Accepted for publication in Knowledge and Information Systems

DOI: 10.1007/s10115-013-0711-1

MILDE: Multiple Instance Learning by Discriminative Embedding

Jaume Amores

Computer Vision Center, Universitat Autònoma de Barcelona, Spain

Abstract. While the objective of the standard supervised learning problem is to classify feature vectors, in the Multiple Instance Learning problem the objective is to classify *bags*, where each bag contains multiple feature vectors. This represents a generalization of the standard problem, and this generalization becomes necessary in many real applications such as drug activity prediction, content-based image retrieval and others. While the existing paradigms are based on learning the discriminant information *either* at the instance level *or* at the bag level, we propose to incorporate both levels of information. This is done by defining a discriminative embedding of the original space based on the responses of cluster-adapted instance classifiers. Results clearly show the advantage of the proposed method over the state-of-the-art, where we tested the performance through a variety of well-known databases that come from real problems, and we also included an analysis of the performance using synthetically generated data.

Keywords: Multi-Instance Learning ; Codebook ; Bag of Words

1. Introduction

In Multiple Instance Learning (MIL) we have a training set where individual feature vectors have not an associated class label. Instead, class labels are associated with so-called *bags*, which are collections of feature vectors. This type of problem, which is a generalization of the standard learning one, can be found in many areas such as drug activity prediction (Dietterich et al, 1997), bankruptcy prediction (Kotsiantis and Kanellopoulos, 2008), content-based image retrieval (Chen

Received Mar 15, 2013
Revised Jul 30, 2013
Accepted Oct 18, 2013
The original publication is available at `http://www.springerlink.com`
Link to the article: `http://link.springer.com/article/10.1007/s10115-013-0711-1`

et al, 2006), speaker recognition (Reynolds et al, 2000), and others. In our analysis we focus on the the usual binary classification setting (i.e., bags are classified as either positive or negative), and use the standard one-against-all strategy for expressing multi-class problems as a series binary classifications.

In this work we analyze the methods existing in the literature and categorize them into two big paradigms: the traditional one is based on learning instance-level models, while the embedded-based paradigm is based on learning models at the bag-level. Based on this analysis, we propose a novel method, MILDE (Multiple Instance Learning by Discriminative Embedding), which incorporates the strengths from both paradigms and thus increases the accuracy of recent approaches. Let us explain very briefly the two paradigms and the idea under the proposed MILDE method.

The traditional MIL paradigm has its origins in the early work of Dietterich et al. (Dietterich et al, 1997), and has been followed by many authors (Oded, 1998; Zhang and Goldman, 2001; Andrews et al, 2003; Xu and Frank, 2004; Gehler and Chapelle, 2007; Antic and Ommer, 2013). This paradigm assumes that the individual instances can be classified into just two classes, positive and negative. Furthermore, they assume a given relationship between the labels of the instances and those of the bags they belong to. Given this assumption, the methods learn an instance-level classifier, and the classification score of the whole bag is determined from the (binary) classification scores of the instances.

This type of paradigm has the advantage that it offers a mechanism for learning the characteristics of individual instances in a discriminative way, where remember that we only have access to a training set of labelled bags. The disadvantage of this paradigm is two fold. First, the assumption that there are only two classes of instances inside the bags does not necessarily hold in practice. It might well happen that a positive bag is characterized by containing several classes of instances. A second drawback of this paradigm is the fact that the discriminative learning process occurs at the instance level, and not at the bag level. This is due to the fact that this paradigm assumes that by obtaining a discriminative model of the instances we are able to classify the whole bag, based on the binary scores of the instances. This prevents the method to learn information beyond the individual instances, i.e., at the bag level. For example, it might happen that positive bags are characterized by having a certain combination of classes of instances. In this case, the same classes of instances can be found in both positive and negative bags, and the difference lies in the specific combination (e.g, positive bags contain instances of class 1 *and* 2, while negative bags contain instances of class 1 *or* 2, but not both classes at the same time). This type of situation occurs frequently and will be illustrated along the paper. In any case, there is certain type of information that can only be discovered if we look at the whole bag, i.e., at the combination of instances lying inside, and not only at the characteristics of the individual instances.

More recently there have been several authors (Chen et al, 2006; Scott et al, 2005; Zhou and Zhang, 2007; Gärtner et al, 2002) that propose MIL methods able to learn bag-level information in a discriminative way. These methods learn discriminative models about the whole bag, instead of learning models about the positive instances. In one way or the other, all these methods perform an embedding from the original bag space into a new vector space. This embedding can be performed implicitly through a SVM-based kernel mapping (Gärtner et al, 2002) (see also the review (Foulds and Frank, 2010)), or explicitly (Chen et al, 2006; Scott et al, 2005; Zhou and Zhang, 2007). In the latter case, the methods

extract a feature vector for each bag, where the vector summarizes relevant information about the whole bag. The resulting feature vectors are fed into a standard learning algorithm that obtains a model. This way, the resulting model considers information at the bag level, as it is fed with vectors that summarize bag-level characteristics.

While the methods from the embedded-based paradigm learn bag-level information, they lose the capability of learning the characteristics of the individual instances. In this work we propose a new MIL method that incorporates the strengths of both paradigms: it learns bag-level information and instance-level information, both in a discriminative way. Furthermore, we obtain several instance-level models, where each one is adapted to a particular region of the instance space, hence discriminating the instances into more than two classes, positive and negative, in contrast with the traditional methods. In order to achieve these objectives, we propose an embedded-based method where each bag is represented by a single feature vector. This feature vector is extracted in such a way that it measures the matching degree between several instance-level models and the instances of the bag. This in turn provides relevant bag-level information such as the combination of classes of instances that is characteristic of positive bags, which can be learned by feeding the obtained feature vectors into a standard learning algorithm.

The rest of the paper is organized as follows. In section 2 we introduce some basic concepts and terminology, and characterize the traditional and embedded-based paradigms. In section 3 we introduce MILDE. In section 4 we describe the experimental set-up and in section 5 we present quantitative results. Finally, section 6 presents the conclusions and summarizes the main contributions.

