Journal of Global Optimization
https://doi.org/10.1007/s10898-022-01219-y

Multi-instance learning by maximizing the area under receiver operating characteristic curve

I. Edhem Sakarya[1] · O. Erhun Kundakcioglu[2]

Received: 4 November 2021 / Accepted: 24 July 2022

Abstract

The purpose of this study is to solve the multi-instance classification problem by maximizing the area under the Receiver Operating Characteristic (ROC) curve obtained for witness instances. We derive a mixed integer linear programming model that chooses witnesses and produces the best possible ROC curve using a linear ranking function for multi-instance classification. The formulation is solved using a commercial mathematical optimization solver as well as a fast metaheuristic approach. When the data is not linearly separable, we illustrate how new features can be generated to tackle the problem. We present a comprehensive computational study to compare our methods against the state-of-the-art approaches in the literature. Our study reveals the success of an optimal linear ranking function through cross validation for several benchmark instances.

Keywords Multi-instance learning · Mixed integer linear programming · Area under curve

1 Introduction

Supervised learning studies pairs of $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is a set of features (attributes) for instance $i \in I$ and y_i is the corresponding response or label [1]. The goal is to learn a mapping function from input matrix $\boldsymbol{X}$ to vector $\boldsymbol{y}$ with approximations. This mapping function is used for prediction of an output variable, when there is new input data. Multi-instance learning (MIL) identifies this mapping over *bags* rather than instances. A bag is a collection of instances that are collectively labeled, where instance labels are not necessarily known. One of the classical examples is drug activity classification, where the drugs can be labeled as positive or negative depending on their effectiveness. A positive label indicates a drug is effective, that is an effective conformation of that drug exists. However, that does not *necessarily* imply all these conformations are effective. This and similar ambiguities

✉ O. Erhun Kundakcioglu
erhun.kundakcioglu@ozyegin.edu.tr

[1] Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven, Netherlands

[2] Department of Industrial Engineering, Ozyegin University, Istanbul, Turkey

Published online: 12 August 2022

in instance labels is what makes MIL settings particularly attractive and computationally challenging.

Supervised learning algorithms such as Bayesian classifiers, logistic regression, decision trees, random forests, neural networks, support vector machines map input matrix $\boldsymbol{X}$ to vector $\boldsymbol{y}$, but with different approaches and assumptions [2]. These procedures mostly arise from numerical analysis or optimization theory depending on the type of data mining problems. Indeed, diversity among these algorithms demonstrate the need of various applications with different settings. Each method addresses a particular level of sufficiency on computational complexity and performance offerings. That is why there is no *best method* for any type of data mining problem. This motivates our research on observing the performance of an exact optimization method on a key metric.

Decision support and machine learning literature have increasing attention on binary classification problems [3]. We study multi-instance binary classification with bags of data. Under the widely-used *standard assumption*, positive labeled bags contain at least one positive instance (*witness*) and negative labeled bags are full of negative instances [4, 5]. This approach leads to a straightforward transition from instance level prediction to bag level prediction, when a classifier is obtained. A bag is predicted as positive unless all instances in the bag are predicted to be negative. Several decision problems benefit from multi-instance learning in domains such as image annotation, economic predictions, audio processing, and text mining [6].

1.1 Related literature

There are several aspects of MIL. Aside from the standard assumption we adopt, there is also *collective assumption*, where one witness instance is not enough; but a distribution, an interaction, or an accumulation of instances is desired [7, 8]. Further approaches based on presence, threshold, and count can be found in [9]. Further characteristics of the MIL problem arises from prediction level, bag composition, data distributions, and label ambiguity [8]. The transition from classical instance label predictions to *bag label predictions* are made by existence of a witness, rate of witnesses, or relationship between instances. *Bag compositions* are studied using relationship between instances with measures such as intra-bag similarities, instance co-occurrences, or structure. *Data distributions* are examined using multimodal distributions of positive instances and non-representative negative distribution. The former refers to shapes of distributions, if positive instances are located in one cluster or not. The latter refers to hardness of modeling the negative class distributions. Finally, there are variations of *ambiguities*: aside from the standard label ambiguity (on positive bags), label noise and different label spaces are studied.

MIL is a widely-studied data-driven decision support mechanism. Zhou [10] introduces the origin of MIL and discusses the learning algorithms, applications, and extensions. A unified view of the most frequently used approaches is also presented: Diverse Density [11], Citation-*k*NN [12], ID3-MI and RIPPER-MI [13], and BP-MIP [14]. Appraches to MIL problems utilize a cost function that is optimized during training, based on *maximum likelihood* or *margin* [15]. Maximum likelihood is frequently used in standard supervised learning due to its flexibility. Margin based approaches focus on the distance between hyperplane and data points, which is maximized for classification and minimized for regression.

Binary classification is one of the most widely-studied problems in MIL. Dietterich et al. [16] present and compare three different types of algorithms that solve drug activity prediction problem via learning axis-parallel rectangles. Maron and Lozano-Pérez [11] describe the idea of diverse density through a motivating example of moleculer shape, where molecules are represented as manifolds in an n-dimensional space; and labeled positive if at least one place along the molecules' manifold fit into the target protein. Andrews et al. [17] derive two new mixed integer quadratic programming formulations to solve multi-instance classification problem as extensions to Support Vector Machines (SVMs). They can only solve these models heuristically but report excellent classification results. In contrast to the widely adopted standard assumption, Weidmann et al. [9] approach the MIL from a different angle, considering the labels of bags according to the interaction between instances. Kundakcioglu et al. [18] derive a new combinatorial optimization formulation and solve the margin maximization problem, in accordance with SVMs, for multi-instance binary classification. They show that their formulation is $\mathcal{NP}$-hard, present how kernel trick can be applied, and propose a branch and bound algorithm to solve their formulation. There are studies that address a major practical issue for MIL: robustness of classifiers to outliers. Poursaeidi and Kundakcioglu [19] propose using hard margin loss formulations and Carbonneau et al. [20] present a random subspace instance selection procedure to provide a more robust classifier for different data distributions. More recent approaches employ deep-learning algorithms that usually yield a higher testing accuracy [21, 22].

1.2 Our contribution

This work circumvents the search for an algorithm to solve classification problems within MIL. Based on [23], we develop a mixed integer linear programming model for multi-instance binary classification. Our model maximizes the area under Receiver Operating Characteristic (ROC) Curve (i.e., AUC) for the witness instances of an MIL problem. AUC is known to be the golden standard in measuring the success of a classifier [24, 25]. On the development of our model, we make use of standard MIL assumption and witness selection procedure. We take into account the witness (a single instance) from each positive bag. Similar to the model presented in [23], selection of witness instances for positive bags is performed by pairwise comparison of instances within the bags. This is essentially a *supervised bipartite ranking problem*, which is synonymous with AUC maximization in binary classification [26]. This also explains why these problems are generally favored in the machine learning community [27].

Small and medium-sized problems can be solved to optimality in acceptable time using our model with a commercial solver. Our approach also provides a bound on the optimal objective value for large-scale problems. Furthermore, model objective can be altered based on different rank statistics for different cases or specialized problems, making this approach flexible. As our main concern is the generalization performance of the classifier, an exact solution approach with a commercial solver might be impractical for large cases. In order to address this issue, we also present how a metaheuristic approach, namely Particle Swarm Optimization (PSO), can be used to solve the optimization problem. PSO scales up better than the exact approach in terms of computation time, and our results show that the solution quality and generalization performance are acceptable. We provide benchmark results on well-known multi-instance classification datasets for our exact and metaheuristic approach against five algorithms from the literature. Furthermore, in order to perform nonlinear classification, we accommodate a feature generation approach from the literature and present numerical results.

1.3 Organization

The remainder of this paper is organized as follows: In Sect. 2, we derive a novel mathematical formulation for the multi-instance classification problem. We develop a reformulation that can be solved more effectively and present a metaheuristic approach that provides fast near-optimal solutions in Sect. 3. We compare our methods with the state-of-the-art approaches in the literature using publicly available datasets in Sect. 4. We provide concluding remarks and directions for future research in Sect. 5.

2 Problem definition

In this section, we present the foundations of our approach, and elaborate on the details of our mathematical optimization model.

