative MI assumptions have been proposed to date [22]. A recent review describing the taxonomy of multi-instance classification was presented by [3]. The review presents various methods and algorithms used in literature which are categorized based on their approach to handling the MI input space. Instance-based classifiers that fall under the *instance-space paradigm*, aim to separate instances in positive bags from those in negative ones. Bag-level classifiers (*bag-space paradigm*) treat each bag as a whole entity, implicitly extracting information from each bag in order to accurately predict their labels. Methods that fall under the *embedded-space paradigm* map the input bags to a single feature vector that explicitly encapsulates all the relevant information contained within each bag.

Instance-based methods that follow the SMI assumption attempt to identify desirable instance properties that make a bag positive. One traditional method in this category is the Axis-Parallel Rectangle (APR) [18], which trains a model that assigns a positive label to an instance if it belongs

$\mathcal{B}$	F_1	F_2	F_3	...	F_d	$\boldsymbol{Y}$
B_1	0.1	0.8	2.5	...	0.8	POS
	0.2	2.0	5.5	...	3.0	
	0.1	0.1	4.5	...	0.1	
B_2	1.5	4.0	0.8	...	0.1	NEG
	0.8	0.4	2.9	...	1.1	
	2.3	0.2	4.0	...	5.5	
	6.7	5.0	0.1	...	0.5	
	0.1	4.0	8.7	...	3.3	

Figure 1: Example of multi-instance data representation. The figure shows a sample binary MI dataset with 2 bags, identified as B_1 and B_2. The bags each contain differing number of instances, each with d dimension, and are assigned positive and negative labels indicated by column $\boldsymbol{Y}$.

to an axis-parallel rectangle in feature space, and assigns a negative label otherwise. The APR is optimized by maximizing the number of positive bags in the training set containing at least one instance in the APR, while concurrently maximizing the number of negative bags that do not contain any instance in the APR. Another similar method is the Diverse Density (DD) [38] framework which is maximized for instances in feature space that are near at least one instance in a positive bag and far from all instances in negative bags. In the Expectation-Maximization Diverse Density (EM-DD) algorithm, [56] propose a similar framework that iteratively maximizes the DD measure. [5] present a boosting approach that uses balls centered around positive bags to solve the MI problem called Multi-Instance Optimal Ball (MIOptimalBall). This approach is similar to that of APR and DD, except that [5] propose computing optimal balls per positive bags. A major challenge affecting these methods is that the distributions of the positive and negative bags affect their performance. Methods based on the DD metric [14, 15, 13, 43] assume the positive instances form a cluster, which may not be the case. Alternatively, [23] models the distribution of negative bags with Gaussian kernels, which can prove difficult when the quantity of data is limited.

Some methods in literature [44, 48], such as Simple-MI [19], transform the MI dataset to a traditional instance-level dataset. Simple-MI represents each bag with the mean vector of the instances within it. This approach was evaluated by [11] and they proposed mapping each bag to a max-min vector, a concatenation of the features with the highest and lowest values. The major disadvantage of these types of approaches is that they assume the distribution the instances in positive bags is positive, when it may not be.

An extension of traditional single-instance k-nearest neighbors method (KNN) was proposed by [49] to be applied to the bag-level, named CitationKNN. This method uses a distance function between bags in order to determine bag similarities. Not only are the set of closest bags to a single bag considered, but also how many bags is the single bag closest to. A voting scheme is then used to determine the bag class labels. [53] proposed a multi-instance Decision-Based Neural Network (MI-DBNN), a probabilistic variant of the traditional DBNN, which is a neural network with a modular structure [31].

[9] introduced the Multi-Instance Tree Inducer (MITI), based on the standard MI assumption, which uses decision trees as a heuristic to solve the MI problem. This approach aims to identify whether an instance within a bag is truly positive and eliminate false positives within the same bag. The disadvantage of this approach stems from removing instances considered as false positives

from partially grown trees without updating the existing tree structure. [7] then enhanced this approach by creating the method Multi-Instance Rule Induction (MIRI). The algorithm aims to eliminate any possibility of a suboptimal split because the tree is discarded and regrown. Other popular methods include MIBoost [29], MIWrapper [29], and Two-Level Classifier (TLC) [51].

For most of the methods described above, implicit or explicit assumptions have been made about the distribution of the data. Selecting a method that is robust for a problem such as MIL can be difficult when little is known about the nature of the data, especially considering the unknown distribution of the instances within bags [12, 39]. The proposed method, MIRSVM, is a general method that uses support vector machines to design a MIL model without making prior assumptions about the data. Classifiers of this type are known to provide better generalization capabilities and performance, as well as sparser models.

2.3. Support Vector Machines

Most classical machine learning techniques require knowledge of the data distribution in order to learn accurate models. This is a serious restriction because, in most cases, the distribution of the data is unknown. Another disadvantage of these methods stems from high dimensional, sparse datasets, which are very common in real-world applications. Small sample size also poses problems of model reliability, especially when coupled with high dimensional feature spaces. SVMs represent learning techniques that have been introduced under the *structural risk minimization* (SRM) framework and VC theory [34]. Rather than optimizing over L1 or L2 norms and classification error, SVMs perform SRM [46], minimizing the expected probability of classification error, resulting in a generalized model without making assumptions about the data distribution [16].

SVMs are a particularly useful for learning linear predictors in high dimensional feature spaces, which is a computationally complex learning problem. In the context of classification, this problem is approached by searching for the optimal *maximal margin* of separability between classes. A training set $\mathcal{D}$ is *linearly separable* if a halfspace exists, $(\boldsymbol{w}, b)$, such that $y_i = \text{sign}\,(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b), \forall i \in \{1, \ldots, m\}$, where $\boldsymbol{w} \in \mathbb{R}^d$ is a d-dimensional weight vector, and $b \in \mathbb{R}$ is a bias term. All halfspaces, defined as $d(\boldsymbol{x}_i) = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b$, satisfying $y_i\,(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) > 0,\ \forall i \in \{1, \ldots, m\}$, have no error. This hard constraint is associated with the *Hard-Margin SVM*. No feasible solution exists for the Hard-Margin SVM problem if the dataset is non-linearly separable, which is the case for most datasets. To overcome this, [16] introduced the *Soft-Margin L*1*-SVM*:

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \ \frac{1}{2}||\boldsymbol{w}||^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \tag{3}$$

$$\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i,\ \forall i \in \{1, \ldots, m\} \tag{3a}$$

$$\xi_i \geq 0,\ \forall i \in \{1, \ldots, m\}, \tag{3b}$$

where $C \in \mathbb{R}$ is the penalty parameter that controls the trade-off between margin maximization and classification error minimization, penalizing large norms and errors. The slack variable $\boldsymbol{\xi} \in \mathbb{R}^m$ allows for optimizing over sample errors. The resulting hyperplane is called *soft-margin hyperplane*. Although the Soft-SVM learns an optimal hyperplane, if the training data set is not linearly separable, the classifier learned may not have a good generalization capability [41]. Generalization and linear separability can be enhanced by mapping the original input space to a higher dimensional

dot-product space by using a kernel function shown in Equation 4:

$$\mathcal{K}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right) \rangle, \tag{4}$$

where $\phi\left(\cdot\right)$ represents a function mapping from the original feature space to a higher dimensional space. This kernel mapping is particularly helpful when solving the dual SVM optimization problem shown in Equation (5). Rather than calculating the inner product of two mapped vectors, their corresponding *scalar* kernel value can be used.

$$\max_{\boldsymbol{\alpha}} \quad \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \mathcal{K}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{m} \alpha_i \tag{5}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \tag{5a}$$

$$0 \leq \alpha_i \leq \frac{C}{m}, \ \forall i \in \{1, \ldots, m\}. \tag{5b}$$

A traditional and widely used method of solving the L1-SVM problem is the *Sequential Minimal Optimization* (SMO) technique proposed by [42]. It is an iterative procedure that divides the SVM dual problem into a series of sub-problems, which are then solved analytically by finding the optimal $\boldsymbol{\alpha}$ values that satisfy the Karush-Kuhn-Tucker (KKT) conditions [10]. Although SMO is guaranteed to converge, heuristics are used to choose $\boldsymbol{\alpha}$ values in order to accelerate the convergence rate. This is a critical step because the convergence speed of the SMO algorithm is highly dependent on the dataset size, as well as the SVM hyperparameters [45]. *Iterative Single Data Algorithm* (ISDA) [33, 35] is a more recent and efficient approach for solving the L2-SVM problem, shown to be faster than the SMO algorithm and equal in terms of accuracy [37, 36]. It iteratively updates the objective function by working on one data point at a time, using coordinate descent to find the optimal objective function value. Other methods for solving the SVM problem include *Quadratic Programming* solvers, such as the interior point method. These types of algorithms find the true optimal objective function value at the trade-off of having a relatively slower run-time.

2.4. Multiple-Instance Support Vector Machines

The MI adaptation of the SVM presents two contexts for solving the problem: the instance-level and the bag-level. The first tries to identify instances, either all within a bag or just key instances, that help find the optimal separating hyperplane between positive and negative bags. The latter uses kernels defined over whole bags to optimize the margin.