2. Background

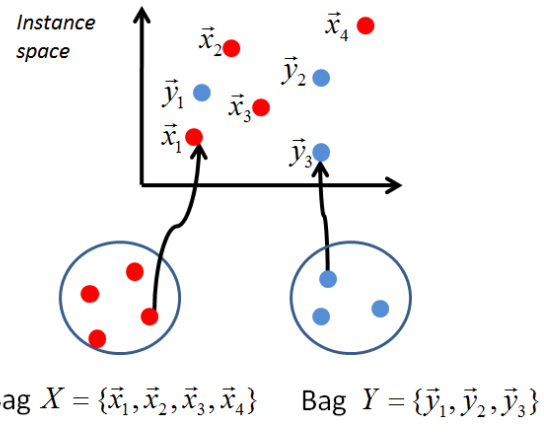
We introduce some basic notation in section 2.1 and then we characterize the traditional and embedded-based paradigms in section 2.2. In order to illustrate how these paradigms work, in section 2.3 we describe two common MIL problems. Finally, in section 2.4 we review methods from the literature that fall into each paradigm.

2.1. Basic concepts and notation

A bag (or multi-set) is mathematically defined as a collection of elements $\vec{x}_1, \ldots, \vec{x}_N$ where there might be repetitions. In this work, however, we denote a bag X as a set $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$, following a notation that is usual in the literature. This notation is used because of its simplicity and due to the fact that the possible existence of repetitions does not really affect the analysis of the methods [1].

The individual elements of the bag $\vec{x}_j \in X$ are called instances, and all of them live in the same *instance space* $\mathcal{I}$. In practice, $\mathcal{I} = \mathbb{R}^d$, i.e., the instances are defined as d-dimensional real feature vectors. The cardinality of the bag N varies from bag to bag, i.e., different bags might have different numbers of

[1] Furthermore, in practice, the repetition of elements occurs very rarely. This is due to the fact that the elements $\vec{x}_j$ are almost always defined as vectors of real components, i.e., $\vec{x}_j \in \mathbb{R}^d$ for $j = 1, \ldots, N$.



[htb]

Fig. 1. Illustration of bags and instances.

instances. Fig. 1 illustrates a synthetic MIL example where there are two bags X and Y. The bag X contains four instances $X = \{\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4\}$, depicted as red points, and the bag Y contains three instances $Y = \{\vec{y}_1, \vec{y}_2, \vec{y}_3\}$, depicted as blue points. In this example, the instance space is $\mathbb{R}^2$, i.e., the instances $\vec{x}_j$ and $\vec{y}_j$ are bi-dimensional vectors.

Given the above definitions, the objective of the MIL problem is to obtain a classifier of bags, given a training set of labelled bags. In this work we focus on binary classification, where we only have a positive and a negative class [2]. The objective is to obtain a classification function $F(X) \in [0, 1]$ that provides the classification score for bag X: the higher the value $F(X)$, the higher the confidence that X belongs to the positive class. In order to *learn* the classifier $F(X)$, we are given a training set $\mathcal{T}$ that contains M bags X_i and their corresponding labels L_i:

$$\mathcal{T} = \{(X_1, L_1), \ldots, (X_M, L_M)\}, \tag{1}$$

where $L_i = 1$ if X_i is positive, and $L_i = 0$ otherwise.

In addition to the bag-level classification function $F(X)$, many methods try to learn an instance-level classification function $f(\vec{x}_j)$ that operates directly on the instances $\vec{x}_j \in X$. As we will see, the instance classifier $f(\vec{x}_j)$ is learned in a discriminative way by building a training set of instances

$$\tau = \{(\vec{x}_1, l_1), \ldots, (\vec{x}_S, l_S)\}, \tag{2}$$

Note that this training set is not given as input, and thus it must be obtained by the MIL method, by estimating the (hidden) values of the instance labels $l_j \in \{0, 1\}$, for $j = 1, \ldots, S$. In order to estimate these values, the methods usually rely on some assumption about the relationship of the bag labels L_i and the hidden instance labels l_j.

Throughout this work we will use uppercase to refer to bags X_i, their labels L_i and the bag-level classifier F, and we will use lowercase to refer to instances

[2] Given a multi-class problem, it can be reduced to several binary classification problems by means of common strategies such as one-against-all

$\vec{x}_j$, the estimated labels l_j (if they are used at all, depending on the method) and the instance-level classifier f (if it is used, depending on the method).

2.2. Characterization of MIL paradigms

We describe here the governing equations and steps followed by the two main MIL paradigms of the literature. Later, in sections 2.4.1 and 2.4.2, we discuss briefly some methods from the literature that belong to each of these two paradigms.

2.2.1. Traditional paradigm

The traditional paradigm is based on the so-called *standard MI* assumption (Foulds and Frank, 2010). This states that the positive bags are characterized by containing *at least one* instance that belongs to a so-called positive class, whereas the negative bags are characterized by not containing any instance that belongs to this positive class. Given this assumption, the methods first try to learn an instance-level classifier $f(\vec{x}) \in [0, 1]$ that determines whether an instance $\vec{x}$ is positive ($f(\vec{x}) > 0.5$) or not ($f(\vec{x}) < 0.5$). Once this instance-level classifier $f(\vec{x})$ has been estimated, the bag-level classifier $F(X)$ can be calculated by applying the max-rule:

$$F(X) = \max_{\vec{x} \in X} f(\vec{x}) \tag{3}$$

This assures that the bag X is classified as positive ($F(X) > 0.5$) if and only if at least one instance $\vec{x} \in X$ is positive ($f(\vec{x}) > 0.5$).

In particular, the methods of this paradigm follow three steps:

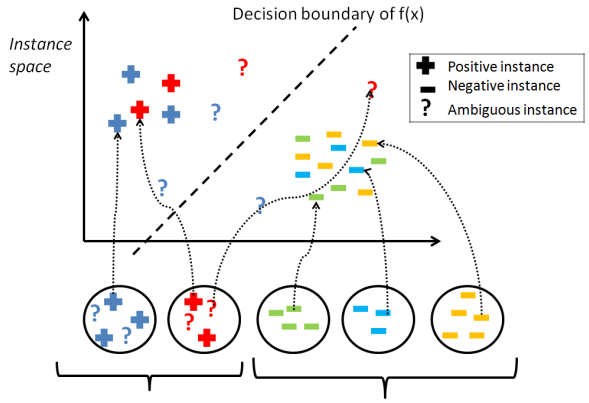
1. Learn an instance-level classifier $f(\vec{x})$. The methods learn this classifier by using constraints about the relationship between the labels of the instances and the labels of the bag. Given a new instance $\vec{x} \in X$, the function $f(\vec{x})$ provides a classification score by considering only the information in $\vec{x}$, i.e., without looking at the rest of the bag X.
2. Calculate the bag-level classifier $F(X)$ as an aggregation of instance-level scores:

$$F(X) = \frac{f(\vec{x}_1) \circ f(\vec{x}_2) \circ \ldots \circ f(\vec{x}_N)}{Z}, \tag{4}$$

where $\circ$ denotes the aggregation operator (for example the sum, the product, or the maximum) and Z is an optional normalization factor. In practice, the max-rule in Eq. 3 is the most usual aggregation.