2.1 Multi-instance bipartite ranking

The *supervised bipartite ranking* problem consists of a set of training instances and their associated binary labels, hence the term *supervised*. In binary classification, classes are usually labeled as $+1$ or -1, to denote positivity or negativity, i.e., to show the (non)existence of some characteristic for the instance. *Scorer function* is crucial in bipartite ranking problem, which assigns a real number to each instance. Without loss of generality, positive instances are expected to have higher scores, whereas negative instances are expected to have relatively lower scores. In fact, it would be the ideal ranking if all negative instances have a lower score than each and every one of the positive instances. This is formalized with the following iterative procedure that assigns different ranks to each instance:

- *Minimum rank* for each instance is the number of instances that have a strictly lower score that that instance.
- *Rank* of each instance is equal to the minimum rank unless there is another instance with the same minimum rank.
- In case of more than one instance with the same minimum rank (i.e., same score), positive instances receive lower ranks. Among more than one positive or more than one negative, ties are broken arbitrarily.

The ultimate goal is to minimize the number of *misranks*. There occurs a *misrank*, if a negative instance has a higher rank than a positive instance. That implies a score for a negative instance that is greater than or equal to that of a positive instance.

Bertsimas et al. [23] study a *supervised bipartite ranking* problem for single instance classification. Our approach in this notation is to transform the single instance classification problem to a multi-instance classification problem. Note that labels are defined over bags of instances with a *standard assumption*. Thus, the *multi-instance bipartite ranking problem* can be formalized similar to the single instance case, except for the definition of misranking.

Remark 1 The goal of multi-instance bipartite ranking problem is to minimize the number of misranks across **bags**. There occurs a misrank, if an instance from a negative bag has a higher rank than all instances in a positive bag. That implies the largest score among instances of a negative bag is greater than or equal to the largest score among instances of a positive bag.

It should be noted that we use a linear ranking/scoring function in this study. That is $f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i$ is the scorer function, where $\mathbf{w} \in \mathbb{R}^d$ is the orthogonal of the ranking hyperplane.

2.2 Maximizing the area under ROC curve

ROC curve is developed around the 1950s to detect signals. ROC curve is constructed by plotting true positive rate (TPR) on y axis and false positive rate (FPR) on x axis for various values of a threshold parameter; from one extreme value that classifies all instances as negative (i.e., TPR=0, FPR=0) to the other extreme that classifies all instances as positive (i.e., TPR=1, FPR=1). For a random classification, curve is expected to be a straight line from (0,0) to (1,1), therefore any classifier that outperforms a random classifier should dominate this line. AUC stands for the *area under ROC curve* and is the ultimate measure for success in classification tasks.

AUC metric is proven to be also obtained by pairwise comparison of rankings: There is a three stage proof for this proposition. First, Green and Swets [28] prove the equality of AUC and probability $P(X > Y)$, where X and Y are the random variables generated using the distribution of the positive and negative examples, respectively. Next, Hanley and McNeil [29] prove that the Wilcoxon-Mann-Whitney statistic is exactly equal to the aforementioned probability in discrete cases, hence the AUC as follows:

$$P(X > Y) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{L} 1_{m_i > n_j}}{KL} \tag{1}$$

In (1), m_1,...,m_K and n_1,...,n_L are the outputs of a fixed classifier for positive and negative data points respectively. Finally, AUC's relation with ranking quality of the classification and pairwise comparison is explained and statistically proven in [30]. In the light of these, our approach of directly maximizing AUC for the witnesses of the multi-instance classification problem, we expect to maximize the probability of correctly classifying new data generated coming from the distributions of (i) positive witnesses and (ii) negative instances.

2.3 Multi-instance AUC maximization model

In this section we present our mathematical modeling approach. Our multi-instance AUC maximizer is formulated as a mathematical optimization formulation for the multi-instance bipartite ranking problem. Tables 1 and 2 contain the nomenclature we use in our mathematical developments.

Next, we present the initial version of our multi-instance bipartite ranking formulation as:

$$\textbf{[MI-BR]} \qquad \max \sum_{p \in S^+} \sum_{n \in S^-} \Omega_{pn} \tag{2a}$$

$$\text{subject to } v_i = \mathbf{w}^T\mathbf{x_i} \qquad i \in I^+ \cup I^- \tag{2b}$$

$$-1 \le w_j \le 1 \qquad j \in \{1, \dots, d\} \tag{2c}$$

$$z_{ik} \le v_i - v_k + 1 - \epsilon \qquad i \in I^+,\ k \in I^- \tag{2d}$$

$$\sum_{k \in I_n} z_{ik} \ge |I_n|\gamma_{in} \qquad i \in I^+,\ n \in S^- \tag{2e}$$

Table 1 Nomenclature — Sets and Parameters

Sets and Parameters	Description
d	Number of features, i.e., number of dimensions for feature space
I^+	Set of instances in positive bags
I^-	Set of instances in negative bags
$\mathbf{x}_i \in \mathbb{R}^d$	Vector of features for data instance $i \in I^+ \cup I^-$
S^+	Set of positive bags
S^-	Set of negative bags
I_p	Set of instances in bag $p \in S^+ \cup S^-$

Table 2 Nomenclature — Decision Variables

Decision Variables	Description
v_i	Score of instance $i \in I^+ \cup I^-$
w_j	Hyperplane coefficient variable $j \in \{1, \ldots, d\}$
$\mathbf{w}$	Hyperplane vector of variables
z_{ik}	1 if score of instance $i \in I^+$ is larger than instance $k \in I^-$, 0 otherwise
γ_{in}	1 if score of instance $i \in I^+$ is larger than all instances in bag $n \in S^-$, 0 otherwise
θ_i	1 if instance $i \in I^+$ is selected as a witness instance, 0 otherwise
χ_{in}	1 if $i \in I^+$ is a witness instance and score of $i \in I^+$ is larger than all instances in bag $n \in S^-$, 0 otherwise
Ω_{pn}	1 if score of witness instance of positive bag $p \in S^+$ is larger than all instances in bag $n \in S^-$, 0 otherwise

$$\sum_{i \in I_p} \theta_i = 1 \qquad p \in S^+ \tag{2f}$$

$$\theta_i + \gamma_{in} \geq 2\chi_{in} \qquad i \in I^+,\ n \in S^- \tag{2g}$$

$$\sum_{i \in I_p} \chi_{in} = \Omega_{pn} \qquad p \in S^+,\ n \in S^- \tag{2h}$$

$$z_{ik} \in \{0, 1\} \qquad i \in I^+,\ k \in I^- \tag{2i}$$

$$\theta_i \in \{0, 1\} \qquad i \in I^+ \tag{2j}$$

$$\chi_{in}, \gamma_{in} \in \{0, 1\} \qquad i \in I^+,\ n \in S^- \tag{2k}$$

$$\Omega_{pn} \in \{0, 1\} \qquad p \in S^+,\ n \in S^- \tag{2l}$$

Our objective function (2a), as mentioned in previous sections, maximizes correct rankings between pairs of bags. Constraint (2b) computes the scores of all instances. Constraint (2c) is for scaling as well as bounding the feasible region to speed up the solution procedure. Constraint (2d) ensures correct ranking among each pair of instances is reflected with the associated binary variable. Here, ϵ is a small enough number that has to be input by the user to ensure same scored positive and negative instances are misranked, as explained in Sect. 2.1. Constraint (2e) implies that if a positive instance scores greater than *all* instances of a negative bag, then the associated binary variable reflects that. Constraint (2f) ensures that there can be only one witness instance in a positive bag. Constraints (2g) and (2h) guarantee if an instance of a positive bag is a witness and scores greater than all instances of a negative bag, then these two bags are correctly ranked.

It should be noted that the formulation in (2) is a mixed integer linear programming problem, which are computationally challenging unless they possess a special structure. Even the single instance bipartite ranking problem, which is a special case of multi-instance problem with bags of size one, is $\mathcal{NP}$-hard [23]. Thus, we conclude that the formulation (2) cannot be solved efficiently. The binary restrictions for Ω_{pn} can be relaxed for all $p \in S^+,\ n \in S^-$ and the solution would still be integral. However, there are still $|I^+||I^-| + |I^+| + 2|I^+||S^-| \sim \mathcal{O}(|I^+||I^-|)$ binary variables. Next, we present our solution approaches for this challenging problem.

3 Solution approaches

In this section, we present our approach in solving the mathematical optimization formulation in (2). We propose an exact approach that uses a reformulation and a commercial solver as well as a metaheuristic approach.

3.1 Reformulation for an exact approach

The mathematical model we propose in the previous section aims to find *the best* ranking hyperplane in terms of ROC performance. However, the model is computationally intractable, making it impractical for a validation scheme, where the model needs to be solved several times. and not eligible to be solved in commercial solvers in a reasonable time and also not suitable for cross validation. Therefore, to speed up the solution, we propose reformulations on the constraints and variables.