[4] proposed a mixed-integer quadratic program that solves the MI problem at an instance-level, using a support vector machine, named MISVM, that can be solved heuristically. Rather than maximizing the margin of separability between instances of different classes, this instance-based method maximizes the margin between bags. It tries to identify the key instance from each positive bag that makes the bag positive by assuming it has the highest margin value [31]. Instances from positive bags are selected as bag-representatives, and the algorithm iteratively creates a classifier that separates those representatives from all instances from the negative bags. Using bag-representatives from one class and all instances from the other is an example of an approach that combines rules from the SMI assumption and the collective assumption. This approach was used in

the context of active learning. [50] proposed a multi-criteria decision making system that measures the significance of unlabeled instances and selects the best instance iteratively using MISVM. A disadvantage of the approach MISVM takes, stems from the assumption that all instances within positive bags are also positive, which is an implicit step in the initialization of MISVM. [4] also proposed a second mixed-integer instance-level approach, named mi-SVM, which does not discard the negative instances of the positive bags. It rather tries to identify the instances within positive bags that are negative and utilize them in the construction of the negative margin. The main disadvantage of these approaches is that they create an imbalanced class problem that favors the negative class, resulting in a biased classifier.

[54] presented an instance-based SVM that uses an asymmetric loss function that follows the SMI assumption. The idea behind this loss function is that positive and negative misclassification costs are different. A false negative instance in a positive bag might not cause an error on the bag label, but a false positive instance within a negative bag would lead to a misclassification error. The algorithm ensures all negative instances are correctly classified while minimizing false positives.

One of the first bag-level approaches to the multi-instance SVM problem was proposed by [27], who defined a bag-level kernel. A bag-level kernel determines the similarity between two bags in a higher dimensional space and using such a kernel with the standard SVM problem, the margin can be optimized over bag classes without any modification to the SVM problem. [8] propose conformal kernels which manipulate each attribute's dimension based on its importance, without affecting the angles between vectors in the transformed space. Bag level kernels transform the bags into a single-instance representation which enables standard SVMs to be directly applied to multi-instance data [31].

3. Multiple-Instance Representative SVM

MIRSVM is based on the idea of selecting representative instances from both positive and negative bags which are used to find an unbiased, optimal separating hyperplane. A representative is iteratively chosen from each bag, and a new hyperplane is formed according to the representatives until they converge. Based on the SMI hypothesis, only one instance in a bag is required to be positive for the bag to adopt a positive label. Due to the unknown distribution of instances within positive bags, MIRSVM is designed to give preference to negative bags during training, because their distribution is known, i.e. all instances are guaranteed to be negative. This is evident during the representative selection process, by taking the maximum output value within each bag based on the current hyperplane using the following rule, $s_I = \max_{i \in I}(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b), \forall I \in \{1, \ldots, n\}$. In other words, the most positive sample is chosen from each positive bag and the least negative sample is chosen from each negative bag (samples with the largest output value based on the current hyperplane), pushing the decision boundary towards the positive bags. Equation (6) presents the primal MIRSVM optimization problem:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{w}||^2 + \frac{C}{n}\sum_I \xi_I, \tag{6}$$

$$\text{s.t. } y_I(\langle \boldsymbol{w}, \boldsymbol{x}_{s_I} \rangle + b) \geq 1 - \xi_I, \forall I \in \{1, \ldots, n\} \tag{6a}$$

$$\xi_I \geq 0, \forall I \in \{1, \ldots, n\}, \tag{6b}$$

where $\boldsymbol{S}_I$ is a set of the bag representatives' indices and $\boldsymbol{x}_{s_I}$ is the sample representative of bag $\boldsymbol{B}_I$. Note the variables in MIRSVMs formulation are the similar to those of the traditional SVM, except they are now representing each bag as an instance. Solving the optimization problem given in Equation (6) using a quadratic programming solver is a computationally expensive task due to the number of constraints, which scales by the number of bags n, as well as the calculation of the inner product between two d-dimensional vectors in constraint (6a). The proposed solution for these problems was deriving and solving the dual of the optimization problem given by Equation (6).

The dual can be found by first forming the primal Lagrangian given by Equation 7, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the non-negative Lagrange multipliers.

$$\mathcal{L}\left(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right) = \frac{1}{2}\sum_{j=1}^{d} w_j^2 + \frac{C}{n}\sum_{I}\xi_I - \sum_{I}\beta_I\xi_I - \sum_{I}\alpha_I\left(y_I\left(\sum_{j=1}^{d} w_j \boldsymbol{x}_{s_I j} + b\right) - 1 + \xi_I\right) \tag{7}$$

The following Karush-Kuhn-Tucker (KKT) [10] conditions must be satisfied:

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial w_j} = 0,\ \forall j \in \{1, \ldots, d\} \\
&\frac{\partial \mathcal{L}}{\partial b} = 0 \\
&\frac{\partial \mathcal{L}}{\partial \xi_I} = 0,\ \forall I \in \{1, \ldots, n\} \\
&\alpha_I\left(y_I(\langle \boldsymbol{w}, \boldsymbol{x}_{s_I}\rangle + b) - 1 + \xi_I\right) = 0,\ \forall I \in \{1, \ldots, n\} \\
&\beta_I\xi_I = 0, \forall I \in \{1, \ldots, n\} \\
&\alpha_I \geq 0,\ \beta_I \geq 0,\ \xi_I \geq 0,\ \forall I \in \{1, \ldots, n\}
\end{aligned}$$

At optimality, $\bigtriangledown_{w,b,\xi}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$ and the following conditions are met:

$$\frac{\partial \mathcal{L}}{\partial w_j} : w_j = \sum_{I} \alpha_I\, y_I\, \boldsymbol{x}_{s_I j},\ \forall j \in \{1, \ldots, d\} \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial b} : \sum_{I} \alpha_I y_I = 0, \tag{9}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_I} : \alpha_I + \beta_I = \frac{C}{n},\ \forall I \in \{1, \ldots, n\} \tag{10}$$

By substituting Equations 8, 9, and 10 into the Lagrangian in 7, the following dual problem, $q(\boldsymbol{\alpha}, \boldsymbol{\beta})$, is obtained:

$$\begin{aligned}
q(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \inf_{\boldsymbol{w}, b, \boldsymbol{\xi}} &\frac{1}{2}\sum_{I}\sum_{K \in I}\sum_{j=1}^{d} \alpha_I \alpha_K y_I y_K x_{s_I j} x_{s_K j} + \sum_{I}\xi_I\left(\alpha_I + \beta_I\right) - \sum_{I}\xi_I\left(\alpha_I + \beta_I\right) \\
&- \sum_{I}\sum_{K \in I}\sum_{j=1}^{d} \alpha_I \alpha_K y_I y_K x_{s_I j} x_{s_K j} - \sum_{I}\alpha_I y_I b + \sum_{I}\alpha_I.
\end{aligned}$$

At optimality, $\sum_I \alpha_I y_I = 0$, so the term with b can be removed. The terms with $\boldsymbol{\xi}$ can also

be removed because they negate each other. The resulting function is now with respect to the dual variables, so the infimum can be dropped. The dual MIRSVM formulation then becomes:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_I \alpha_I - \frac{1}{2} \sum_I \sum_{K \in I} \sum_{j=1}^{d} \alpha_I \alpha_K y_I y_K x_{s_I j} x_{s_K j} \tag{11}$$

$$\text{s.t.} \sum_I \alpha_I y_I = 0 \tag{11a}$$

$$\alpha_I + \beta_I = \frac{C}{n}, \forall I \in \{1, \ldots, n\} \tag{11b}$$

$$\alpha_I \geq 0, \forall I \in \{1, \ldots, n\} \tag{11c}$$

$$\beta_I \geq 0, \forall I \in \{1, \ldots, n\} \tag{11d}$$

where s_I is computed for each bag, as shown in Equation (12):

$$s_I = \underset{i \in I}{\text{argmax}} (\sum_{K \in I} \sum_{j=1}^{d} \alpha_K y_K x_{s_K j} x_{ij} + b), \forall I \in \{1, \ldots, n\}. \tag{12}$$

The implicit constraints (11b) through (11d) imply three possible cases for the α_I values:

1. If $\alpha_I = 0$, then $\beta_I = C/n$ and $\xi_I = 0$, implying the instance is correctly classified and outside the margin.

2. If $0 < \alpha_I < C/n$, then $\beta_I > 0$ and $\xi_I = 0$, indicating that the instance sits on the margin boundary, i.e. is an *unbounded support vector*.

3. If $\alpha_I = C/n$, then $\beta_I = 0$ and there is no restriction for $\xi_I \geq 0$. This also indicates that the instance is a support vector that is *bounded*. If $0 \leq \xi_I < 1$, then the instance is correctly classified, and is misclassified otherwise.