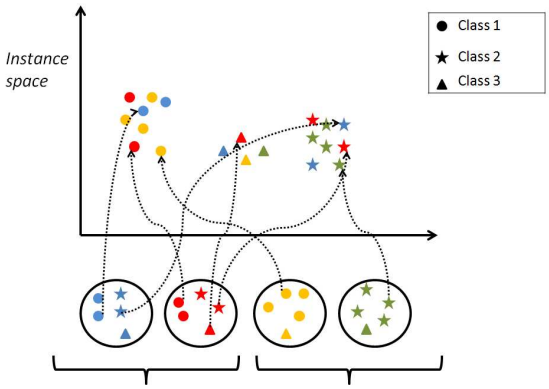
The most important characteristic of this paradigm is that the learning is produced only at the instance level, in the step 1. At the bag-level, there is no learning, and the bag classification $F(X)$ is based on a simple aggregation of instance-level classification scores. This is the most important difference with the embedded-based paradigm (see below) where the learning is produced at the bag level.

This type of approach is reasonable for problems such as the drug activity prediction, discussed in section 2.3.1, where the standard MI assumption might hold (at least to some extent). In Fig. 2 we illustrate the idea under the standard MI assumption and the solution obtained by this paradigm.



[htb]

Fig. 2. Standard MI assumption (best seen in color).



[htb]

Fig. 3. Example of MIL problem where bag-level information becomes necessary (best seen in color).

In Fig. 3 we illustrate an example of MIL problem where this paradigm cannot succeed. Here, the positive bags are characterized by having instances of class 1 *and* class 2 *at the same time*, while the negative bags might contain instances of either class 1 *or* class 2, but not both of them at the same time. Instances of class 3 are just ambiguous instances present in both positive and negative bags. In section 2.3.2 we describe a real problem where something similar happens.

In this type of problem, there is not a single class of instances that characterizes the positive bags. For example, if we take the class of instances 1 as "positive", then Eq. 3 will give a high classification score for both positive and negative bags. The same happens if we consider class 2 as positive or if we consider the union of class 1 and 2. The only solution for this problem is to learn the *combination* of instances that makes a bag positive. This is done in the embedding-based paradigm that is briefly characterized below.

2.2.2. Embedded-based paradigm

The methods of this paradigm follow the steps:

1. Define a mapping function $\mathcal{M}(X) = \vec{v} \in \mathbb{R}^K$ that maps the bag X to a feature vector $\vec{v} \in \mathbb{R}^K$. This feature vector summarizes the content of the bag X using K features.
2. Map the bags of the training set $\mathcal{T}$ (defined in Eq. 1) so that we obtain a new training set in the embedded space:

$$\mathcal{V} = \{(\vec{v}_1, L_1), \ldots, (\vec{v}_M, L_M)\}, \tag{5}$$

   where $v_i = \mathcal{M}(X_i)$ for $i = 1, \ldots, M$
3. Train a standard classifier $\mathcal{G} : \mathbb{R}^K \mapsto [0, 1]$ using the training set $\mathcal{V}$. This can be any classifier such as SVM, AdaBoost, or others.
4. Given a new bag X to be classified, define the bag-level classifier $F(X)$ as:

$$F(X) = \mathcal{G}(\mathcal{M}(X)), \tag{6}$$

   i.e., $F(X)$ provides the score of the standard classifier $\mathcal{G}$ applied to the feature vector $\vec{v} = \mathcal{M}(X)$, where this feature vector summarizes the content of the bag.

The vectors $\vec{v}_i$ convey information about the combination of instances that appear in each bag X_i, and this information is learned by the standard classifier $\mathcal{G}$. For example, let us consider a mapping $\mathcal{M}(X) = \vec{v}$ that provides vectors $\vec{v} = (v_1, \ldots, v_K)$ where the i-th component v_i measures the confidence that the bag X contains instances of class i. If we use such a mapping with the problem of Fig. 3, the learning algorithm will infer that the positive bags contain high values in both the first and second components, v_1 and v_2, indicating that positive bags contain both classes of instances, while negative bags will only contain a high value in either v_1 or v_2. Although this is just an ideal example, in practice the mapping $\mathcal{M}$ will usually convey some information (at least in an approximate way) about the classes of instances that are present in the bag, and therefore the learned classifier $\mathcal{G}$ is based on this type of information.

Let us describe two real MIL examples in order to clarify the concepts.

2.3. Examples of MIL problems

2.3.1. Drug activity prediction

The first example is the classical drug activity prediction problem studied in the early work of Dietterich et al. (Dietterich et al, 1997). In this problem, the task is to predict whether or not a given molecule is a valid drug. Each molecule can adopt several three dimensional shapes (called *configurations*) by rotating its internal bonds. The molecule is a valid drug if at least one of its configurations binds to a specific target site, called *binding site*, which is usually a hole or a cavity in a bigger molecule (e.g., a protein).

In order to describe the molecule, a common technique is to describe each one of its possible configurations, where the j-th configuration is described by means

of a feature vector $\vec{x}_j$. This way, the molecule is described by means of a bag X that is a collection of feature vectors (instances) $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$, where the number of instances (i.e., the number of configurations) can vary across molecules.

This type of problem has motivated the so-called *standard MI assumption* (Foulds and Frank, 2010), that states that a bag X is positive (i.e., the corresponding molecule is a valid drug) if *at least one* of its instances is positive (i.e., at least one of its configurations binds to the target binding site). Based on this assumption, the methods of the traditional paradigm are based on estimating an instance-level classifier $f(\vec{x})$ that provides a positive score for instances $\vec{x}$ corresponding to configurations that bind to the target site. Given this instance classifier, the bag classifier $F(X)$ is calculated using an aggregation rule such as the logical-or or the maximum of the scores $f(\vec{x}_j)$.

The difficulty with such an approach is that we do not have access to the labels of the individual instances, i.e., we are not able to determine what configurations of a molecule bind to the binding site.