First, in proposed model we have four sets of binary variables in $\mathcal{O}(|I^+||S^-|)$. In order to address the computational burden created due to these binary variables, we transformed constraint (2e) to a tighter constraint as

$$z_{ik} \geq \gamma_{in} \qquad i \in I^+,\ k \in I_n,\ n \in S^-. \tag{3}$$

Instead of constraint (2g), we use

$$\theta_i \geq \chi_{in} \qquad i \in I^+,\ n \in S^-, \tag{4a}$$

$$\gamma_{in} \geq \chi_{in} \qquad i \in I^+,\ n \in S^-. \tag{4b}$$

After these modifications, θ_i, γ_{in} and χ_{in} can be relaxed to a continuous variable between 0 and 1 even though they can only take the values 0 and 1, due to the tightness of the renewed feasible region definition. Thus, we have

$$0 \leq \theta_i \leq 1 \qquad\qquad i \in I^+, \tag{5a}$$

$$0 \leq \chi_{in} \leq 1 \qquad i \in I^+,\ n \in S^-, \tag{5b}$$

$$0 \leq \gamma_{in} \leq 1 \qquad i \in I^+,\ n \in S^-. \tag{5c}$$

After these modifications, variables γ_{in} for $i \in I^+, n \in S^-$ become auxiliary, and can be omitted without changing the optimal solution. The associated constraints in (3) and (4b) can then be replaced with

$$z_{ik} \geq \chi_{in} \qquad i \in I^+,\ k \in I_n,\ n \in S^-. \tag{6}$$

and our enhanced multi-instance bipartite ranking formulation can be presented as follows:

$$\textbf{[eMI-BR]} \qquad \max \sum_{p\in S^+}\sum_{n\in S^-}\Omega_{pn} \tag{7a}$$

$$\text{subject to} \quad v_i = \mathbf{w}^T\mathbf{x_i} \qquad i\in I^+\cup I^- \tag{7b}$$

$$-1\le w_j\le 1 \qquad j\in\{1,\dots,d\} \tag{7c}$$

$$z_{ik}\le v_i - v_k + 1 - \epsilon \qquad i\in I^+,\ k\in I^- \tag{7d}$$

$$z_{ik}\ge \chi_{in} \qquad i\in I^+,\ k\in I_n,\ n\in S^- \tag{7e}$$

$$\sum_{i\in I_p}\theta_i = 1 \qquad p\in S^+ \tag{7f}$$

$$\theta_i\ge\chi_{in} \qquad i\in I^+,\ n\in S^- \tag{7g}$$

$$\sum_{i\in I_p}\chi_{in} = \Omega_{pn} \qquad p\in S^+,\ n\in S^- \tag{7h}$$

$$0\le\Omega_{pn}\le 1 \qquad p\in S^+,\ n\in S^- \tag{7i}$$

$$0\le\theta_i\le 1 \qquad i\in I^+ \tag{7j}$$

$$0\le\chi_{in}\le 1 \qquad i\in I^+,\ n\in S^- \tag{7k}$$

$$z_{ik}\in\{0,1\} \qquad i\in I^+,\ k\in I^- \tag{7l}$$

Solving **[eMI-BR]** with the multi-instance *training* data, we obtain the coefficients of the ranking hyperplane (**w**), which performs the bipartite ranking, thus classification of bags. To clarify, z_{ik} are the only binary variables defined in this formulation that is defined for each $i\in I^+, k\in I^-$ pair. There are still $\mathcal{O}(|I^+||I^-|)$ binary variables, similar to **[MI-BR]**, but the number of binary variables are reduced by $|I^+| + 2|I^+||S^-|$. The binary variables in **[MI-BR]** that are relaxed are bounded from above by z_{ik}, and are not aggregated except for (7h). Thus, the optimal objective function value for **[MI-BR]** is always equal to that of **[eMI-BR]**, and the integral optimal solution can be obtained using **[eMI-BR]**. The following lemma formally proves this.

Theorem 3.1 *The optimal objective function values of* **[eMI-BR]** *and* **[MI-BR]** *are equal and an integral optimal solution of* **[MI-BR]** *is an optimal solution for* **[eMI-BR]**.

Proof The proof is by contradiction. The objective functions of **[MI-BR]** and **[eMI-BR]** are the same, and equal to $\sum_{p\in S^+}\sum_{n\in S^-}\sum_{i\in I_p}\chi_{in}$, using (2h) and (7h), respectively. Suppose that the optimal objective function value of **[eMI-BR]** ($z^{\text{eMI-BR}}$) and the optimal objective function value of **[MI-BR]** ($z^{\text{MI-BR}}$) are different. In the same fashion, assume each problem name as a superscript denotes the associated value of a variable at optimality for the corresponding problem.

First, suppose $z^{\text{eMI-BR}} < z^{\text{MI-BR}}$. It can be seen that a feasible (hence integral) solution of **[MI-BR]** is guaranteed to be feasible for **[eMI-BR]**. As a tighter formulation can never have a more favorable objective function value and we have a maximization problem, we conclude $z^{\text{eMI-BR}} < z^{\text{MI-BR}}$ cannot be valid.

Next, suppose $z^{\text{eMI-BR}} > z^{\text{MI-BR}}$. Notice that any ranking hyperplane is feasible for both problems. That implies the optimal ranking hyperplane for **[eMI-BR]**, $\mathbf{w}^{\text{eMI-BR}}$, while being feasible to **[MI-BR]**, definitely has a worse objective function value. Formally, there exists at least one positive bag $p'\in S^+$ and one negative bag $n'\in S^-$, where $\sum_{i\in I_{p'}}\chi_{in'} = 0$, when $\mathbf{w}^{\text{eMI-BR}}$ is used on (2b–2l), whereas $\sum_{i\in I_{p'}}\chi_{in'} > 0$, when $\mathbf{w}^{\text{eMI-BR}}$ is used on (7b–7l). That implies there exists an $i' \in I_{p'}$ with $\chi_{i'n'} > 0$. The only way this is possible within (7b–7l) is if both of the following two conditions hold:

1. $z_{i'k} = 1, \forall k \in I_{n'}$, that is the score of instance i' is larger than all instances in bag n' as per (7e), and
2. $\theta_i' > 0$, that is instance i' is selected as one of the witnesses[1] within its bag as per (7g).

Now we use these facts to see how constraints (2b–2l) come into play. Condition 1 ensures that $\gamma_{i'n'} = 1$ is possible as per (2e). Using condition 2, witness selection constraint (2f) can be satisfied by choosing i' as the *only* witness within that bag, i.e., $\theta_{i'} = 1$. Note that, despite the fact that there might exist other instances with $z_{ik} = 1$, one such instance with witness variable set to one is sufficient. Using (2g) with $\gamma_{i'n'} = 1$ and $\theta_{i'} = 1$, we obtain $\sum_{i \in I_{p'}} \chi_{in'} = 1$, when $\mathbf{w}^{\text{eMI-BR}}$ is used on (2b–2l), which is a contradiction.

□

Testing Procedure. Once *training* is completed, we use the output ranking vector $\mathbf{w}^*$, with the same mathematical model for *testing* with the following updates:

1. Testing instances are used as vector of features
2. Decision variable vector is enforced to satify $\mathbf{w} = \mathbf{w}^*$ through additional constraints
3. Update constraint (7d) as follows:

$$M z_{ik} \leq v_i - v_k + M - \epsilon \qquad i \in I^+,\ k \in I^-, \tag{8}$$

where M is a sufficiently large number, input by the user.

4. Solve to obtain the optimal values of remaining decision variables, ultimately Ω_{pn} values to compute the *testing AUC*

The procedure is straightforward except the constraint modification in step 3. This modification is needed due to possible differences in the scale of training and test data. In other words, constraint (7d) also scales the ranking vector based on training data, ensuring a maximum functional distance between pairs of instances, i.e., either $v_i - v_k + 1 - \epsilon \geq 0$ or $v_i - v_k + 1 - \epsilon \geq 1$. However, neither one of these possibilities ($v_i - v_k \geq -1 + \epsilon$, $v_i - v_k - \epsilon \geq 0$) are satisfied if $v_k >> v_i$ for some pair $i \in I^+,\ k \in I^-$ in the test data. In order to address this issue, we propose using the constraint in (8), which ensures z_{ik} becomes 1 if $v_i > v_k$, and becomes 0 otherwise, without any issue, no matter the functional distance. It should also be noted that, these undesired so-called fixed-charge constraints do not cause a major computational issue here, because AUC computation for the testing set with fixed $\mathbf{w}$ is not an optimization challenge, but a straightforward evaluation of functions.