We then kernelize the dual function by replacing the inner product of the samples in feature space with their corresponding kernel values, $\mathcal{K}(\boldsymbol{x}_{s_I}, \boldsymbol{x}_{s_K})$. The dual function is now written as:

$$\max_{\boldsymbol{\alpha}} \sum_I \alpha_I - \frac{1}{2} \sum_I \sum_{K \in I} \alpha_I \alpha_K y_I y_K \mathcal{K}(\boldsymbol{x}_{s_I}, \boldsymbol{x}_{s_K}) \tag{13}$$

$$\text{s.t.} \sum_I \alpha_I y_I = 0 \tag{13a}$$

$$0 \leq \alpha_I \leq \frac{C}{n}, \forall I \in \{1, \ldots, n\}. \tag{13b}$$

One of the biggest advantages of the dual SVM formulation is the sparseness of the resulting model. This is because support vectors, instances that have their corresponding $\alpha_I \neq 0$, are only considered when forming the decision boundary. MIRSVM uses a Gaussian RBF kernel, given by Equation (14), where σ is the Gaussian shape parameter.

$$\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{14}$$

To evaluate the output vector, $\boldsymbol{o}_I$, of bag I using the kernel, the following equation is used [33],

$$\boldsymbol{o} = \mathcal{K}(\boldsymbol{B}_I, \boldsymbol{X}_S) * (\boldsymbol{\alpha} \cdot Y_S) + b \tag{15}$$

where $\boldsymbol{B}_I$ are the instances of bag I, $\boldsymbol{X}_S$ are the optimal bag representatives, and Y_S are the representative bag labels. The bias term b is calculated as shown in Equation 16, where $\boldsymbol{sv}$ is the vector of support vector indices and n_{sv} is the number of support vectors [33].

$$b = \frac{1}{n_{sv}} \sum_{\boldsymbol{sv}} Y_{\boldsymbol{sv}} - \mathcal{K}(\boldsymbol{X}_{\boldsymbol{sv}}, \boldsymbol{X}_{\boldsymbol{sv}}) * (\boldsymbol{\alpha}_{\boldsymbol{sv}} \cdot Y_{\boldsymbol{sv}}) \tag{16}$$

Algorithm 1 shows the procedure for training the multi-instance representative SVM classifier and obtaining the optimal representatives from each bag. During training, the representatives, $\boldsymbol{S}$, are first initialized by randomly selecting an instance from each bag. A hyper-plane is then obtained using the representative instances, and new optimal representatives are found with respect to the current hyper-plane, by using the rule given in Equation (12). At each step, the previous values in $\boldsymbol{S}$ are stored in $\boldsymbol{S}_{old}$. The training procedure ends when the bag representatives stop changing from one iteration to the next ($\boldsymbol{S} = \boldsymbol{S}_{old}$). Examples of the convergence of bag-representatives are shown in Figure 3. During the testing procedure, each bag produces an output vector based on the hyper-plane found in the training procedure. The bag label is then assigned by taking the sign of the output vector's maximum value, following the SMI assumption.

This formulation is designed to utilize and select representatives from positive and negative bags, unlike MISVM, which only optimizes over representatives from positive bags, while flattening the negative bag instances. MISVM allows multiple representatives to be chosen from negative bags and limits positive bag-representatives to be one, while MIRSVM allows for balanced bag-representative selection, where each bag is allowed one. MISVM also uses a wrapper method to initialize the positive bag-representatives by taking the mean vector of the instances within each positive bag. This is an implicit assumption that the instances within the positive bags are all positive, whereas MIRSVM's initialization procedure selects an instance from all bags at random, ensuring no noise is added by any wrapper techniques during initialization and no assumptions are made about the instances. Due to the constraints on the representatives, MIRSVM produces sparser models while MISVM has the freedom to select as many negative support vectors as it needs and restricts the support vectors chosen from positive bags to be one. Figure 4 shows the decision boundaries produced by MIRSVM and MISVM to highlight the differences in their solutions. As Figure 4 shows, MISVM produces a larger number of support vectors from the negative bags, which greatly influences the final decision boundary in favor of the negative class.

4. Experiments

This section presents the experimental setup and comparison of our contribution, as well as ten other state-of-the-art methods on 15 different benchmark datasets. First, the experimental setup is described and the state-of-the-art methods are listed. The results for each metric, as well as the statistical analysis, are then presented and analyzed. Finally, the run-time and meta-ranks of each algorithm are shown, and the overall performance of the algorithms across each metric is

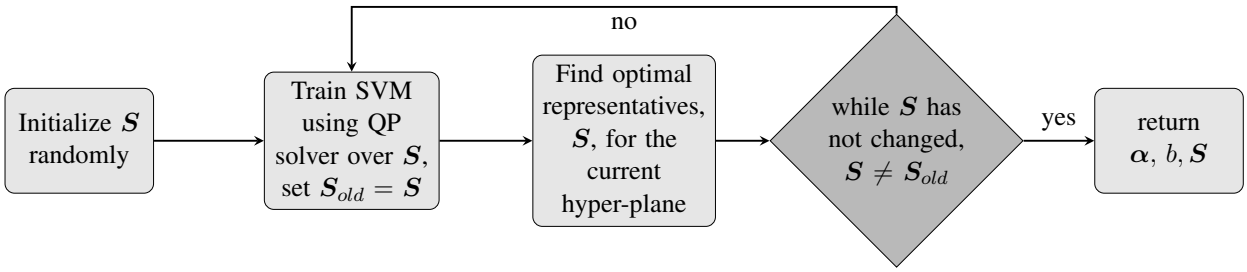

Figure 2: MIRSVM Flow Diagram. This figure represents a summary of the steps performed by the MIRSVM algorithm. The representatives are first randomly initialized and continuously updated according to the current hyper-plane, which is found using a quadratic programming (QP) solver. Upon completion, the model is returned along with the optimal bag-representatives.

Algorithm 1 Multi-Instance Representative SVM (MIRSVM)

Input: Training dataset $\mathcal{D}$, SVM Parameters C and σ

Output: SVM model parameters $\boldsymbol{\alpha}$ and b, Bag Representative IDs $\boldsymbol{S}$

$\boldsymbol{S}_{old} \leftarrow -\infty$
for $I \in \{1, \ldots, n\}$ **do**
    $S_I \leftarrow$ rand $(|\boldsymbol{B}_I|, 1, 1)$ ▷ Assign each bag a random instance
end for
$\boldsymbol{X}_S \leftarrow \boldsymbol{X}(\boldsymbol{S})$, $Y_S \leftarrow Y(\boldsymbol{S})$ ▷ Initialize the representative dataset
while $\boldsymbol{S} \neq \boldsymbol{S}_{old}$ **do**
    $\boldsymbol{S}_{old} \leftarrow \boldsymbol{S}$
    $\boldsymbol{G} \leftarrow (Y_S \times Y_S) \cdot \mathcal{K}(\boldsymbol{X}_S, \boldsymbol{X}_S, \sigma)$ ▷ Build Graham matrix
    $\boldsymbol{\alpha} \leftarrow$ quadprog $(\boldsymbol{G}, -\mathbf{1}^n, Y_S, \mathbf{0}^n, \mathbf{0}^n, \boldsymbol{C}^n)$ ▷ Solve QP Problem
    $\boldsymbol{sv} \leftarrow$ find $(0 < \boldsymbol{\alpha} \leq C)$ ▷ Get the support vector indices
    $n_{sv} \leftarrow$ count $(0 < \boldsymbol{\alpha} \leq C)$ ▷ Get the number of support vectors
    $b \leftarrow \frac{1}{n_{sv}} \sum_{i=1}^{n_{sv}} (Y_{\boldsymbol{sv}} - \boldsymbol{G}_{\boldsymbol{sv}} * (\boldsymbol{\alpha}_{\boldsymbol{sv}} \cdot Y_{\boldsymbol{sv}}))$ ▷ Calculate the bias term
    for $I \in \{1, \ldots, n\}$ **do**
        $\boldsymbol{G}_I \leftarrow (Y_I \times Y_S) \cdot \mathcal{K}(\boldsymbol{B}_I, \boldsymbol{X}_S, \sigma)$
        $\boldsymbol{S}_I \leftarrow \text{argmax}_{i \in I} (\boldsymbol{G}_I * \boldsymbol{\alpha} + b)$ ▷ Select optimal bag-representatives using SMI assumption
    end for
    $\boldsymbol{X}_S \leftarrow \boldsymbol{X}(\boldsymbol{S})$, $Y_S \leftarrow Y(\boldsymbol{S})$ ▷ Re-set the representative dataset
end while

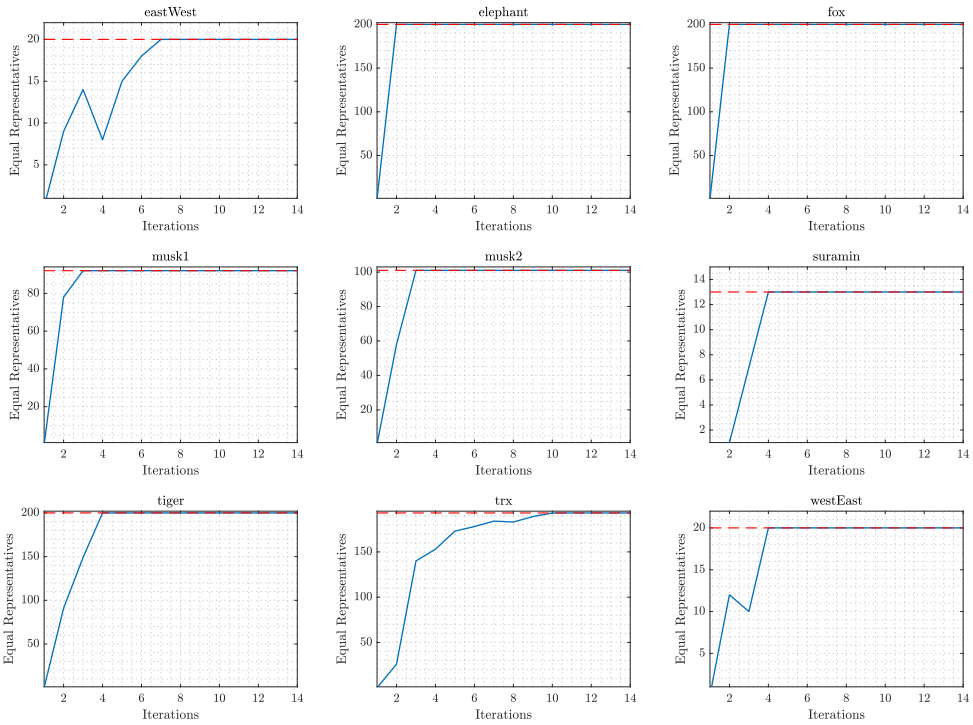

Figure 3: Bag representative convergence plots on 9 datasets. The blue line shows the number of bag representatives that are equal from one iteration to the next. The red dashed line represents the total number of bags.