An additional disadvantage of this paradigm is that it disregards a lot of information contained in the bags. For example, a positive bag might contain, in addition to the configuration that binds to the target binding site, other configurations whose features are also characteristic of positive bags and provide useful information, despite not binding to the target site. In other words, by considering only one class of instances (the one that binds to the target binding site), we are disregarding the information of other classes of instances that are characteristic of positive bags (i.e., instances that appear frequently in positive bags and that, consequently, provide useful information). Thus, a method that exploits the information from the whole bag, in terms of the presence of multiple classes of instances, might be more accurate.

2.3.2. Content-based image classification

Another real example of MIL problem is content-based image classification. Fig. 4 shows an example where the objective is to classify images as either beach (top row) or non-beach (bottom row), i.e., where the positive class is formed by images that contain a beach and the negative class is formed by images that contain other types of visual content, such as images of sea and images of desert (in the bottom row of Fig. 4), but also images of mountain, city, food, etc.

If we look at the images in Fig. 4, we can see that each one depicts several objects (e.g., trees, people, sky, sand, sea, mountain, etc.), where the meaning of the red circles is explained below. Some of the objects appearing in the images, such as the sea and the sand, are related with the target class ("beach"), while other objects (e.g., the trees, the mountain, the sky, etc.) are not specific of the class and thus are not relevant. Taking this into account, the usual approach is to describe each region of the image separately, in order to avoid mixing up the features of relevant regions (sea and sand) with the ones of irrelevant regions. As a result, the image is described by a set of feature vectors, i.e., a bag $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$, where the j-th feature vector $\vec{x}_j$ describes the j-th region of the image. This is illustrated in Fig. 4, where each red circle symbolizes one of the regions where we extract a feature vector. The number of regions considered depends on the specific algorithm used for detecting regions of interest (Nowak et al, 2006; Mikolajczyk et al, 2005), and this number might vary across images.

Regarding the feature vectors $\vec{x}_j$, they usually describe characteristics such as the R,G,B color and texture of the region.

In this problem, the individual instances could be manually labelled according to the semantic class of the region that the instance describes. However, manually labelling every region in every image is a very time consuming task, impractical if we have a large repository of images. As a consequence, this problem is many times expressed as a MIL problem, where we have bags with unlabelled instances.

In this type of problem it is not enough to learn a model about the class of the individual instances (i.e., the class of the regions that an image might contain: "tree", "sky", "sand", "sea", etc.). This is due to the fact that there is not a single class of instances that explains the class of the whole bag. Instead, it is the global *composition* of the bag, i.e., the simultaneous presence of several classes of instances inside the bag, what determines the class of the bag. For example, in the example illustrated in Fig. 4, neither the presence of sand instances or sea instances alone explains the fact that the image depicts a beach. Instead, it is the simultaneous presence of both classes of instances what determines the existence of the beach.

For this type of problem, the paradigm of embedded-based methods described in section 2.4.2 is especially appropriate. This is due to the fact that these methods represent each bag by a feature vector that summarizes the global composition of the bag. These feature vectors are then introduced into a standard learner that, based on this information, is able to extract a model about the combination of instances that makes a bag positive.

We must note that in the example of Fig. 4 the characteristic combination is an "and" of two classes (the presence of sand instances *and* sea instances). However, there might be other cases (in image classification and in other fields) where the characteristic combination is an "*or*" of two classes (e.g., the presence of class 3 or class 4), and still other cases where the characteristic combination is a more complex combination that involves 'and" and "or" operators and involves several classes. The problem is that we do not know a priori what is the combination that is informative. The problem here is that we must perform some learning at the bag level in order to obtain a model about what classes of instances are discriminative, and what combination of these instances is characteristic. This is precisely what the embedded-based paradigm attempts to do.

2.4. Methods in the literature

Let us describe the methods of the literature that fall into the traditional and the embedded-based paradigm, which were introduced in section 2.2. We restrict the discussion to the methods that are more closely related to our proposal.

2.4.1. Traditional paradigm

As we explained in section 2.2.1, the main characteristic of this paradigm is that the discriminative learning process occurs *at the instance level*. As a result, the methods from this paradigm learn an instance classifier $f(\vec{x})$ and they compute the bag level classification $F(X)$ as an aggregation of instance scores, see Eqs. 4 and 3.

One of the methods that fall into this paradigm is probably the first MIL

[htb]

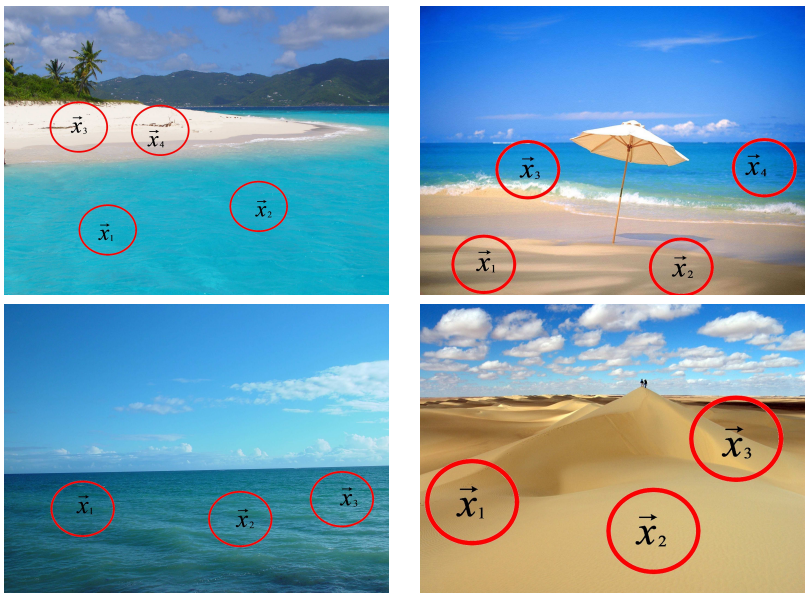


Fig. 4. Classification of images into beach (top row) and non beach (bottom row).

method published in literature, in the classical paper of Dietterich et al. (Dietterich et al, 1997). This method is called the Axis-Parallel Rectangles (APR) algorithm.

In the APR algorithm, the instance level classifier $f(\vec{x})$ is obtained by estimating an axis parallel rectangle $\mathcal{R}$. Based on this rectangle, $f(\vec{x})$ classifies as positive those instances that fall inside, and it classifies as negative those instances that fall outside. Therefore, the rectangle $\mathcal{R}$ represents a discriminative model of the positive versus the negative instances. It is obtained as part of a learning process that makes use of the standard MI assumption, and where the objective is to maximize the number of positive bags that have *at least one* instance inside the rectangle, and the number of negative bags that *do not have any* instance inside. After learning the instance-level classifier $f(\vec{x})$, the bag-level classifier $F(X)$ is constructed as an aggregation of instance scores, using the max-rule in Eq. 3, or the logical-or rule $F(X) = f(\vec{x}_1) \vee f(\vec{x}_2) \vee \ldots \vee f(\vec{x}_N)$ (note that both aggregation rules are equivalent in this case, as the classifier provides a binary output).