3.2 A metaheuristic algorithm: particle swarm optimization

Besides solving the proposed mathematical optimization model, we also propose a metaheuristic method to solve AUC maximization problem. We employ the widely-used Particle Swarm Optimization Algorithm (PSO) [31]. PSO starts with an initial population of particles, each representing a solution, that are traveling through the solution space with random velocities. These particles have their own memories and keep track of their previous best positions ($\mathbf{p}_{best}$) and corresponding objective values throughout the iterations. They also record the

[1] In **[eMI-BR]**, the witness selection variable is relaxed, which might technically lead to more than one variable in the same bag having nonzero values. However, as shown later in the proof, either one of these instances can be chosen as a witness under the standard assumption.

information on the best position of the whole swarm ($\mathbf{g}_{best}$) and its objective value. The idea of PSO is to change the direction of each particle's movement towards its best position and best position of all particles. Velocity ($\mathbf{V}_{it}$) and position ($\mathbf{w}_{it}$) of each particle $i \in I^{\text{PSO}}$ at each discrete time epoch $t \in T^{\text{PSO}}$ is updated as follows:

$$\mathbf{V}_{it} = \mathbf{V}_{i(t-1)} + \rho_1 C_1(\mathbf{p}_{best} - \mathbf{w}_{it}) + \rho_2 C_2(\mathbf{g}_{best} - \mathbf{w}_{it}) \tag{9}$$

$$\mathbf{w}_{i,t+1} = \mathbf{w}_{it} + \mathbf{V}_{it} \tag{10}$$

In these equations, ρ_1 and ρ_2 are randomly generated numbers that are uniformly distributed in [0, 1], C_1 and C_2 are user-defined parameters for respective weights for particle's own best position and the best position of whole swarm. I^{PSO} is the number of particles (swarm size) and T^{PSO} is the maximum number of time epochs considered (epoch length). In our solution approach, we consider each particles position $\mathbf{w}_{it}$ as a ranking hyperplane and the associated AUC as the corresponding objective value to each particle, which is calculated using the Wilcoxon-Mann-Whitney statistic in (1). Each particle is enclosed in a d-dimensional hypercube with boundaries at -1 and 1 in each dimension. That is, if the particle is out of bounds on one dimension, we force it to stay at the bound it exceeds. For efficiency, we also add a convergence criterion to our algorithm, that is checked at the end of each epoch. The algorithm terminates if the best solution has not changed in the last 15 iterations, that is it has converged to a high quality solution. The pseudocode of the PSO algorithm is presented below, and a detailed flowchart is in Appendix A.

Algorithm 1 Particle Swarm Optimization Algorithm for MIL

```
Initialize particles
while Termination condition is not met do
    for each particle do
        Calculate objective value using Eq. (1)
        if Objective value is better than particle best ($\mathbf{p}_{best}$) then
            Update $\mathbf{p}_{best}$
            if $\mathbf{p}_{best}$ is better than global best ($\mathbf{g}_{best}$) then
                Update $\mathbf{g}_{best}$
        Update velocity and position for the particle using Eqs. (9) and (10)
    Update termination condition status
```

4 Computational results

In this section, we present the benchmark results for our approaches against the state-of-the-art algorithms in the literature. In particular, we compare our results with well-known multi-instance learning methods such as kNN, Citation-*k*NN [12], EM-DD [32], MI-SVM, mi-SVM [17], and more recent *deep-learning* algorithms such as mi-Net, MI-Net, MI-Net with RC [21], and an *attention-based deep-learning* method that incorporates interpretability [22]. We also use techniques in [33] to extract additional features for nonlinear classification; and compare our results against theirs, where a mathematical optimization approach is proposed as well.

Datasets. Table 3 provides information on some of the most commonly studied MIL datasets. For each dataset, we present number of instances, the min and max number of instances in a bag, number of features, total number of bags with positive and negative labeled information.

Validation. In our approach, we adapt Min-Max normalization. We normalize the training data at each step of the cross validation process. Consequently, we fit each test data using

Table 3 Description of common small to medium-sized MIL datasets

Name	Instances	Min	Max	Features	Bags	+ bags	- bags
Musk 1	476	2	40	166	92	47	45
Musk 2	6598	1	1044	166	102	39	63
Tiger	1220	2	13	230	200	100	100
Fox	1302	1	13	230	200	100	100
Elephant	1391	2	13	230	200	100	100

the scale obtained from the corresponding training set. Other than that, we neither change nor generate any additional features [2]. We use cross validation to compare all the methods in this paper. We perform five replications of ten-fold cross validation (CV) procedures and report the average of key performance indices for different methods using the same randomly generated bootstraps. During the generation of bootstraps, we randomly partition the data over *bags*, i.e., 90% of the bags to be chosen as training set and 10% of the bags to be chosen as test set.

KPIs. In judging the performance of a classifier, we report the area under ROC curve, AUC. As explained before, this is the ultimate metric that addresses many issues and the bias associated with other measures. Thus, we report the average training AUC and test AUC for each method, while we repeat 10-fold cross validation five times. Aside from these, we also compute the widely-discussed average accuracy over test datasets during CV. In order to predict the label of each test bag we first calculate the threshold on training data. In order to compute the threshold for a ranking function, we consider the *highest* scored negative instance from each negative bag and the witness instances from positive bags as representatives. Next, we sort these representatives based on their scores in ascending order. We consider the average score of each pair of consecutive instances in sorted order as a threshold, and calculate the associated accuracy. In this calculation, an instance is misclassified if it has a lower (higher) score than the threshold and is labeled positive (negative). We use the threshold value that maximizes the accuracy, and use it to predict the labels for the test set. If the score of an instance in the test set is above (below) this threshold, we predict that instance is positive (negative). Consequently, if there is at least one positive predicted instance in a bag, that bag is predicted positive, as per the standard assumption.

Used-defined Parameters. As far as the used defined parameters in the optimization model are concerned, we performed minimal preprocessing to compute the scale of the data. We compute the absolute difference between each pair of instances on each dimension and set ϵ to a fraction of the minimum of these values. On the contrary, we set M to a multiple of the sum of each input feature value. We observe that, as expected, the computations are not sensitive to ϵ and M values. For the PSO algorithm, we do not perform a controlled study on the parameters, but use three parameter sets to illustrate the approximate effect of parameters. These parameters, presented in Table 4, are chosen based on the commonly experimented ranges in the literature. For instance, a swarm size of 50 is known to work well in general [34]. Thus, we use a swarm size of 50 in all parameter sets. In line with the literature, we define acceleration coefficients (C1,C2) for the favor of global best in PSO 1, particle best in PSO 2; and kept them equal in PSO 3.

[2] One exception to this, as explained later, is where we add features for nonlinear classification; but we make it explicit and compare against a study that uses the same approach.

Table 4 Parameters used in PSO

	Swarm Size	C1	C2	Epoch Length
PSO 1	50	0.5	1.5	50
PSO 2	50	1.5	0.5	50
PSO 3	50	1.0	1.0	50

Table 5 Results of the three parameter sets for PSO — Average train AUC and average time spent in seconds (in parenthesis), and average test AUC (lower left) and accuracy (upper right) after five repetitions of 10-fold cross validation

Dataset	Train			Test		
	PSO 1	PSO 2	PSO 3	PSO 1	PSO 2	PSO 3
Musk1	78.4%	**82.7**%	78.8%	71.6%	**78.8**%	75.0%
	(40.9)	(74.5)	(54.2)	69.8%	**71.2**%	69.2%
Musk2	76.0%	**80.8**%	75.6%	69.6%	**76.2**%	70.1%
	(50.3)	(89.2)	(56.8)	69.2%	**73.9**%	68.4%
Tiger	79.8%	**84.0**%	80.4%	71.3%	**78.0**%	73.3%
	(93.8)	(186.4)	(114.5)	70.7%	74.8%	**75.0**%
Fox	63.9%	**68.1**%	64.5%	60.9%	**65.8**%	62.3%
	(96.0)	(193.3)	(116.9)	52.7%	51.0%	**55.2**%
Elephant	79.7%	**83.9**%	80.1%	72.0%	**77.5**%	73.3%
	(96.0)	(190.0)	(112.1)	70.2%	**74.8**%	69.5%

Codebase. The source code for our approach and benchmark datasets are made freely available for download at https://github.com/OEKundakcioglu/MI-ROCMax.
Configuration. All computations are performed using Python, calling Gurobi 8.01 [35] to solve optimization problems, on a computer running a Linux operating system with 3.6 GHz Intel i7-7700 quad-core processor and 16 GB DDR4-2400 RAM.

4.1 Linear classification: training AUC and time

From this point forward, the best performances in each comparison table are highlighted in bold, i.e., shortest computation time, largest training AUC, largest testing accuracy, etc.

We start our discussion with the PSO results. Table 5 shows training AUC, which is the objective function of our PSO algorithm, and time it takes to solve an instance on average among the 50 training bootstraps using different PSO parameters. The same table also features test performance of the PSO, where we present test AUC and accuracy averages during five repetitions of 10-fold cross validation.