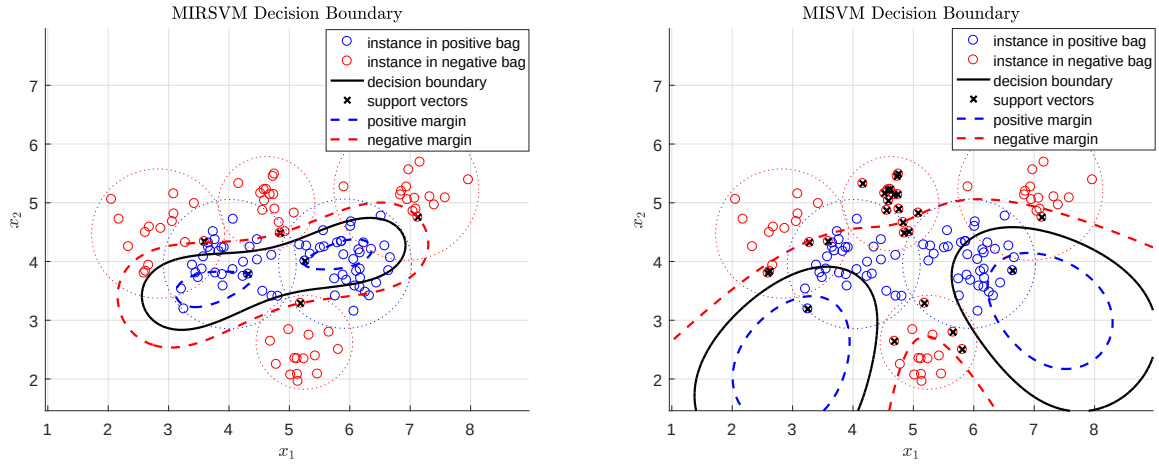

Figure 4: Difference between MIRSVM and MISVM on a random 2-dimensional toy dataset. The instances of this dataset were drawn from the standard normal distribution, with 6 bags, 4 of which are negative (red) and 2 of which are positive (blue), indicated by the red and blue dotted circles. There are a total of 125 instances, 65 belonging to the positive bags and 60 belonging to the negative bags. The decision boundaries (black lines) are shown for each of the algorithms, along with the SVM margin (red and blue dashed lines) and support vectors (black crosses). In this example, both MIRSVM and MISVM were trained using $C = 1000$ and $\sigma = 1.5$. Note the differing number of support vectors produced by the two methods. MIRSVM has 6, one for each bag, and MISVM has 29. Also note the smoother representation of the data distribution given by MIRSVM's decision boundary, unlike MISVM whose decision boundary was greatly affected by the larger number of support vectors belonging to the negative class with respect to the only 2 positive support vectors.

discussed. The main aim of the experiments is to compare our contribution to other multi-instance support vector machines, state-of-the-art multi-instance learners, and ensemble methods.

4.1. Experimental Setup

Table 2 presents a summary of the 15 datasets used throughout the experiments, where the number of attributes, bags, and total number of instances are shown. The datasets were obtained from the Weka[1] [29] and KEEL[2] [1] dataset repositories.

The experimental environment was designed to test the difference in performance of the proposed method against 10 state-of-the-art algorithms, contrasting instance-level methods and bag-level methods. Instance-level methods include MIOptimalBall, MIBoost, MISVM, MIDD, and MIWrapper. The bag-level methods include MISMO, SimpleMI, and TLC. The ensemble-based bag-space methods, Bagging and Stacking, were also used. The base algorithms selected for the ensembles Bagging and Stacking were TLC, and TLC and SimpleMI, respectively. These algorithms were chosen because they have shown considerable performance in learning multi-instance models, while also having their frameworks readily available for reproducing their results through MILK, the Multi-Instance Learning Kit[3] [52], used in conjunction with the Weka framework. Experiments were run in an Intel i7-6700k CPU with 32GB RAM. MIRSVM was implemented in MATLAB while the referenced algorithms are available in the Java implementation of Weka.

[1] http://www.cs.waikato.ac.nz/ml/weka
[2] http://sci2s.ugr.es/keel/datasets.php
[3] http://www.cs.waikato.ac.nz/ml/milk

Table 2: Datasets used for the experiments in 4.2

Dataset	Attributes	Positive Bags	Negative Bags	Total	Instances
Suramin	20	7	4	11	2378
EastWest	24	10	10	20	213
WestEast	24	10	10	20	213
Musk1	166	47	45	92	476
Musk2	166	39	63	102	6598
Webmining	5863	17	58	75	2212
TRX	8	25	193	168	26611
Mutagenesis-Atoms	10	125	63	188	1618
Mutagenesis-Bonds	16	125	63	188	3995
Mutagenesis-Chains	24	125	63	188	5349
Tiger	230	100	100	200	1391
Elephant	230	100	100	200	1220
Fox	230	100	100	200	1320
Component	22	423	2707	3130	36894
Function	202	443	4799	5242	55536

Experiments were performed using k-fold cross validation, with $k = 10$, in order to evaluate the models' performances and tune hyper-parameters. The data is separated fairly into 10 equally sized sections where, at every iteration of the cross-validation loop, a section is held out as the test set, while the remainder of the data is used for training. This procedure ensures the model is not optimistically biased towards the full dataset. The tuning of the model during cross-validation includes finding the best penalty parameter, C, as well as the best shape parameter for the Gaussian radial basis function kernel, σ. The best hyper-parameters were chosen from the following 6×6 possible combination runs, shown in Equations (17a) and (17b), referred to as (17).

$$C \in \{0.1, 1, 10, 100, 1000, 10000\} \tag{17a}$$

$$\sigma \in \{0.1, 0.5, 1, 2, 5, 10\} \tag{17b}$$

These parameters were also used for the state-of-the-art SVM methods. This was done in order to keep the experimental environment controlled and ensure fair evaluation of the multi-instance SVM algorithms. The parameters for the referenced algorithms used throughout the experiments were those specified by their authors.

4.2. Results & Statistical Analysis

The classification performance was measured using five metrics: Accuracy (18a), Precision (18b), Recall (18c), Cohen's kappa rate (18d), and Area under ROC curve (AUC) (18e). Accuracy can be misleading when classes are imbalanced, as is the case with the *component* and *function* datasets, which have six and ten times as many negative instances than positive, respectively. Cohen's Kappa Rate and the AUC measures are used as complementary measures in order to evaluate the algorithms comprehensively [6]. Cohen's kappa rate, shown in Equation (18d), evaluates classifier merit according to the class distribution and ranges between -1 (full disagreement), 0 (random classification), and 1 (full agreement). The AUC metric highlights the trade-off between the true

positive rate, or recall, and the false positive rate, as shown in Equation (18e). The values of the true positive (TP), true negative (TN), false positive (FP), and false negative samples (FN) were first collected for each of the classifiers, then the metrics were computed using the equations shown in (18) on the n' bags of the test data. The run times of each algorithm are also reported to analyze the usability and speed of each of the algorithms across differently sized datasets. The results for these metrics are shown in Tables 3, 5, 7, 9, 11, and 13.

$$\text{Accuracy} \qquad \frac{TP + TN}{n'} \tag{18a}$$

$$\text{Precision} \qquad \frac{TP}{TP + FP} \tag{18b}$$

$$\text{Recall} \qquad \frac{TP}{TP + FN} \tag{18c}$$

$$\text{Cohen's Kappa Rate} \qquad \frac{n' - \frac{(TP + FN) * (TP + FP)}{n'}}{1 - \frac{(TP + FN) * (TP + FP)}{n'}} \tag{18d}$$

$$\text{Area Under ROC Curve} \qquad \frac{1 + \frac{TP}{TP + FN} - \frac{FP}{FP + TN}}{2} \tag{18e}$$

In order to analyze the performances of the multiple models, non-parametric statistical tests are used to validate the experimental results obtained [17, 26]. The Iman-Davenport non-parametric test is run to investigate whether significant differences exist among the performance of the algorithms [24] by ranking them over the datasets used, using the Friedman test. The algorithm ranks for each metric in Equations (18) are presented in the last row of the results tables, and the lowest (best) rank value is typeset in bold. Table 14 contains the ranks and meta-rank of all methods, which helps determine and visualize the best performing algorithms across all datasets and metrics.

After the Iman-Davenport test indicates significant differences, the Bonferroni-Dunn post-hoc test [20] is then used to find where they occur between algorithms by assuming the classifiers' performances are different by at least some critical value [25]. Below each result table, a figure highlighting the critical distance (in gray), from the best ranking algorithm to the rest, is shown. The algorithms to the right of the critical distance bar perform statistically significantly worse than the control algorithm, MIRSVM. Figures 5, 6, 7, 8, 9, 10 show the results of the Bonferroni-Dunn post-hoc procedure over the metrics in (18), as well as the meta-rank results in Table 14.

The Nemenyi, Holm, and Shaffer post-hoc [28, 32] tests were then run for each of the metrics to compute multiple pairwise comparisons between the proposed algorithm and the state-of-the-art methods, investigating whether statistical differences exist among pairs of algorithms. Tables 4, 6, 8, 10, and 12 show the p-values for the Nemenyi, Holm, and Shaffer tests for $\alpha = 0.05$.