Inspired by the APR method, Maron et al. (Maron and Lozano-Pérez, 1998) proposed the Diverse Density method. The idea is similar to the previous one, but instead of estimating a rectangle the method estimates the Gaussian that maximizes a so-called Diverse Density. This is a continuous measure that considers the probability that *at least one* instance inside each positive bag belongs to the Gaussian and, at the same time, the probability that there is no instance from negative bags that belongs to the Gaussian. In particular, the Gaussian is parameterized by $(\vec{t}, \vec{s})$, where $\vec{t} \in \mathbb{R}^d$ is the center of the Gaussian and $\vec{s} \in \mathbb{R}^d$ represents the size of the Gaussian along each one of the d dimensions of the instance space $\mathbb{R}^d$. Based on these parameters, the probability that an instance

$\vec{x}$ belongs to the Gaussian is approximated as:

$$f(\vec{x};\vec{t},\vec{s}) = \exp[-\sum_{i=1}^{d}\left(\frac{\vec{x}-t_i}{s_i}\right)^2] \tag{7}$$

This can also be expressed in a more compact form as:

$$f(\vec{x};\vec{t},\Sigma) = \exp\left(-(\vec{x}-\vec{t})^T\Sigma^{-1}(\vec{x}-\vec{t})\right), \tag{8}$$

where Σ is a diagonal matrix derived from $\vec{s} = (s_1,\ldots,s_d)$ as follows:

$$\Sigma_{ij} = \begin{cases} s_i^2 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Once the Gaussian has been estimated, the bag-level classifier $F(X)$ is calculated by using the max-rule in Eq. 3 over the scores provided by the instance classifier in Eq. 7.

An evolved version of the latter method is the Expectation-Maximization Diverse Density (EM-DD) (Zhang and Goldman, 2001). This algorithm is based on expectation-maximization and provides an efficient mechanism for maximizing the DD measure. We use this algorithm as part of our method MILDE, explained below. Many other methods follow the same paradigm based on instance-level classifiers. Some well-known examples are the MI-SVM method (Andrews et al, 2003) and the method of (Bunescu and Mooney, 2007), which make use of bag-level constraints related with the Standard MI assumption in order to learn an instance-level classifier $f(\vec{x})$. Also, more recently Antic and Ommer (Antic and Ommer, 2013) propose to slit the training set into multiple sub-sets and learn an independent instance-level classifier $f(\vec{x})$ for each one. Theses independent classifiers are then used to robustly label the instances in the training set by aggregating multiple classifier decisions over each instance. Finally, a single instance-level classifier $f(\vec{x})$ is learned based on the previously labelled instances of the training set. At the end, all these methods obtain a model of the individual instances, and the bag-level classification is based on aggregating the instance-level scores (e.g., using the max-rule in Eq. 3) as mentioned in section 2.2.1.

2.4.2. Embedded-based paradigm

In this paradigm, the discriminant decision is taken at the bag level. For this purpose, all the information from the bag X is extracted and expressed as a single feature vector $\vec{v} = \mathcal{M}(X)$ which summarizes the relevant aspects of X. This is done for every bag of the training set and the resulting feature vectors $\vec{v}_1,\ldots,\vec{v}_M$ are introduced into a discriminant learner $\mathcal{G}$ that obtains a bag-level model using this information. Finally, given a new bag X, the bag-level classifier is expressed as $F(X) = \mathcal{G}(\mathcal{M}(X))$, as explained in section 2.2.2.

A number of well-known methods follow this paradigm (Chen et al, 2006; Gärtner et al, 2002; Scott et al, 2005; Zhou et al, 2009; Zhang and Zhou, 2009; Zhou and Zhang, 2007), including those that can be seen as performing implicit mappings as discussed in (Foulds and Frank, 2010). Here we review briefly the MILES method in (Chen et al, 2006) which is closely related to MILDE.

The MILES method gathers all the *positive* instances (i.e., all the instances belonging to the positive bags of the training set), obtaining a large pool of instances: $\mathcal{P} = \{\vec{t}_1,\ldots,\vec{t}_R\}$, where $\vec{t}_j$ is the j-th positive instance and R is the

total number of positive instances in the training set. Each one of the instances $\vec{t}_j \in \mathcal{P}$ act as a prototype. Given these prototypes, the MILES method maps each bag X into a feature vector $\mathcal{M}(X) = (v_1, \ldots, v_R)$ where the j-th component v_j measures the similarity between the bag X and the j-th prototype $\vec{t}_j$ as follows:

$$v_j = \max_{\vec{x} \in X} \exp\left(-\frac{\|\vec{x} - \vec{t}_j\|^2}{\sigma^2}\right), \tag{10}$$

i.e., the j-th component v_j provides the maximum similarity between the instances in X and the j-th prototype $\vec{t}_j$ of the pool $\mathcal{P}$. This similarity function has a Gaussian form with center $\vec{t}_j$ and scale σ^2, and can be interpreted as the probability that at least one instance of X belongs to the Gaussian class parameterized by $(\vec{t}_j, \sigma^2)$. This expression was derived in (Chen et al, 2006) based on the Diverse Density framework, where $\vec{t}_j$ is seen as a candidate *concept* representing one of the classes of positive instances, and it is approximated by a Gaussian.

Given the mapping $\mathcal{M}(X) = \vec{v}$ computed with Eq. 10, the MILES method transforms each bag X_i of the training set into a feature vector $\mathcal{M}(X_i) = \vec{v}_i$. As a result, the training set of bags $\mathcal{T} = \{(X_1, L_1), \ldots, (X_M, L_M)\}$ is mapped to a training set of feature vectors $\mathcal{V} = \{(\vec{v}_1, L_1), \ldots, (\vec{v}_M, L_M)\}$. This can be introduced into any standard learning algorithm in order to obtain a discriminant classifier $\mathcal{G}(\vec{v}) \mapsto [0, 1]$. At test time, given a bag X, MILES maps it to $\mathcal{M}(X) = \vec{v}$ and classifies it using $\mathcal{G}(\vec{v})$.