We observe that even though results are close to each other, PSO 1 is the fastest with poorer solution quality due to early convergence caused by larger magnitude on global best. Likewise, PSO 3 performs poorly, never providing the best result for any dataset except the test AUC on Tiger and Fox. Overall, PSO 2 clearly outperforms other parameters, dominating in all metrics, except for the training time. Thus, without further parameter tuning and without sacrificing the efficiency for performance, we choose PSO 2 to compare against other methods.

We present the average training AUC and average time needed to train the algorithms during CV in Table 6. The algorithms we compare over each bootstrap are kNN, Citation-*k*NN [12], EM-DD [32], MI-SVM, mi-SVM [17], mi-Net, MI-Net, MI-Net with RC [21], attention-based deep-learning [22], our mixed integer linear programming formulation **[eMI-BR]** solved by Gurobi 8.01 [35], and PSO 2.

Table 6 Training Performance — Average training AUC and average time spent in seconds (in parenthesis) for all algorithms after five repetitions of 10-fold cross validation

Dataset	Classical Methods					Deep-Learning				Our Approaches	
	kNN	c-*k*NN	EM-DD	mi-SVM	MI-SVM	mi-Net	MI-Net	MI-NetRC	Attention	eMI-BR	PSO 2
Musk1	89.8%	96.2%	85.3%	88.0%	90.9%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	82.7%
	(5.1)	(14.2)	(9.4)	(75.6)	(30.4)	(12.4)	(12.9)	(12.0)	(13.9)	(6.9)	(74.5)
Musk2	86.2%	92.2%	77.7%	77.8%	92.3%	99.8%	99.%	**100.0%**	99.8%	**100.0%**	80.8%
	(940.4)	(2646.3)	(14.9)	(204.0)	(67.7)	(20.4)	(21.2)	(19.8)	(22.9)	(10.6)	(89.2)
Tiger	88.0%	90.2%	68.0%	92.9%	90.9%	68.9%	52.1%	99.8%	53.2%	**100.0%**	84.0%
	(33.4)	(94.7)	(30.3)	(236.6)	(44.6)	(28.2)	(29.2)	**(27.6)**	(31.7)	(1354.3)	(186.4)
Fox	79.2%	86.4%	55.7%	81.9%	73.2%	80.6%	74.7%	97.4%	75.7	**100.0%**	68.1%
	(38.9)	(110.3)	(29.0)	(439.1)	(60.8)	(28.2)	(29.3)	(27.8)	(31.9)	(5.6)	(193.3)
Elephant	81.7%	92.1%	76.4%	86.6%	89.1%	90.5%	87.3%	**99.3%**	88.2%	95.7%	83.9%
	(43.4)	(123.1)	(32.1)	(528.2)	(42.3)	(28.3)	(29.5)	**(27.9)**	(319)	(6542.2)	(190.0)

There are three key takeaways from Table 6:

- As expected, our exact solution for formulation **[eMI-BR]** provides the best AUC, however, *the margins are noteworthy*. Regardless of the dataset, the best AUC that can be obtained outperforms all other approaches usually by 5-20% with instances up to 35-40%.
- Our exact solution is the fastest on two out of five datasets: Musk2 and Fox. It does not scale up well, as it is the slowest on the complex and large Tiger and Elephant datasets. That is promising for new approaches that can be devised, e.g., preprocessing dataset to a size that can be handled by formulation **[eMI-BR]**. This also shows despite the largest number of binary variables in Musk2, the models are easier to solve due to the relatively low number of bags and feature values for instances.
- PSO, despite lacking a tedious parameter tuning procedure, solves formulation **[eMI-BR]** to a degree that occasionally outperforms some classical methods (e.g., EM-DD in 4 out of 5 instances). It is not better than solving the exact formulation, even slower than exact approach for smaller instances. However, considering how it scales up, the proposed metaheuristic might be useful for larger datasets.

4.2 Linear classification: test AUC and accuracy

Next, we present the test performance for these classifiers. We present average AUC for test set and test accuracy in CV. The test times are not presented because each algorithm takes less than one second to compute AUC or classes of all bags on average.

Several conclusions can be deducted from the results in Table 7. First and foremost, test AUC and accuracy are not closely associated. On the AUC side, despite outperforming all methods in train AUC by a large margin, **[eMI-BR]** does not provide exalted test AUC's. The best test AUC's are provided by three different *deep-learning* methods, and there are relatively large gaps even among those. It should be noted that Musk2 is a dataset that requires a nonlinear classifier, which explains the poor performance of linear classifiers in general.

On test accuracy, one of the key metrics, our exact approach clearly outperforms earlier approaches on Musk1, Tiger, and Elephant datasets, with more than 10% improvement on average. The relatively poor performance on the Fox and Musk2 datasets can be credited to the fast solution time for 100% average AUC on training set, which leads us to a possibly easy separation of classes with a leeway in the ranking vector (*alternative optima* for the bipartite ranking problem) for this data. Together with our exact approach, modern deep-learning approaches seem successful. Test performance of the PSO is even better than its training performance, including an overall best on Fox data. However, our assessment on PSO is still the same as before; a metaheuristic might be useful for larger datasets due to its speed, but for small instances an exact approach is superior.

4.3 Nonlinear classification

Results in Sect. 4.1 show that our exact approach performs successfully especially on those datasets, where instances are known to come from two distributions that are linearly separable. In order to make our approach more flexible, we propose introducing additional features to handle nonlinear classification problems. We still consider the linear ranking function in formulation **[eMI-BR]**, but in a different feature space that accommodates additional features.

For that purpose, we use the approach introduced in [33], where two types of features ($R^{instance}$ and $R^{Cluster}$) are introduced and a solution algorithm is proposed. In $R^{instance}$,

Table 7 Test Performance — Average AUC (top) and accuracy (bottom) for all algorithms after five repetitions of 10-fold cross validation

Dataset	Classical Methods					Deep-Learning				Our Approaches	
	kNN	c-*k*NN	EM-DD	mi-SVM	MI-SVM	mi-Net	MI-Net	MI-NetRC	Attention	[eMI-BR]	PSO 2
Musk1	78.0%	87.8%	80.9%	66.3%	75.1%	95.0%	**96.1%**	94.9%	94.8%	78.6%	71.2%
	78.3%	87.6%	81.2%	49.8%	53.2%	86.8%	88.5%	86.6%	89.0%	**98.9%**	78.8%
Musk2	73.8%	81.7%	77.4%	72.6%	83.1%	91.5%	92.8%	**93.5%**	92.3%	40.1%	73.9%
	68.7%	83.8%	74.0%	65.1%	61.7%	84.5%	84.5%	**85.7%**	84.3%	60.7%	76.2%
Tiger	69.1%	76.4%	66.4%	87.2%	83.3%	62.5%	50.5%	**87.7%**	50.4%	77.2%	74.8%
	69.1%	76.4%	66.4%	56.3%	67.1%	54.4%	50.0%	81.4%	50.3%	**99.5%**	78.0%
Fox	57.6%	60.1%	54.9%	58.3%	54.7%	64.0%	63.3%	58.8%	**64.1%**	55.2%	51.0%
	57.6%	60.1%	54.9%	50.9%	50.1%	58.9%	60.1%	56.4%	60.7%	57.0%	**65.8%**
Elephant	69.7%	80.3%	76.6%	78.1%	87.5%	93.1%	79.5%	**93.8%**	81.4%	83.2%	74.8%
	69.7%	80.3%	76.6%	45.8%	77.6%	57.9%	51.1%	86.2%	53.1%	**98.1%**	77.5%

new features are added for each instance according to their dissimilarities with all other instances. In $R^{Cluster}$, instances are clustered with k-means clustering algorithm, and new features are added for each instance according to their dissimilarities with cluster centers. We fix an issue with the number of clusters (k) in the k-means clustering algorithm. In each replication, Kucukasci et al. [33] perform ten-fold cross validation, however, their algorithm finds k only during the first training, which is used for the rest of the cross validation procedure. In that case, the number of clusters might become larger than the number of instances in a subsequent training set and the algorithm terminates without a solution. We address this issue using two alternative approaches: In what we call $R_1^{Cluster}$, we use their algorithm in each fold of each replication and make the number of clusters specific to each fold. With this method it is not possible to have k that is greater than the number of instances. Alternatively, in what we call $R_2^{Cluster}$, we follow their steps and find k only in the first fold and use it in each of the remaining folds within a replication, if possible. If the number of instances is less than k, which means the original k cannot be used, in any fold, we reduce k only for those folds and continue clustering.