4.3. Accuracy

Table 3 shows the accuracy results of the 11 algorithms over 15 multi-instance datasets, along with their average and rank. The results indicate that the bag-based and ensemble learners perform better than the instance-based and wrapper methods. Specifically, MIRSVM achieves the best accuracy over 6 of the 15 datasets with a competitive average against the Bagging, Stacking, and

TLC algorithms. Note that MIRSVM performs better than MISVM for all datasets, indicating that using representatives from each bag and limiting the number of support-vectors per negative bag improves the classification performance. The instance-level classifiers and wrapper methods, such as MIBoost, MIWrapper, and SimpleMI perform the worst. This behavior emphasizes the importance of not making prior assumptions about the positive bags' distributions.

Figure 5 and Table 4 show the results for the statistical analysis on the accuracy results. The algorithms with ranking higher than 5.51 (MIRSVM rank + Bonferroni-Dunn critical value), to the right of the grey bar in Figure 5, perform statistically worse than MIRSVM. Table 4 shows the p-values of the Nemenyi, Holm, and Shaffer tests. The results from these tests are complement each other. Nemenyi's procedure indicates that MIRSVM performs significantly better than all methods (with p-values < 0.01), except Bagging, Stacking, TLC, and MISMO. The Nemenyi p-values for MIBoost, MIWrapper, and SimpleMI are $= 0$, showing that MIRSVM has significantly better accuracy. However, the stricter Holm and Shaffer tests indicate that significant differences exist between MIRSVM and all other classifiers, highlighting MIRSVM's superior classification accuracy.

Table 3: Accuracy

Datasets	MIRSVM	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	0.6000	0.5000	0.7250	0.4250	0.5000	0.7250	0.5000	0.2308	0.6923	0.6750	**0.7564**
eastWest	**0.8000**	0.5000	0.7250	0.6125	0.5000	0.7125	0.6375	0.5000	0.5000	0.6375	0.4500
westEast	0.6500	0.5000	0.3750	0.4500	0.5000	**0.7375**	0.4625	0.5000	0.5000	0.6875	0.6375
musk1	**0.9022**	0.5109	0.7717	0.8804	0.5109	0.7826	0.8043	0.5109	0.8587	0.8804	0.8587
musk2	**0.8146**	0.6139	0.7723	0.7228	0.6139	0.7030	0.7129	0.6139	0.6238	0.7129	0.6733
webmining	**0.8500**	0.8142	0.7699	0.8142	0.8142	0.8407	0.6903	0.8142	0.8142	0.7876	0.8053
trx	0.8825	0.8705	**0.9016**	0.8808	0.8705	0.8705	0.8705	0.8705	0.8756	0.8964	0.8860
mutagenesis-atoms	0.7714	0.6649	0.6436	0.7074	0.6649	0.6915	0.6649	0.6649	0.7766	**0.8032**	0.7606
mutagenesis-bonds	0.8252	0.6649	0.6915	0.7713	0.6649	0.7979	0.6649	0.6649	0.8351	**0.8830**	0.8564
mutagenesis-chains	0.8411	0.6649	0.6702	0.7766	0.6649	0.8351	0.6649	0.6649	0.8404	**0.8457**	0.8351
tiger	**0.7750**	0.5000	0.5000	0.7100	0.5000	0.7200	0.7550	0.5000	0.6650	0.7700	0.7250
elephant	0.8300	0.5000	0.5000	0.7900	0.5000	0.8100	0.8000	0.5000	0.8000	**0.8500**	0.8250
fox	**0.6550**	0.5000	0.5000	0.5800	0.5000	0.5250	0.5900	0.5000	0.6450	0.6200	0.6500
component	0.9366	0.8649	0.8696	0.8780	0.8649	0.8968	0.8703	0.8649	0.9358	**0.9371**	0.9355
function	0.9523	0.9155	0.9138	0.9193	0.9155	0.9376	0.9195	0.9155	0.9649	**0.9655**	0.9647
Average	**0.8057**	0.6390	0.6886	0.7279	0.6390	0.7724	0.7072	0.6210	0.7552	0.7968	0.7746
Ranks	**2.4000**	8.8000	7.2667	6.0333	8.8000	4.8000	7.0667	9.0000	4.7333	2.7667	4.3333

Figure 5: Bonferroni-Dunn test for Accuracy

Table 4: Nemenyi, Holm, and Shaffer tests for Accuracy

MIRSVM vs.	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
Nemenyi p-value	0.00000	0.00006	0.00270	0.00000	0.04751	0.00012	0.00000	0.05402	0.76207	0.11040
Holm p-value	0.00093	0.00102	0.00139	0.00094	0.00185	0.00104	0.00091	0.00192	0.00833	0.00238
Shaffer p-value	0.00111	0.00111	0.00139	0.00111	0.00185	0.00111	0.00091	0.00192	0.00833	0.00238

Table 5: Precision

Datasets	MIRSVM	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	0.6385	**1.0000**	**1.0000**	0.2857	**1.0000**	**1.0000**	0.5000	0.0000	0.6667	0.7333	0.7116
eastWest	0.7143	0.5000	**0.8750**	0.5882	0.5000	0.7429	0.8667	0.5000	0.5000	0.6667	0.4444
westEast	0.6300	0.5000	0.2727	0.4600	0.5000	**0.6939**	0.3846	0.5000	0.5000	0.6364	0.6038
musk1	0.8519	**1.0000**	0.9286	0.9048	**1.0000**	0.8049	0.8857	**1.0000**	0.8478	0.9250	0.8478
musk2	0.7167	0.6139	**0.7826**	0.7576	0.6139	0.7424	0.7538	0.6139	0.7400	0.7797	0.7164
webmining	0.7500	0.8142	0.8173	0.8142	0.8142	0.8936	**1.0000**	0.8142	0.8817	0.8469	0.8500
trx	0.6500	0.8705	**0.9306**	0.9191	0.8705	0.8705	0.8705	0.8705	0.9138	0.8936	0.9011
mutagenesis-atoms	0.7872	**1.0000**	0.4630	0.6111	**1.0000**	0.5439	**1.0000**	**1.0000**	0.7059	0.7321	0.6667
mutagenesis-bonds	0.8468	**1.0000**	0.5385	0.7500	**1.0000**	0.6812	**1.0000**	**1.0000**	0.7857	0.8596	0.8333
mutagenesis-chains	0.8571	**1.0000**	0.5091	0.7059	**1.0000**	0.7759	**1.0000**	**1.0000**	0.7705	0.7742	0.7581
tiger	0.7365	0.5000	0.5000	0.6944	0.5000	0.7444	0.7802	0.5000	0.6514	**0.7935**	0.7320
elephant	0.8576	0.5000	0.5000	0.7959	0.5000	0.8444	0.7679	0.5000	0.8000	**0.8804**	0.8283
fox	0.6040	0.5000	0.5000	0.5833	0.5000	0.5287	0.6216	0.5000	**0.6747**	0.6304	0.6705
component	**0.9866**	0.8649	0.8778	0.8902	0.8649	0.8958	0.8696	0.8649	0.9462	0.9431	0.9449
function	0.8459	0.9155	0.9202	0.9317	0.9155	0.9376	0.9197	0.9155	**0.9729**	0.9720	0.9726
Average	0.7649	0.7719	0.6944	0.7128	0.7719	0.7800	**0.8147**	0.7053	0.7572	0.8045	0.7654
Ranks	6.0667	6.7333	6.7333	6.7667	6.7333	5.3667	5.1000	7.3000	5.5000	**3.9333**	5.7667

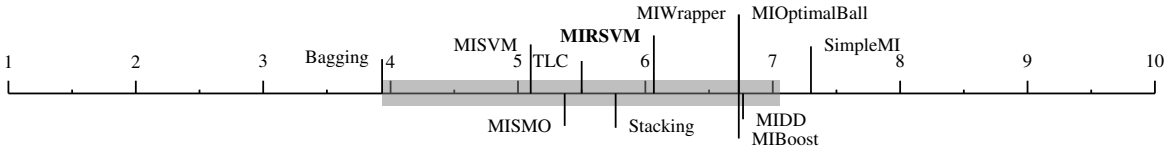

Figure 6: Bonferroni-Dunn test for Precision

Table 6: Nemenyi, Holm, and Shaffer tests for Precision

MIRSVM vs.	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
Nemenyi p-value	0.58199	0.58199	0.56326	0.58199	0.56326	0.42475	0.30849	0.63985	0.07815	0.80435
Holm p-value	0.00263	0.00250	0.00217	0.00278	0.00227	0.00185	0.00156	0.00294	0.00102	0.00500
Shaffer p-value	0.00263	0.00250	0.00217	0.00278	0.00227	0.00185	0.00156	0.00294	0.00102	0.00500

4.4. Precision & Recall

Precision and recall are metrics that must be evaluated together in order to observe their behavior simultaneously, since they are both metrics used to measure relevance. Tables 5 and 7 show the precision and recall results obtained by each algorithm. The precision and recall results for MIWrapper and SimpleMI indicate that they are unstable classifiers, exhibiting extreme variance in behavior, making them unsuitable for real-world applications. It is also interesting to analyze the performance on the mutagenesis datasets, where MISVM, MIBoost, MIWrapper, and SimpleMI predict all bags as negative. Additionally, while MISMO obtains unbiased results on these datasets, MIRSVM significantly outperforms it over both precision and recall.

Figure 6 shows that there are no statistically significant differences between the precision results obtained by all algorithms, except SimpleMI; and Figure 7 shows none over all. It is worth noting that MIRSVM outperforms both ensemble methods according to recall, despite them exhibiting relatively good accuracy and precision. This indicates that they are strongly conservative towards predicting positive bags. The Holm and Shaffer tests indicate that significant differences exist between MIRSVM and the state-of-the-art, with the exception of SimpleMI.