Although we can use any standard learning algorithm, the MILES method makes use of linear SVM in order to obtain a classifier $\mathcal{G}(\vec{v}) = \vec{w} \cdot \vec{v} + b$, where $\vec{w}$ and b are the weight and the bias parameters defining the classifier. These parameters are optimized by minimizing the objective function:

$$R_{emp}(\vec{w}, b) + \lambda \|\vec{w}\|,$$

where the first term is the empirical error, and the second is the regularization term (Chen et al, 2006). In the case of the MILES algorithm, the authors propose to use a L1 norm $\|\vec{w}\| = \sum_j |w_j|$. This forces the weight vector $\vec{w}$ to be sparse, which can be exploited for selecting only the few components that are non-zero. Given the fact that each component of the vector $\vec{w}$ is associated with one instance of $\mathcal{P}$ (see Eq. 10), learning this type of classifier allows to obtain instance selection. The authors call this type of classifier a 1-norm linear SVM (also known as L1-regularized linear SVM), which can be found in standard SVM lybraries such as (Fan et al, 2008), along with the more common L2-regularized solution.

We must note that, in general, the mapping function $\mathcal{M}(X)$ and the standard classifier $\mathcal{G}(\vec{v})$ are two separate components. In this sense, different standard classifiers $\mathcal{G}(\vec{v})$ can be used, and in this work we tested both non-linear SVM and 1-norm linear SVM. If we use the latter, however, we can avoid computing all the components of the vector, so that we obtain a more efficient mapping function $\mathcal{M}'(X) = (v_{i_1}, \ldots, v_{i_{R'}})$, where $i_1, \ldots, i_{R'}$ are the components selected.

3. Proposed method

As discussed in the previous section, the embedded-based approaches allow to take into account the whole content of the bag in the learning process. On the

other hand, the traditional MIL methods obtain a discriminative model of the individual instances.

In this work we propose a method, MILDE (Multiple Instance Learning by Discriminative Embedding), that incorporates the strengths of both paradigms. For this purpose, MILDE makes use of an embedded-based system built upon the discriminative responses of K instance-level models.

Although there are some important clarifications that we discuss below, the basic steps used by MILDE are as follows. First, the method identifies K classes of instances $\mathcal{C}_1, \ldots, \mathcal{C}_K$ that are present in positive bags. In practice, this is done by unsupervised clustering. Given the k-th class of instances $\mathcal{C}_k$, MILDE learns a *discriminative* instance-level classifier $h_k(\vec{x}) \in [0, 1]$. Based on this, the k-th bag-level classification function $H_k(X)$ is defined as:

$$H_k(X) = \max_{\vec{x} \in X} h_k(\vec{x}), \tag{11}$$

that provides the confidence that at least one instance in the bag X belongs to the class $\mathcal{C}_k$.

Based on the resulting classifiers $H_1, \ldots, H_K$, the method defines the following mapping function:

$$\begin{array}{rl} \mathcal{M}(X) & = \vec{v} \in \mathbb{R}^K \\ \vec{v} & = \left(H_1(X), H_2(X), \ldots, H_K(X)\right), \end{array} \tag{12}$$

By using such a mapping, the bag X is represented by one feature vector $\vec{v}$ that indicates the classes of instances that are present in the bag, providing the confidence for each one of the classes. Finally, the bag-level classifier is defined as $F(X) = \mathcal{G}(\mathcal{M}(X))$, where $\mathcal{G}$ is a standard supervised classifier that has been learned using the feature vectors $\vec{v}_i$ for each bag X_i of the training set (see section 2.2.2).

The classifier $h_k(\vec{x})$ is defined using Eq. 7, based on a Gaussian model with parameters $\Theta_k = (\vec{t}_k, \vec{s}_k)$. This Gaussian model is learned using the EM-DD algorithm, described in section 3.1, in such a way that the center of the Gaussian $\vec{t}_k$ is estimated as a point of the instance space that is close to the positive instances in $\mathcal{C}_k$ and, at the same time, is far away from the instances of all the negative bags.

The resulting method can be seen as a generalization of the MILES method (Chen et al, 2006) if we plug Eq. 8 in Eq. 11, so that we obtain:

$$v_k = H_k(X) = \max_{\vec{x} \in X} e^{-(\vec{x}-\vec{t}_k)^T \Sigma_k^{-1} (\vec{x}-\vec{t}_k)}, \tag{13}$$

where Σ_k is derived from $\vec{s}_k$ using Eq. 9.

We can see in Eq. 13 that the k-th component v_k indicates the best matching between the instances in X and the k-th model $\Theta_k = (\vec{t}_k, \vec{s}_k)$. In contrast, in the MILES method (Eq. 10), the k-th component v_k indicates the best matching between the instances in X and the k-th reference point $\vec{t}_k$. The fundamental difference between MILDE and MILES is that the Gaussian models are obtained here as a result of a discriminative learning process. In contrast, in the MILES algorithm the reference points $\vec{t}_k, k = 1, \ldots, R$ are the raw instances from the training set (i.e., each instance of the training set gives rise to an isotropic Gaussian that is centered on this instance, and all the Gaussian models have the same scale s). Using a set of *learned* Gaussian models (as opposed to pre-fixed

Gaussian models centered at the original instances of the training set) makes the resulting mapping more discriminative, which provides a higher accuracy.

There is an important clarification to be done, regarding the instance level classifiers $h_k(\vec{x})$. In our implementation, this classifier is learned in such a way that it discriminates between *all* the positive instances and *all* the negative instances in the training set, and at the same time it is adapted to the cluster $\mathcal{C}_k$. Note that this is different from discriminating between instances in $\mathcal{C}_k$ and instances in the rest of the classes. This is more clearly explained in section 3.1.

Before explaining the implementation of the method, let us emphasize three important points. First, MILDE is based on discriminative learning both at the bag level, by using an embedded-based approach, and at the instance level, by learning the instance classifiers Θ_k. Second, if we use Gaussian models $\Theta_k = (\vec{t}_k, \vec{s}_k)$, then the method generalizes MILES, as we discussed before. However, we can use other non-Gaussian models in order to obtain Θ_k. In this case, MILDE is no longer a generalization of MILES. Third, the Gaussian models $\Theta_k = (\vec{t}_k, \vec{s}_k)$ are learned using a *discriminative* method. This means that they discriminate between instances from positive bags and instances from negative ones. At the same time, the Gaussian model Θ_k is adapted to the k-th cluster of instances $\mathcal{C}_k$. In section 3.3 we further discuss the differences between MILDE and both the EM-DD and the MILES methods.