We introduce additional features based on $R^{instance}$ and $R^{Cluster}$ as explained, and solve the problem using our exact and heuristic algorithms. We also include the results of the proposed solution approach in [33]. Comparing these results with earlier results on **[eMI-BR]** using original features, we aim to see the contribution of new features and our solution approach.

Average AUC training results and training times for all solution algorithms are reported in Table 8. As far as the PSO parameters are concerned, we only report PSO 2 results here, as they perform relatively better. The results for all parameters are presented in Appendix B. First thing to notice in this table is that the training performance of the model proposed in [33] is not as good as our proposed formulations for $R^{Cluster}$. Judging the contribution of these new features is rather difficult on the training results, because **[eMI-BR]** can already perform perfect ranking even when using the original features only. Thus, there is no significant change to observe after the addition of new features, except a 4.3% increase in AUC and a speed-up in the training of challenging Tiger and Elephant datasets.

In the more important test performance results, given in Table 9, adding new features drastically improves the test AUC and accuracy of **[eMI-BR]** on Musk2. This is expected, as nonlinear classifiers are known to perform better on this dataset. There are marginal improvements on most of the datasets when new features are added, especially using $R^{instance}$. In terms of accuracy, **[eMI-BR]** performs better than the methods in [33] in three out of five instances (Musk1, Tiger, and Elephant), regardless of data representations. On the other two instances, Musk2 and Fox, despite 100% training AUC, accuracy is relatively lower. That clearly illustrates the existence of alternative optima, especially in the new feature space. It worthwhile to note that, PSO 2, while performing subpar in training and testing AUC, performs the best in Fox dataset in terms of accuracy.

Fig. 1 presents the performance profile chart that shows how errors of all methods compare, which is the percentage of mislabeled bags during cross validation. We do not present $\boldsymbol{R}^{Cluster}$ variants as $\boldsymbol{R}^{instance}$ performs similar, with slightly better and more robust performance. This chart shows that both of our exact approaches (i.e., **[eMI-BR]** and **[eMI-BR]** RInstance) outperform other methods in the literature. Furthermore, it can be seen from the performance profile chart that *all methods except our exact approaches have errors 10 times the best method's error, for 3 or 4 out of 5 datasets.*

In the light of these numerical experiments, we have two key observations: (i) maximizing training AUC in the new feature space generally provides high-quality testing accuracy, (ii) there is an alternative optima issue that needs to be addressed with Pareto efficiency, rather

Table 8 Training Performance — Average training AUC and average time spent in seconds (in parenthesis) for the algorithm in [33] against the same feature set solved with our exact approach and PSO 2 after five repetitions of 10-fold cross validation

Dataset	$R^{instance}$	Our Approaches		$R_1^{Cluster}$	Our Approaches		$R_2^{Cluster}$	Our Approaches	
		[eMI-BR] $R^{instance}$	PSO 2 $R^{instance}$		[eMI-BR] $R_1^{Cluster}$	PSO 2 $R_1^{Cluster}$		[eMI-BR] $R_2^{Cluster}$	PSO 2 $R_2^{Cluster}$
Musk1	**100.0%**	**100.0%**	89.3%	**100.0%**	**100.0%**	89.9%	**100.0%**	**100.0%**	89.3%
	(0.4)	(6.0)	(70.4)	**(2.9)**	(5.4)	(69.7)	**(0.9)**	(5.5)	(69.2)
Musk2	**100.0%**	**100.0%**	87.7%	97.9%	**100.0%**	87.5%	97.8%	**100.0%**	87.4%
	(176.7)	**(11.4)**	(532.6)	(19.0)	**(10.6)**	(88.5)	**(5.7)**	(10.5)	(90.7)
Tiger	**100.0%**	**100.0%**	85.7%	96.9%	**100.0%**	85.8%	96.7%	**100.0%**	85.5%
	(2.7)	(147.7)	(213.0)	**(4.8)**	(152.5)	(190.6)	**(1.5)**	(153.0)	(184.1)
Fox	**100.0%**	**100.0%**	66.6%	92.9%	**100.0%**	66.9%	92.8%	**100.0%**	66.1%
	(3.2)	(5.8)	(215.2)	**(5.6)**	(5.7)	(199.1)	**(1.5)**	(5.7)	(187.9)
Elephant	**100.0%**	**100.0%**	87.9%	96.0%	**100.0%**	88.3%	95.8%	**100.0%**	87.9%
	(3.7)	(223.7)	(204.2)	**(4.4)**	(201.8)	(182.2)	**(1.4)**	(120.9)	(181.9)

Table 9 Test Performance — Average AUC (top) and accuracy (bottom) for the algorithm in [33] against the same feature set solved with our exact approach and PSO 2 after five repetitions of 10-fold cross validation

Dataset	$R^{instance}$	Our Approaches		$R_1^{Cluster}$	Our Approaches		$R_2^{Cluster}$	Our Approaches	
		[eMI-BR] $R^{instance}$	PSO 2 $R^{instance}$		[eMI-BR] $R_1^{Cluster}$	PSO 2 $R_1^{Cluster}$		[eMI-BR] $R_2^{Cluster}$	PSO 2 $R_2^{Cluster}$
Musk1	**95.3**%	89.2%	85.1%	**95.3**%	61.2%	83.5%	**95.8**%	68.0%	82.9%
	88.8%	**98.9**%	81.4%	87.1%	**98.8**%	82.6%	87.8%	**98.9**%	82.1%
Musk2	**95.3**%	84.6%	86.2%	**93.4**%	72.6%	83.6%	**93.2**%	72.6%	83.0%
	84.0%	75.4%	79.6%	**82.4**%	75.6%	79.8%	**83.3**%	75.4%	79.6%
Tiger	**87.2**%	80.8%	80.4%	**84.7**%	62.9%	81.4%	**84.6**%	61.1%	79.1%
	81.9%	**99.5**%	79.9%	78.7%	**99.5**%	79.5%	78.2%	**99.5**%	79.0%
Fox	**68.8**%	50.5%	58.3%	**62.2**%	50.4%	57.5%	**63.8**%	50.4%	60.8%
	63.4%	55.5%	**64.3**%	59.1%	54.8%	**64.6**%	58.9%	54.8%	**63.6**%
Elephant	**92.1**%	86.1%	84.9%	**87.8**%	71.9%	83.5%	**87.8**%	68.6%	83.7%
	86.8%	**99.4**%	80.9%	84.5%	**99.5**%	81.1%	84.3%	**99.5**%	80.6%

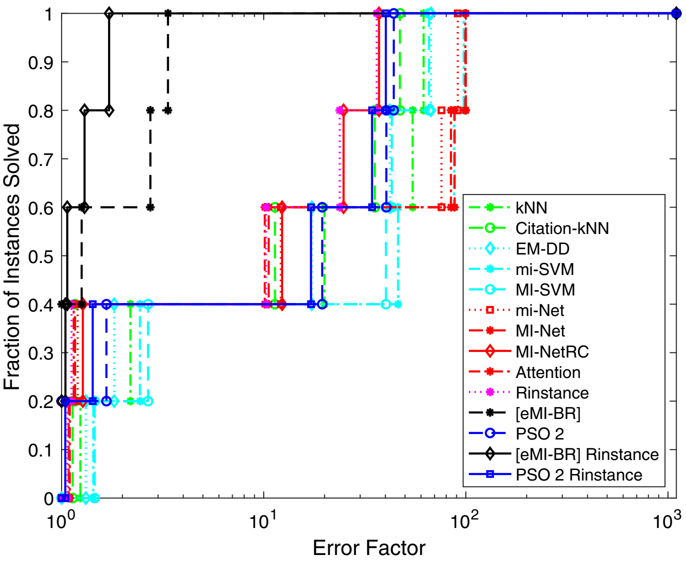

Fig. 1 Performance profile chart for the error of all methods after five repetitions of 10-fold cross validation

Table 10 Description of common medium to large-sized MIL datasets

Name	Instances	Min	Max	Features	Bags	+ bags	- bags
Mutagenesis 1	10486	28	88	7	188	125	63
Mutagenesis 2	2132	26	86	7	42	13	29
Corel, Antique	7947	2	13	9	2000	100	1900
Corel, Battleships	7947	2	13	9	2000	100	1900
Corel, Beach	7947	2	13	9	2000	100	1900

than a haphazard selection by the optimizer. Among a pool of solutions with maximal AUC, there are some solutions that has better generalization performance for small datasets. A good example to this is the dramatically different testing performance of several solutions (all with 100% training AUC) for Musk 1 in Table 9. This is mainly due to the standard assumption, which leads to a large selection of feasible witnesses with different ranking alternatives. Another reason is the nature of the dataset, where additional features contribute further to alternative classifications of classes, leading to possible overfitting with an even larger set of alternative optimal solutions.