Table 7: Recall

Datasets	MIRSVM	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	**1.0000**	0.0000	0.4500	0.1000	0.0000	0.4500	0.5000	0.0000	0.6667	0.5500	0.9125
eastWest	**1.0000**	0.7000	0.5250	0.7500	0.7000	0.6500	0.3250	**1.0000**	0.5000	0.5500	0.4000
westEast	0.9000	0.9000	0.1500	0.5750	0.9000	0.8500	0.1250	**1.0000**	0.5000	0.8750	0.8000
musk1	**0.9787**	0.0000	0.5778	0.8444	0.0000	0.7333	0.6889	0.0000	0.8667	0.8222	0.8667
musk2	0.9250	**1.0000**	0.8710	0.8065	**1.0000**	0.7903	0.7903	**1.0000**	0.5968	0.7419	0.7742
webmining	0.2857	**1.0000**	0.9239	**1.0000**	**1.0000**	0.9130	0.6196	**1.0000**	0.8913	0.9022	0.9239
trx	0.4833	**1.0000**	0.9583	0.9464	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9464	**1.0000**	0.9762
mutagenesis-atoms	**0.8880**	0.0000	0.3968	0.3492	0.0000	0.4921	0.0000	0.0000	0.5714	0.6508	0.5714
mutagenesis-bonds	**0.8960**	0.0000	0.5556	0.4762	0.0000	0.7460	0.0000	0.0000	0.6984	0.7778	0.7143
mutagenesis-chains	**0.9120**	0.0000	0.4444	0.5714	0.0000	0.7143	0.0000	0.0000	0.7460	0.7619	0.7460
tiger	0.8700	0.5000	**1.0000**	0.7500	0.5000	0.6700	0.7100	**1.0000**	0.7100	0.7300	0.7100
elephant	0.9100	0.6000	**1.0000**	0.7800	0.6000	0.7600	0.8600	**1.0000**	0.8000	0.8100	0.8200
fox	0.9000	0.7000	**1.0000**	0.5600	0.7000	0.4600	0.4600	**1.0000**	0.5600	0.5800	0.5900
component	0.5839	**1.0000**	0.9867	0.9797	**1.0000**	0.9967	**1.0000**	**1.0000**	0.9815	0.9867	0.9826
function	0.5327	**1.0000**	0.9919	0.9840	**1.0000**	0.9983	0.9994	**1.0000**	0.9892	0.9908	0.9894
Average	**0.8044**	0.5600	0.7221	0.6982	0.5600	0.7483	0.5385	0.6667	0.7350	0.7820	0.7851
Ranks	**4.4333**	6.2667	5.8000	6.7667	6.2667	6.2667	7.4333	4.5333	7.0333	5.3333	5.8667

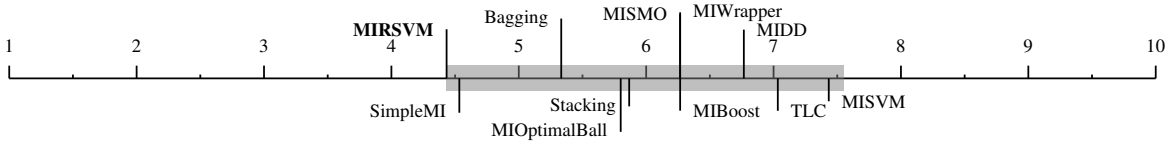

Figure 7: Bonferroni-Dunn test for Recall

Table 8: Nemenyi, Holm, and Shaffer tests for Recall

MIRSVM vs.	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
Nemenyi p-value	0.13007	0.25911	0.05402	0.13007	0.13007	0.01324	0.93419	0.03180	0.45739	0.23660
Holm p-value	0.00104	0.00135	0.00098	0.00106	0.00109	0.00090	0.01000	0.00094	0.00208	0.00132
Shaffer p-value	0.00104	0.00135	0.00098	0.00106	0.00109	0.00090	0.01000	0.00094	0.00208	0.00132

4.5. Cohen's Kappa Rate

Table 9 shows the Cohen's Kappa rate results obtained by the algorithms. These results support the accuracy achieved by the algorithms, in the sense that the instance-based and wrapper methods perform worse than bag-based and ensemble learners. Specifically, MIRSVM achieves the best kappa rate over 7 of the 15 datasets with a competitive average against the Bagging, Stacking, SMO, and TLC algorithms. MIRSVM's kappa values all fall within the range (0.5-1], indicating that its merit as a classifier agrees with the class distribution and is not random. Note that SimpleMI, MIOptimalBall, MIDD, MISVM, and Stacking contain some negative kappa values, indicating performance worse than the default-hypothesis. MIBoost and MIWrapper are shown to randomly classify all 15 datasets.

Figure 8 and Table 10 show the results of the statistical analysis on the Cohen's Kappa Rate results. Nemenyi's procedure reflects results similar to the p-values obtained from the accuracy results, where MIRSVM performs significantly better than MIOptimalBall, MIDD, MISVM, MIWrapper, MIBoost, and SimpleMI, having p-values < 0.01. The Holm and Shaffer procedures indicate that significant differences exist among all algorithms results, supporting MIRSVM's performance as a competitive classifier.

Table 9: Cohen's Kappa Rate

Datasets	MIRSVM	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	**0.6320**	0.0000	0.4500	-0.1500	0.0000	0.4500	0.0000	-0.5854	0.3810	0.3500	0.5121
eastWest	**0.6000**	0.0000	0.4500	0.2250	0.0000	0.4250	0.2750	0.0000	0.0000	0.2750	-0.1000
westEast	0.3500	0.0000	-0.2500	-0.1000	0.0000	**0.4750**	-0.0750	0.0000	0.0000	0.3750	0.2750
musk1	**0.8036**	0.0000	0.5396	0.7604	0.0000	0.5642	0.6067	0.0000	0.7174	0.7602	0.7174
musk2	**0.6263**	0.0000	0.5031	0.4039	0.0000	0.3613	0.3856	0.0000	0.2492	0.4029	0.2940
webmining	0.3468	0.0000	0.0246	0.0000	0.0000	**0.4535**	0.3771	0.0000	0.3744	0.2112	0.2458
trx	0.4542	0.0000	**0.5228**	0.4224	0.0000	0.0000	0.0000	0.0000	0.3858	0.3032	0.3364
mutagenesis-atoms	0.5395	0.0000	0.1709	0.2654	0.0000	0.2909	0.0000	0.0000	0.4738	**0.5458**	0.4431
mutagenesis-bonds	0.5699	0.0000	0.3131	0.4356	0.0000	0.5569	0.0000	0.0000	0.6195	**0.7310**	0.6659
mutagenesis-chains	0.6303	0.0000	0.2359	0.4738	0.0000	0.6225	0.0000	0.0000	0.6391	**0.6525**	0.6285
tiger	**0.5500**	0.0000	0.0000	0.4200	0.0000	0.4400	0.5100	0.0000	0.3300	0.5400	0.4500
elephant	**0.7000**	0.0000	0.0000	0.5800	0.0000	0.6200	0.6000	0.0000	0.6000	**0.7000**	0.6500
fox	**0.3100**	0.0000	0.0000	0.1600	0.0000	0.0500	0.1800	0.0000	0.2900	0.2400	0.3000
component	0.6644	0.0000	0.1613	0.2836	0.0000	0.3656	0.0675	0.0000	**0.6945**	0.6924	0.6906
function	0.6292	0.0000	0.0966	0.2801	0.0000	0.4083	0.0933	0.0000	0.7529	**0.7534**	0.7507
Average	**0.5604**	0.0000	0.2145	0.2973	0.0000	0.4055	0.2013	-0.0390	0.4338	0.5022	0.4573
Ranks	**2.2333**	9.2333	6.5333	6.1667	9.2333	4.7667	6.7667	9.4333	4.4000	3.0000	4.2333

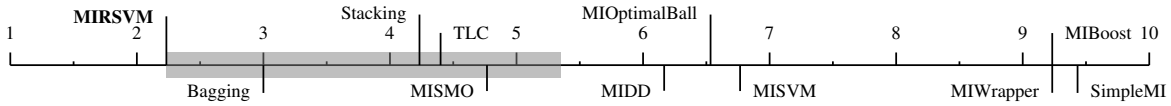

Figure 8: Bonferroni-Dunn test for Cohen's Kappa rate

Table 10: Nemenyi, Holm, and Shaffer tests for Cohen's Kappa rate

MIRSVM vs.	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
Nemenyi p-value	0.00000	0.00105	0.00353	0.00000	0.03180	0.00018	0.00000	0.03899	0.49139	0.13720
Holm p-value	0.00091	0.00128	0.00143	0.00093	0.00200	0.00119	0.00094	0.00208	0.00556	0.00263
Shaffer p-value	0.00091	0.00135	0.00143	0.00111	0.00200	0.00135	0.00111	0.00208	0.00556	0.00263