3.1. Implementation of MILDE

Let us start by describing a simplified version of the implementation, and in section 3.1.2 we explain the complete implementation. Let $\mathcal{P}$ be a set that gathers the instances from all the positive bags in the training set. The proposed MILDE method starts by estimating the K classes of instances that are present in $\mathcal{P}$. For this purpose, the instances in $\mathcal{P}$ are clustered into K clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ with K-means, where the cluster $\mathcal{C}_k$ represents the k-th class of instances.

For each cluster $\mathcal{C}_k$ the method learns an instance classifier $h_k(\vec{x})$ using the EM-DD algorithm, explained in section 3.1.1. EM-DD estimates a Gaussian model with parameters $(\vec{t}, \vec{s})$, where $\vec{t}$ and $\vec{s}$ represent the center and the size of the Gaussian respectively. For this purpose, the EM-DD algorithm starts with an initialization $(\vec{t}_0, \vec{s}_0)$ of the Gaussian model. In our case, we use the center of the cluster $\mathcal{C}_k$ (i.e., the mean of the instances inside $\mathcal{C}_k$) as initial center of the Gaussian $\vec{t}_0$. Using this initialization, the EM-DD algorithm uses a gradient-descent algorithm that iteratively moves the initial point $\vec{t}_0$ towards a local minimum of the energy field. Given the definition of energy field used by EM-DD (discussed in section 3.1.1), the gradient-descent can be seen as the result of applying two forces on the initial point $\vec{t}_0$: the first force makes it move towards positive instances of the training set, and the second force makes it move far away from negative instances. As a result, the center of the Gaussian $\vec{t}$ is the result of moving the center of the cluster $\mathcal{C}_k$ towards a nearby region of the space that is densely populated with positive instances and free of negative ones.

Let us now describe the EM-DD algorithm, and in section 3.1.2 we describe the rest of the algorithm (i.e., how the cluster-adapted classifiers H_k are obtained from h_k, and how the final bag-level classifier $F(X)$ is computed).

Algorithm for function $(\vec{t}, \vec{s}) \leftarrow EM\_DD(\mathcal{T}, \vec{t}_0, \sigma)$

- **Input:** Training set $\mathcal{T} = \{(X_1, L_1), \ldots, (X_M, L_M)\}$ with bags X_i and their corresponding labels L_i. Initial target point in instance space $\vec{t}_0 \in \mathbb{R}^d$, and initial scale σ.
- **Output:** Learned parameters: target point $\vec{t}$ and vector of scales $\vec{s}$.
- **Notation:** We denote $\Theta = (\vec{t}, \vec{s})$ the current hypothesis, Ψ the space of all possible hypothesis, and $\mathcal{C}_\Theta$ the Gaussian defined by Θ.

Method:

1. Initialize current hypothesis $\Theta \leftarrow (\vec{t}_0, \vec{s}_0)$, where $\vec{s}_0 = (\sigma, \ldots, \sigma) \in \mathbb{R}^d$ i.e., a vector whose d elements are set to σ.
2. for $n = 1, \ldots, niter$ do:
   - 2.1 Initialize $\mathcal{Z} \leftarrow \emptyset$
   - 2.2 **Expectation Step:**
     for each bag $X_i \in \mathcal{T}$ do:
     - 2.2.1 $\vec{z}_i \leftarrow \arg\max_{\vec{x} \in X_i} Pr(\vec{x} \in \mathcal{C}_\Theta)$
     - 2.2.2 $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(\vec{z}_i, L_i)\}$
   - 2.3 **Maximization Step:**

$$\Theta' \leftarrow \arg\min_{\Theta \in \Psi} Energy(\Theta, \mathcal{Z}), \quad \text{where:}$$

$$Energy(\Theta, \mathcal{Z}) = -\sum_{(\vec{z}_i, L_i) \in \mathcal{Z}} \log Pr(L_i | \Theta, \vec{z}_i) \tag{14}$$

   - 2.4 $\Theta \leftarrow \Theta'$
3. Return the parameters that conform the final hypothesis $\Theta = (\vec{t}, \vec{s})$.

Fig. 5. EM-DD algorithm

3.1.1. EM-DD algorithm

The EM-DD algorithm is shown in Fig. 5. A detailed explanation about this algorithm can be found in (Zhang and Goldman, 2001). Here we only introduce the equations used in this algorithm and provide a rough interpretation about the underlying idea. Afterwards, we provide a synthetic example of how EM-DD works.

Let us first introduce the notation used in Fig. 5. In line 2.1. of the algorithm, we denote as $Pr(\vec{x} \in \mathcal{C}_\Theta)$ the probability that the instance $\vec{x}$ belongs to the Gaussian defined by $\Theta = (\vec{t}, \vec{s})$. This probability is estimated as follows:

$$Pr(\vec{x} \in \mathcal{C}_\Theta) = e^{-\left((\vec{x}-\vec{t})^T \Sigma^{-1} (\vec{x}-\vec{t})\right)}, \tag{15}$$

where Σ is a diagonal matrix derived from the parameter $\vec{s}$ of the Gaussian (see Eq. 9).

In the E-step (line 2.2), the algorithm obtains a training set $\mathcal{Z}$ with M pairs $(\vec{z}_i, L_i) \in \mathcal{Z}$, where M is the number of bags, $\vec{z}_i$ is the instance that best represents the i-th bag X_i according to the Gaussian model Θ, and L_i is the label of the bag.

The M-step (line 2.3) updates the Gaussian Θ using the training set $\mathcal{Z}$ gathered in the E-step. This is done by using gradient-descent in order to find the hypothesis Θ that minimizes an energy function $Energy(\Theta, \mathcal{Z})$. This functional is a sum of log-likelihoods (see Eq. 14), where the likelihood $Pr(L_i|\Theta, \vec{z}_i)$ is defined as:

$$Pr(L_i|\Theta, \vec{z}_i) = \begin{cases} Pr(\vec{z}_i \in \mathcal{C}_\Theta) & \text{if } L_i = 1 \\ 1 - Pr(\vec{z}_i \in \mathcal{C}_\Theta) & \text{if } L_i = 0 \end{cases} \tag{16}$$

Based on these definitions, we can see that the rough idea under the EM-DD algorithm is the following. In the E-step, the algorithm selects the instances that lie closest to the current center of the Gaussian $\vec{t}$, by using a Mahalanobis distance that takes into account the scale $\vec{s}$ along each dimension (see Eq. 15). In the M-step, the algorithm updates the Gaussian Θ by moving it towards the instances $\vec{z}_i$ that are positive ($L_i = 1$) and far away from the instances that are negative ($L_i = 0$). This can be seen if we plug Eq. 16 into the definition of the energy functional (Eq. 14).