4.4 Results on challenging instances

We finally perform numerical experiments on datasets that are larger or harder to classify. We conjecture the alternative optimal solutions would not be as common in these challenging instances. Therefore, the results in this section reveal the ultimate performance of each approach. Table 10 shows the larger datasets that we study.

We do not introduce additional features presented in Sect. 4.3, and only perform scaling in the preprocessing phase. Solving large datasets to optimality is computationally challenging;

Table 11 Overall Performance — Testing AUC (top), testing accuracy (middle), training time in seconds (bottom) for all algorithms after five repetitions of 10-fold cross validation

Dataset	Classical Methods					Deep-Learning				Our Approach
	kNN	c-*k*NN	EM-DD	mi-SVM	MI-SVM	mi-Net	MI-Net	MI-NetRC	Attention	PSO 2
Mutagenesis1	62.3%	73.9%	50.0%	53.3%	55.4%	79.8%	78.4%	**80.4%**	78.8%	76.4%
	71.5%	77.2%	66.4%	66.4%	33.5%	66.4%	66.4%	71.3%	66.4%	**78.8%**
	(1955.3)	(5492.0)	(18.5)	(32.8)	(15.2)	**(13.0)**	(63.6)	(55.2)	(69.4)	(169.3)
Mutagenesis2	65.5%	62.8%	50.0%	67,3%	**70.5%**	66.3%	67.8%	67.8%	67.1%	53.5%
	66.7%	74.3%	69.5%	69.5%	69.5%	69.5%	69.5%	77.2%	69.5%	**80.7%**
	(81.1)	(227.9)	(3.1)	(5.2)	**(2.8)**	(59.4)	(13.9)	(12.1)	(15.2)	(26.3)
CorelAntique	59.2%	52.0%	50.0%	78.2%	72.8%	85.7%	**89.6%**	89.5%	89.2%	80.8%
	89.4%	94.4%	**95.0%**	**95.0%**	**95.0%**	**95.0%**	**95.0%**	**95.0%**	**95.0%**	94.9%
	(1210.4)	(3837.4)	**(214.7)**	(588.9)	(404.2)	(493.3)	(509.5)	(480.1)	(559.4)	(1987.4)
CorelBattleships	65.2%	54.8%	50.0%	73.2%	73.0%	86.5%	93.4%	**93.9%**	**93.9%**	72.4%
	90.1%	94.8%	**95.0%**	**95.0%**	**95.0%**	**95.0%**	**95.0%**	**95.0%**	**95.0%**	94.9%
	(1205.8)	(3829.8)	**(215.3)**	(240.5)	(363.5)	(494.7)	(510.4)	(482.5)	(560.0)	(2094.8)
CorelBeach	79.1%	71.5%	56.2%	95.5%	97.0%	96.5%	**97.5%**	97.4%	97.4%	95.9%
	94.5%	96.2%	95.2%	95.0%	95.0%	95.0%	95.0%	95.0%	95.0%	**96.3%**
	(1208.5)	(3830.7)	(205.9)	(588.9)	**(175.1)**	(492.0)	(511.2)	(481.0)	(558.9)	(2132.9)

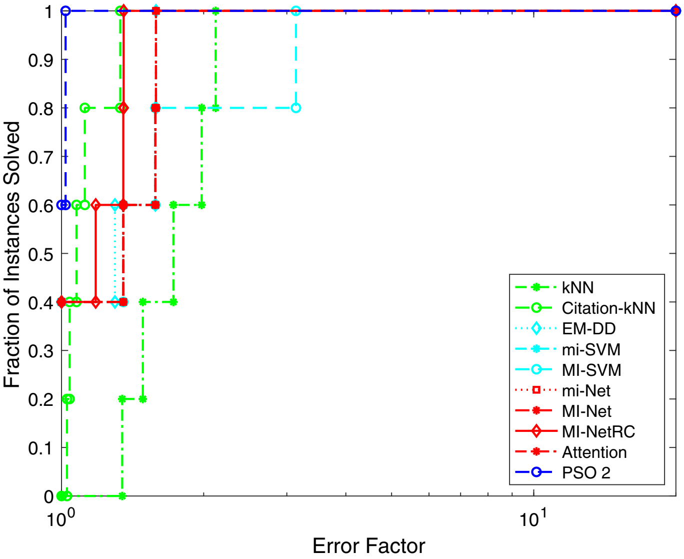

Fig. 2 Performance profile chart for the error of all methods after five repetitions of 10-fold cross validation for the challenging instances

therefore, we exclude **[eMI-BR]** and inevitably use PSO 2 as an alternative to our exact approach. The results are summarized in Table 11.

Table 11 confirms that test AUC and accuracy are not closely related. For instance, even though EM-DD provides $> 90\%$ accuracy in three Corel datasets, the AUC level is $< 60\%$. Citation-kNN performs well in small datasets, yet it takes the longest time to terminate, and it never provides the best AUC nor accuracy result on large datasets. In general, modern approaches seem to provide better solution quality than classical methods reasonably fast. Despite being a nonexact/linear classification approach with simple parameter tuning procedure, PSO yields the best accuracy in three out of five datasets and off by 0.1% in the remaining two.

Fig. 2 shows the success of PSO 2 on a performance profile chart. Citation-kNN might be considered the second best, where a majority of instances are solved with a small error factor of less than 1. The best among deep-learning methods is MI-NetRC that yields an error factor of 1.35 for 60% of the instances.

5 Concluding remarks

In this study we develop a multi-instance binary classification approach by directly maximizing the area under ROC curve. Our mixed integer linear programming model solves the bipartite ranking problem to produce the best possible linear ranking hyperplane for multi-instance data. Recent advances in the commercial optimization software make our formulation practical for medium-sized benchmark datasets. We provide cross validation results against well-known approaches on these instances and shed light on the potential of optimization and hyperplane-based approaches.

We observe that having the best ranking hyperplane for training data generally provides the best accuracy. This remarkable accuracy difference between our exact approach and state-of-the-art approaches clearly demonstrates the potential of solution methods that optimize the Wilcoxon-Mann-Whitney statistic. Even suboptimal linear ranking functions through heuristic approaches in large datasets prove success against classical and modern approaches on benchmark instances. One potential future research direction is development of practical and more powerful algorithms to solve the mathematical optimization problem in larger datasets. It would be interesting to see both exact methods that utilize decomposition or reformulation and more powerful near-optimal methods. From a data mining standpoint, well-thought tie-breakers for alternative optima cases that potentially cause *overfitting* and incorporating nonlinear classification (possible with Kernel trick) are issues that need to be addressed in the future as well.

Acknowledgements The authors would like to thank Gizem Atasoy, who while preferring not to contribute to this paper as a co-author, experimented with the initial version of our mathematical model with some data instances during her MSc Thesis study. The authors are also grateful to Nima Manafzadeh Dizbin, who helped with the implementation of deep-learning methods that are used for benchmarking.

Appendix

A Flowchart of the particle swarm optimization (PSO) algorithm

See Appendix Fig. 3.

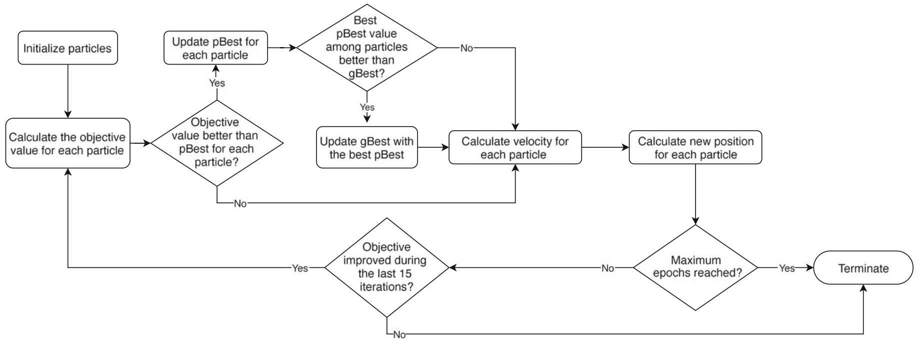
Initialize particles
Update pBest for each particle
Best pBest value among particles better than gBest?
No
Yes
Yes
Calculate the objective value for each particle
Objective value better than pBest for each particle?
Update gBest with the best pBest
Calculate velocity for each particle
Calculate new position for each particle
No
Objective improved during the last 15 iterations?
Yes
No
Maximum epochs reached?
Yes
Terminate
No

Fig. 3 Flowchart of the Particle Swarm Optimization (PSO) algorithm

B Training and test performance for PSO with different parameters

See Appendix Tables 12 and 13.