Table 11: AUC

Datasets	MIRSVM	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	0.5000	0.5000	**0.7250**	0.4250	0.5000	**0.7250**	0.5000	0.2143	0.6905	0.6750	0.6811
eastWest	**0.8000**	0.5000	0.7250	0.6125	0.5000	0.7125	0.6375	0.5000	0.5000	0.6375	0.4500
westEast	0.6500	0.5000	0.3750	0.4500	0.5000	**0.7375**	0.4625	0.5000	0.5000	0.6875	0.6375
musk1	**0.9005**	0.5000	0.7676	0.8797	0.5000	0.7816	0.8019	0.5000	0.8589	0.8792	0.8589
musk2	**0.8351**	0.5000	0.7432	0.6981	0.5000	0.6772	0.6900	0.5000	0.6317	0.7043	0.6435
webmining	0.6320	0.5000	0.5096	0.5000	0.5000	0.7184	**0.8098**	0.5000	0.6837	0.5939	0.6048
trx	0.7243	0.5000	**0.7392**	0.6932	0.5000	0.5000	0.5000	0.5000	0.6732	0.6000	0.6281
mutagenesis-atoms	0.7106	0.5000	0.5824	0.6186	0.5000	0.6420	0.5000	0.5000	0.7257	**0.7654**	0.7137
mutagenesis-bonds	0.7856	0.5000	0.6578	0.6981	0.5000	0.7850	0.5000	0.5000	0.8012	**0.8569**	0.8211
mutagenesis-chains	**0.8252**	0.5000	0.6142	0.7257	0.5000	0.8051	0.5000	0.5000	0.8170	0.8250	0.8130
tiger	**0.7750**	0.5000	0.5000	0.7100	0.5000	0.7200	0.7550	0.5000	0.6650	0.7700	0.7250
elephant	0.8200	0.5000	0.5000	0.7900	0.5000	0.8100	0.8000	0.5000	0.8000	**0.8500**	0.8250
fox	**0.6550**	0.5000	0.5000	0.5800	0.5000	0.5250	0.5900	0.5000	0.6450	0.6200	0.6500
component	0.7855	0.5000	0.5536	0.6033	0.5000	0.6272	0.5201	0.5000	**0.8123**	0.8030	0.8081
function	0.7563	0.5000	0.5298	0.6015	0.5000	0.6391	0.5268	0.5000	**0.8456**	0.8408	0.8434
Average	**0.7437**	0.5000	0.6015	0.6390	0.5000	0.6937	0.6062	0.4810	0.7100	0.7406	0.7135
Ranks	**2.7667**	9.2000	6.4000	6.2333	9.2000	4.7000	6.6667	9.4333	4.2000	3.1000	4.1000

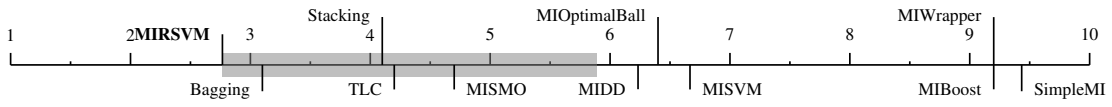

Figure 9: Bonferroni-Dunn test for AUC

Table 12: Nemenyi, Holm, and Shaffer tests for AUC

MIRSVM vs.	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
Nemenyi p-value	0.00000	0.00270	0.00420	0.00000	0.11040	0.00128	0.00000	0.23660	0.78313	0.27091
Holm p-value	0.00093	0.00128	0.00135	0.00094	0.00278	0.00125	0.00091	0.00357	0.00714	0.00385
Shaffer p-value	0.00111	0.00135	0.00135	0.00111	0.00278	0.00135	0.00091	0.00357	0.00714	0.00385

4.6. AUC

Table 11 shows AUC results obtained by the algorithms. These results complement the accuracy and kappa rate, emphasizing the better performance of bag-based methods. MIRSVM achieves the best AUC score on 6 of the 15 datasets, while MIBoost, SimpleMI, and MIWrapper obtain the worst results. Their AUC score indicates random predictor behavior, having values ≤ 0.5. Bag-level methods all obtain scores between 0.7 and 0.75 indicating a high true positive rate and a low false positive rate, which is reflected by the precision and recall results.

Figure 9 and Table 12 show that MIRSVM performs significantly better than 6 out of the 10 competing algorithms. Nemenyi's procedure indicates that significant differences exist between MIRSVM and all algorithms except MISMO, TLC, Bagging, and Stacking. MISVM is ranked 8^{th} with respect to the other state-of-the-art methods. MISVM's true positive rate could be affected because of the possible imbalance of support vectors from the positive and negative classes (favoring the negative). Note that the Nemenyi p-values for MIWrapper, MIBoost, and SimpleMI are $= 0$.

Table 13: Run Time (seconds)

Datasets	MIRSVM	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	**0.1**	8.8	30.5	7922.0	9.5	52.3	333.9	7.3	35.5	80.5	1085.0
eastWest	**0.1**	5.5	9.4	217.1	6.3	14.8	21.4	5.8	15.4	14.2	15.3
westEast	**0.1**	6.5	7.8	79.7	6.5	14.7	99.5	6.0	16.6	11.1	10.8
musk1	**0.4**	13.4	32.1	3542.6	20.6	89.7	198.4	11.1	93.0	474.3	759.5
musk2	**2.3**	97.3	782.9	126016.8	208.3	1799.4	26093.5	16.1	1772.2	14817.3	16759.0
webmining	300.6	45745.4	60474.8	47601.4	68736.7	51923.6	105622.3	2685.9	86272.6	667636.3	**10.8**
trx	61.8	17.6	682.3	339110.5	19.3	8670.3	134622.1	**7.4**	2229.3	17887.3	592948.9
mutagenesis-atoms	9.8	8.8	99.2	2623.0	8.0	55.0	53.5	**6.4**	44.0	182.8	153.9
mutagenesis-bonds	**8.3**	10.2	310.2	17538.7	12.3	457.4	2794.8	8.4	131.1	755.5	853.1
mutagenesis-chains	19.3	12.0	525.0	48982.7	14.9	2451.9	6637.4	**7.2**	224.4	1449.6	1619.0
tiger	29.5	44.5	157.8	23220.5	56.2	208.0	608.8	**16.3**	183.0	1276.7	11927.9
elephant	47.7	45.5	243.9	56456.3	69.7	232.1	1114.3	**20.8**	212.1	1030.7	1462.2
fox	81.0	44.3	206.1	27773.8	66.0	369.6	891.5	**23.5**	243.3	1332.5	1729.1
component	231.7	572.5	228209.6	96263.9	1096.9	629366.4	37224.6	**144.0**	9861.5	74860.9	79149.8
function	740.3	935.5	768458.0	350124.7	1887.5	1052225.3	565026.4	**232.8**	12128.2	138742.0	185918.5
Average	**102.2**	3171.2	70682.0	76498.2	4814.6	116528.7	58756.2	213.3	7564.1	61370.1	59626.8
Rank	2.2	2.8	6.3	10.1	3.9	7.5	8.9	**1.6**	6.3	8.1	8.4

Table 14: Overall ranks comparison

Ranks	MIRSVM	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
accuracy	**2.4000**	8.8000	7.2667	6.0333	8.8000	4.8000	7.0667	9.0000	4.7333	2.7667	4.3333
precision	6.0667	6.7333	6.7333	6.7667	6.7333	5.3667	5.1000	7.3000	5.5000	**3.9333**	5.7667
recall	**4.4333**	6.2667	5.8000	6.7667	6.2667	6.2667	7.4333	4.5333	7.0333	5.3333	5.8667
kappa	**2.2333**	9.2333	6.5333	6.1667	9.2333	4.7667	6.7667	9.4333	4.4000	3.0000	4.2333
auc	**2.7667**	9.2000	6.4000	6.2333	9.2000	4.7000	6.6667	9.4333	4.2000	3.1000	4.1000
run-time	2.2000	2.7667	6.2667	10.1333	3.9000	7.5333	8.8667	**1.6000**	6.2667	8.0667	8.4000
Average	**3.3500**	7.1667	6.5000	7.0167	7.3556	5.5722	6.9833	6.8833	5.3556	4.3667	5.4500
Rank	**2.0000**	7.7500	6.5833	8.0000	7.9167	5.3333	7.6667	7.8333	5.2500	3.0000	4.6667

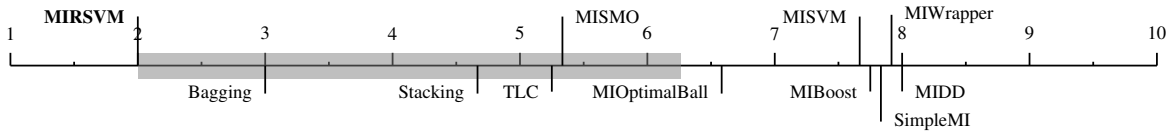

Figure 10: Bonferroni-Dunn test for overall ranks comparison

4.7. Overall Comparison

Table 13 shows the run time results, in seconds, for each algorithm. MIRSVM has the fastest run time and is ranked second. MIRSVM shows very good scalability considering the number of features, such as in the webmining dataset which comprises of 5863 attributes. Additionally, taking into account the number of instances as seen in the two largest datasets, component and function, MIRSVM displays superior scalability. It is important to note that quadratic programming solvers are not the most efficient tools for solving optimization problems in terms of run time, and yet MIRSVM still is shown to perform competitively against the current state-of-the-art algorithms.

SimpleMI achieves the highest rank and competitive run times because, rather than use the instances in each bag to train a model, it takes the mean value of the instances in a bag and uses that for training. Even though SimpleMI has fast run-times, its performance over the previous metrics has been shown to be random and not as efficient as the bag-level methods.