Table 12 Training Performance — Average training AUC and average time spent in seconds (in parenthesis) for PSO using three parameter sets after five repetitions of 10-fold cross validation

Dataset	$R^{instance}$			$R_1^{Cluster}$			$R_2^{Cluster}$		
	PSO 1	PSO 2	PSO 3	PSO 1	PSO 2	PSO 3	PSO 1	PSO 2	PSO 3
Musk1	86.4%	**89.3%**	86.9%	86.4%	**89.9%**	85.6%	86.0%	**89.3%**	86.5%
	(42.9)	(70.4)	(48.3)	**(40.6)**	(69.7)	(45.1)	**(42.7)**	(69.2)	(52.0)
Musk2	83.9%	**87.7%**	85.7%	84.3%	**87.5%**	84.1%	83.7%	**87.4%**	84.4%
	(326.2)	(532.6)	(376.4)	**(52.1)**	(88.5)	(63.6)	**(52.4)**	(90.7)	(64.5)
Tiger	81.5%	**85.7%**	81.7%	82.2%	**85.8%**	82.7%	81.4%	**85.5%**	82.6%
	(102.6)	(213.0)	(118.8)	**(92.7)**	(190.6)	(103.6)	**(96.0)**	(184.1)	(117.5)
Fox	64.6%	**66.6%**	64.5%	63.1%	**66.9%**	63.8%	63.2%	**66.1%**	63.9%
	(103.0)	(215.2)	(123.3)	**(95.1)**	(199.1)	(112.2)	**(95.9)**	(187.9)	(117.0)
Elephant	85.3%	**87.9%**	85.7%	85.8%	**88.3%**	85.3%	84.8%	**87.9%**	85.7%
	(105.7)	(204.2)	(126.6)	**(94.8)**	(182.2)	(113.7)	**(95.1)**	(181.9)	(112.0)

Table 13 Test Performance — Average AUC (top) and accuracy (bottom) for PSO using three parameter sets after five repetitions of 10-fold cross validation

Dataset	$R^{instance}$			$R_1^{Cluster}$			$R_2^{Cluster}$		
	PSO 1	PSO 2	PSO 3	PSO 1	PSO 2	PSO 3	PSO 1	PSO 2	PSO 3
Musk1	79.9%	**85.1%**	79.9%	78.0%	**83.5%**	77.0%	76.7%	**82.9%**	82.1%
	76.8%	**81.4%**	77.9%	76.4%	**82.6%**	76.5%	76.6%	**82.1%**	78.2%
Musk2	77.8%	**86.2%**	80.2%	79.7%	**83.6%**	78.8%	74.9%	**83.0%**	81.4%
	74.2%	**79.6%**	76.7%	74.3%	**79.8%**	76.1%	73.3%	**79.6%**	76.0%
Tiger	70.7%	**80.4%**	71.0%	72.6%	**81.4%**	72.6%	75.8%	**79.1%**	74.1%
	71.3%	**79.9%**	73.0%	72.9%	**79.5%**	74.9%	72.2%	**79.0%**	75.1%
Fox	58.3%	58.3%	**58.6%**	57.4%	**57.5%**	56.9%	56.8%	**60.8%**	58.2%
	61.0%	**64.3%**	62.0%	60.3%	**64.6%**	61.0%	60.5%	**63.6%**	61.3%
Elephant	80.2%	**84.9%**	79.8%	79.8%	**83.5%**	79.9%	78.2%	**83.7%**	83.0%
	76.6%	**80.9%**	77.8%	75.9%	**81.1%**	76.8%	76.0%	**80.6%**	77.7%

References

1. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning, 1st edn. The MIT Press (2010)
2. Jordan, M.I., Mitchell, T.M.: Machine learning: Trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015)

3. Mannino, M., Yang, Y., Ryu, Y.: Classification algorithm sensitivity to training data with non representative attribute noise. Decision Support Systems **46**(3), 743–751 (2009)
4. Foulds, J., Frank, E.: A review of multi-instance learning assumptions. The Knowledge Engineering Review **25**(1), 1–25 (2010)
5. Vanwinckelen, G., Fierens, D., Blockeel, H., et al.: Instance-level accuracy versus bag-level accuracy in multi-instance learning. Data Mining and Knowledge Discovery **30**(2), 313–341 (2016)
6. Amores, J.: Multiple instance classification: Review, taxonomy and comparative study. Artificial Intelligence **201**, 81–105 (2013)
7. Xu, X.: Statistical learning in multiple instance problems. Master's thesis, The University of Waikato, (2003)
8. Carbonneau, M.-A., Cheplygina, V., Granger, E., Gagnon, G.: Multiple instance learning: A survey of problem characteristics and applications. Pattern Recognition **77**, 329–353 (2018)
9. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In Proceedings of the 14th European Conference on Machine Learning, ECML'03, pp 468–479, Berlin, Heidelberg. Springer-Verlag. (2003)
10. Zhou, Z.-H.: Multi-instance learning?: A survey. Technical report, AI Lab, Department of Computer Science & Technology, Nanjing University, Nanjing, China (2004)
11. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10, NIPS '97, pp 570–576, Cambridge, MA, USA. MIT Press. (1998)
12. Wang, J., Zucker, J.-D.: Solving multiple-instance problem: A lazy learning approach. In Proceedings of the 17th International Conference on Machine Learning, pp 1119—1125. Morgan Kaufmann, (2000)
13. Zucker, J.-D., Chevaleyre, Y.: Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. application to the mutagenesis problem. In Proceedings of the 14th Canadian Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, Ottawa, Canada, pp 204–214, (2000)
14. Zhou, Z.-H., Zhang, M.-L.: Neural networks for multi-instance learning. In Proceedings of the International Conference on Intelligent Information Technology, Beijing, China, pp 455–459, (2002)
15. Babenko, Boris: Multiple instance learning?: Algorithms and applications. Technical report, Department of Computer Science and Engineering. University of California, San Diego, USA (2008)
16. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence **89**(1–2), 31–71 (1997)
17. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems 15, pp. 577–584. MIT Press (2003)
18. Erhun Kundakcioglu, O., Seref, O., Pardalos, P.M.: Multiple instance learning via margin maximization. Applied Numerical Mathematics **60**(4), 358–369 (2010)
19. Poursaeidi, M.H., Erhun Kundakcioglu, O.: Robust support vector machines for multiple instance learning. Annals of Operations Research **216**(1), 205–227 (2014)
20. Carbonneau, M.-A., Granger, E., Raymond, A.J., Gagnon, G.: Robust multiple-instance learning ensembles using random subspace instance selection. Pattern Recognition **58**, 83–99 (2016)
21. Wang, X., Yan, Y., Tang, P., Bai, X., Liu, W.: Revisiting multiple instance neural networks. Pattern Recognition **74**, 15–24 (2018)
22. Ilse, M., Tomczak, J., Welling, M.: Attention-based deep multiple instance learning. In International conference on machine learning, pp 2127–2136. PMLR, (2018)
23. Bertsimas, D., Chang, A., Rudin, C.: A discrete optimization approach to supervised ranking. In Proceedings of the 5th INFORMS Workshop on Data Mining and Health Informatics (DM-HI 2010), (2010)
24. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**(8), 861–874 (2006)
25. Fawcett, T.: Prie: a system for generating rulelists to maximize roc performance. Data Mining and Knowledge Discovery **17**(2), 207–224 (2008)
26. Yan, L., Dodier, R.H., Mozer, M., Wolniewicz, R. H.: Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp 848–855, (2003)
27. Krishna Menon, A., Williamson, R.C.: Bipartite ranking: A risk-theoretic perspective. The Journal of Machine Learning Research **17**(1), 6766–6867 (2016)
28. Green, D.M., Swets, J.A.: Signal Detection Theory and Psychophysics. Wiley, New York (1966)
29. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology **143**(1), 29–36 (1982)
30. Cortes, C., Mohri, M.: AUC optimization vs. error rate minimization. Advances in Neural Information Processing Systems **16**, 313–320 (2004)

31. Eberhart, R.C., Shi, Y., Kennedy, J.: Swarm Intelligence. Elsevier (2001)
32. Zhang, Q., Goldman, S.A.: EM-DD: An improved multiple-instance learning technique. In Advances in Neural Information Processing Systems **14**, 1073–1080 (2002)
33. Kucukasci, E. S., Baydogan, M. G., Taskin, Z. C.: A linear programming approach to multiple instance learning. Turkish Journal of Electrical Engineering & Computer Sciences, 1–16, (2021). https://doi.org/10.3906/elk-2009-144
34. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm intelligence **1**(1), 33–57 (2007)
35. Gurobi Optimization. Gurobi optimizer reference manual, (2020). URL http://www.gurobi.com

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.