Table 14 shows the ranks achieved by each of the metrics along with the average and meta-ranks. MIRSVM has the best meta-rank (rank of the ranks) and the Bagging ensemble method has the next best. The meta-ranks also highlight the better performance of bag-level methods over instance-level and wrapper methods, emphasizing the importance of training at the bag-level. Not only does MIRSVM use bag-level information during classification, but it also optimizes over the instances within the bag, which helps determine which instances contribute the most information about the bags label. SimpleMI, MIWrapper, MIBoost, MISVM, MIOptimalBall, and MDD have the worst performance compared to MIRSVM and Bagging. Although these algorithms are popular in literature, the experimental study clearly shows that recent bag-level and ensemble methods easily overcome traditional multi-instance learning algorithms.

In summary, MIRSVM offers improvement in terms of both accuracy and run-time when compared to referenced methods, especially those utilizing SVM-based algorithms.

5. Conclusion

This paper proposed a novel formulation and algorithm for the multiple-instance support vector machine problem, which optimizes bags classification via bag-representative selection. First, the primal formulation was posed and its dual was then derived and computed using a quadratic programming solver. This formulation was designed to utilize bag-level information and find an optimal separating hyperplane between bags, rather than individual instances, using the standard multi-instance assumption. The SMI assumption states that a bag is labeled positive if and only if at least one instance within a bag is positive, and is negative otherwise. The key features of the proposed algorithm MIRSVM are its ability to identify instances within positive and negative bags, i.e. the support vectors or representatives, that highly impact the decision boundaries as well as avoiding uncertainties and issues caused by techniques that flatten, subset, or under-represent positive instances within positively labeled bags. Additionally, it exhibits desirable scalability, making it suitable for large-scale learning tasks.

The experimental study showed the better performance of MIRSVM compared with multi-instance support vector machines, traditional multi-instance learners, as well as with ensemble methods. The results, according to a variety of performance metrics, were compared and further validated using statistical analysis with non-parametric tests which highlight the advantages of using bag-level based and ensemble learners such as Bagging and Stacking, while showing the instance-level based learners performed poorly in comparison or were deemed as strongly biased

and unstable classifiers. On the contrary, our proposal MIRSVM performs statistically better, neither compromising accuracy nor run-time while displaying a robust performance across all of the evaluated datasets.

Acknowledgment

This research was supported by the Spanish Ministry of Economy and Competitiveness, project TIN2014-55252-P, and by FEDER funds.

References

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework, Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17:255–287, 2011.

[2] E. Alpaydin, V. Cheplygina, M. Loog, and D.M.J. Tax. Single- vs. multiple-instance classification. *Pattern Recognition*, 48(9):2831–2838, 2015.

[3] J. Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105, 2013.

[4] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support Vector Machines for Multiple-Instance Learning. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, pages 577–584, 2002.

[5] P. Auer and R. Ortner. A Boosting Approach to Multiple Instance Learning. In *Proceedings of the 15th European Conference on Machine Learning*, volume 3201 LNCS, pages 63–74, 2004.

[6] A. Ben-David. About the Relationship Between ROC Curves and Cohens Kappa. *Engineering Applications of Artificial Intelligence*, 21(6):874–882, 2008.

[7] L. Bjerring and E. Frank. Beyond Trees: Adopting MITI to Learn Rules and Ensemble Classifiers for Multi-instance Data. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pages 41–50, 2011.

[8] M. Blaschko and T. Hofmann. Conformal Multi-instance Kernels. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 1–6, 2006.

[9] H. Blockeel, D. Page, and A. Srinivasan. Multi-instance tree learning. In *Proceedings of the International Conference on Machine Learning*, pages 57–64, 2005.

[10] S. Boyd and L. Vanderberghe. *Convex Optimization*. Cambridge University Press, 2004.

[11] R. Bunescu and R. Mooney. Multiple Instance Learning for Sparse Positive Bags. In *Proceedings of the Annual International Conference on Machine Learning*, pages 105–112, 2007.

[12] A. Cano, A. Zafra, and S. Ventura. Speeding up multiple instance learning classification rules on GPUs. *Knowledge and Information Systems*, 44(1):127–145, 2015.

[13] M.A. Carbonneau, E. Granger, A.J. Raymond, and G. Gagnon. Robust multiple-instance learning ensembles using random subspace instance selection. *Pattern Recognition*, 58:83–99, 2016.

[14] Y. Chen and J. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

[15] Y. Chen, J. Bi, and J. Wang. MILES: Multiple-Instance Learning via Embedded Instance Selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1931–1947, 2006.

[16] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[17] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[18] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

[19] L. Dong. A comparison of multi-instance learning algorithms. *Master of Science Thesis, The University of Waikato*, 2006.

[20] O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

[21] A.W.C. Faria, F.G.F. Coelho, A.M. Silva, H.P. Rocha, G.M. Almeida, A.P. Lemos, and A.P. Braga. Milkde: A new approach for multiple instance learning based on positive instance selection and kernel density estimation. *Engineering Applications of Artificial Intelligence*, 59:196–204, 2017.

[22] J. Foulds and E. Frank. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25-1:1–24, 2010.

[23] Z. Fu, A. Robles-Kelly, and J. Zhou. MILIS: Multiple Instance Learning with Instance Selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):958–977, 2011.

[24] S. García and F. Herrera. An Extension on Statistical Comparisons of Classifiers over Multiple Data Sets for all Pairwise Comparisons. *Journal of Machine Learning Research*, 9: 2677–2694, 2008.

[25] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour - a case study on the CEC2005 Special Session on Real Parameter Optimization. *Heuristics*, 15:617–644, 2008.

[26] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, 2010.

[27] T. Gärtner, P.A. Flach, A. Kowalczyk, and A.J. Smola. Multi-instance kernels. In *Proceedings of the International Conference on Machine Learning*, pages 179–186, 2002.

[28] J.D. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference*. Chapman & Hall/CRC Press, 5th edition, 2011.

[29] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemannr, and I.H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11:10–18, 2009.

[30] G. Herman, G. Ye, J. Xu, and B. Zhang. Region-based image categorization with reduced feature set. In *Proceedings of the 10th IEEE Workshop on Multimedia Signal Processing*, pages 586–591, 2008.

[31] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D. Sánchez-Tarragó, and S. Vluymans. *Multiple Instance Learning Foundations and Methods*. Springer, 2016.

[32] M. Hollander and D.A. Wolfe. *Nonparametric Statistical Methods*. John Wiley & Sons, Inc., 1999.

[33] T.M. Huang, V. Kecman, and I. Kopriva. *Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi- supervised, and Unsupervised Learning*. Springer-Verlag, 2006.

[34] V. Kecman. *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. MIT Press, 2001.

[35] V. Kecman. Iterative k Data Algorithm for Solving Both The Least Squares SVM and The System of Linear Equations. In *Proceedings of the IEEE SoutheastCon*, pages 1–6, 2015.

[36] V. Kecman and L. Zigic. Algorithms for Direct L2 Support Vector Machines. In *Proceedings of the IEEE International Symposium on Innovations in Intelligent Systems and Applications*, pages 419–424, 2014.

[37] V. Kecman, T.M Huang, and M. Vogt. Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance. *Studies in Computational Intelligence*, 177:255–274, 2005.

[38] O. Maron T. Lozano-Pérez. A framework for multiple-instance learning. *Neural Information Processing Systems*, 3201:570–576, 1998.

[39] J.M. Luna, A. Cano, V. Sakalauskas, and S. Ventura. Discovering useful patterns from multiple instance data. *Information Sciences*, 357:23–38, 2016.

[40] G. Melki and V. Kecman. Speeding up online training of l1 support vector machines. In *Proceedings of the IEEE SoutheastCon*, pages 1–6, 2016.

[41] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 2006.

[42] J. Platt. Sequantial Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. *Technical Report: MSR-TR-98-14*, 1998.

[43] Xiaojun Qi and Yutao Han. Incorporating multiple svms for automatic image annotation. *Pattern Recognition*, 40(2):728–741, 2007.

[44] S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceeding of the International Conference on Machine Learning*, pages 697–704, 2005.

[45] B. Schöelkopf and A. Smola. *Learning with Kernels; Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[46] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[47] Gitte Vanwinckelen, Vinicius Tragante do O, Daan Fierens, and Hendrik Blockeel. Instance-level accuracy versus bag-level accuracy in multi-instance learning. *Data Mining and Knowledge Discovery*, 30(2):313–341, 2016.

[48] P. Viola, J.C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *Advances in Neural Information Processing Systems*, pages 1417–1424, 2005.

[49] J. Wang and J. Zucker. Solving the multiple-instance problem: a lazy learning approach. In *Proceedings of the International Conference on Machine Learning*, pages 1119–1126, 2000.

[50] R. Wang and S. Kwong. Active learning with multi-criteria decision making systems. *Pattern Recognition*, 47(9):3106–3119, 2014.

[51] X. Wang, X. Liu, S. Matwin, N. Japkowicz, and H. Guo. A multi-view two-level classification method for generalized multi-instance problems. In *Proceeding of the IEEE International Conference on Big Data*, pages 104–111, 2014.

[52] X. Xu. Statistical learning in multiple instance problem. *Master's thesis, University of Waikato*, 2003.

[53] Y.Y. Xu and C.H. Shih. Multiple-instance learning via decision-based neural networks. *Intelligent Decision Technologies*, 10:885–895, 2011.

[54] C.Y.C Yang, M.D.M Dong, and J.H.J Hua. Region-based image annotation using asymmetrical support vector machine-based multiple-instance learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2057–2063, 2006.

[55] Y. Yi and M. Lin. Human action recognition with graph-based multiple-instance learning. *Pattern Recognition*, 53:148–162, 2016.

[56] Q. Zhang and S. Goldman. Em-DD: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems*, pages 1073–1080, 2002.