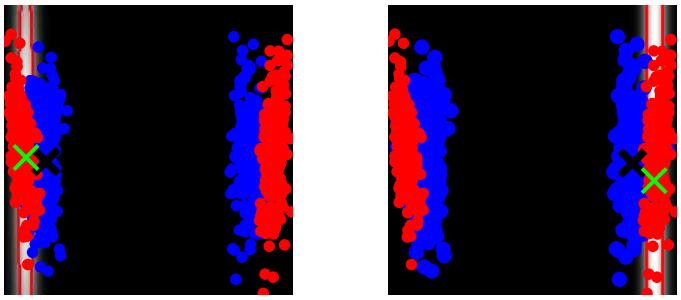
Let us now see how this algorithm works using a synthetic example. For this purpose, let us first look at Fig. 6(a). This figure shows instances $\vec{x} \in \mathbb{R}^2$, where the red points are instances that belong to positive bags, and the blue points are instances that belong to negative bags. The black cross symbolizes the initial center of the Gaussian $\vec{t}_0$ introduced as parameter to EM-DD. The green cross symbolizes the center $\vec{t}$ of the Gaussian obtained at the end. We can see that the initial center $\vec{t}_0$ lies in a region where there are many negative instances (blue points), while the final center $\vec{t}$ has moved to a region where there are positive instances (red points). The figure also shows the probability map of the estimated Gaussian: the higher the intensity of white the higher the probability $Pr(\vec{x} \in \mathcal{C}_\Theta)$. We can see that the estimated Gaussian fits one of the two regions of positive instances. In this sense we can say that the estimated model specializes to one of the clusters of positive instances. In Fig. 6(b) we show another estimated model, this time using as initialization a black cross that lies on the right side of the instance space. As we can see, the resulting model specializes to the other cluster of positive instances. Thus, by using two different initializations we obtain two different models, one for each cluster of the instance space.

3.1.2. Complete algorithm

In practice, we use a multi-scale approach by initializing the EM-DD model with T possible scales, and thus obtaining KT resulting classifiers, where K is the number of clusters and T the number of scales. Fig. 7 shows the resulting algorithm for obtaining these KT classifiers $H_1(X), \ldots, H_{KT}(X)$. These classifiers are obtained by repeatedly calling the EM-DD function in Fig. 5, each time with a different initialization.

Based on the algorithm of Fig. 7, the final algorithm for obtaining the bag-level classifier $F(X)$ is shown in Fig. 8. As we can see, the algorithm estimates $2KT$ cluster-adapted classifiers $H_1, \ldots, H_{2KT}$. The first KT classifiers $H_1, \ldots, H_{KT}$ are adapted to each one of the K clusters of instances found in *positive* bags. The last KT classifiers $H_{KT+1}, \ldots, H_{2KT}$ are adapted to each one

[htb]



(a) (b)

Fig. 6. Two Gaussian models estimated by EM-DD using 2D synthetic data.

Algorithm for function $\{H_1, \ldots, H_{KT}\} \leftarrow Obtain\_Classifiers(\mathcal{T}, K, \mathcal{S})$

– **Input:** Training set $\mathcal{T} = \{(X_1, L_1), \ldots, (X_M, L_M)\}$ with bags X_i and their corresponding labels L_i. Number of clusters K to be obtained, and initial scales $\mathcal{S} = \{\sigma_1, \ldots, \sigma_T\}$.
– **Output:** KT cluster-adapted classifiers $\{H_1, \ldots, H_{KT}\}$

Method:

– Let $\mathcal{P}$ be a set that gathers the instances from all the positive bags of the training set.
– Cluster the instances in $\mathcal{P}$ using K-means, obtaining K clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$.
– Let $\vec{p}_k$ be the center of cluster $\mathcal{C}_k$, for $k = 1, \ldots, K$, i.e.,

$$\vec{p}_k \leftarrow \frac{1}{|\mathcal{C}_k|} \sum_{\vec{x} \in \mathcal{C}_k} \vec{x}$$

– Set $j \leftarrow 1$
– For $k = 1, \ldots, K$ and $t = 1, \ldots T$ do:
  · $(\vec{t}_j, \vec{s}_j) \leftarrow EM\_DD(\mathcal{T}, \vec{p}_k, \sigma_t)$
  · Define the j-th instance classifier h_j as $h_j(\vec{x}) = \exp\left(-(\vec{x} - \vec{t}_j)^T \Sigma_j^{-1} (\vec{x} - \vec{t}_j)\right)$, where Σ_j is a diagonal matrix that is derived from $\vec{s}_j$ using Eq. 9.
  · Define the j-th cluster-adapted bag classifier as $H_j(X) = \max_{\vec{x} \in X} h_j(\vec{x})$
  · $j \leftarrow j + 1$
– Return the KT cluster-adapted bag classifiers $\{H_1, \ldots, H_{KT}\}$.

Fig. 7. Intermediate function of the proposed method, used for obtaining the cluster-adapted discriminant classifiers.

Algorithm for function $F(X) \leftarrow Obtain\_Bag\_Classifier(\mathcal{T}, K, \mathcal{S})$

- **Input:** Training set $\mathcal{T} = \{(X_1, L_1), \ldots, (X_M, L_M)\}$ with bags X_i and their corresponding labels L_i. Number of clusters K to be obtained, and initial scales $\mathcal{S} = \{\sigma_1, \ldots, \sigma_T\}$.
- **Output:** Bag-level classifier $F(X)$

Method:

- $\{H_1, \ldots, H_{KT}\} \leftarrow Obtain\_Classifiers(\mathcal{T}, K, \mathcal{S})$
- Let $\mathcal{T}'$ be a training set where the bag labels L'_i have a value opposite to the original: $L'_i = |1 - L_i|$.
- $\{H_{KT+1}, \ldots, H_{2KT}\} \leftarrow Obtain\_Classifiers(\mathcal{T}', K, \mathcal{S})$
- Define the mapping function $\mathcal{M}(X) = \vec{v} \in \mathbb{R}^{2KT}$ as:

$$\mathcal{M}(X) = (H_1(X), \ldots H_{2KT}(X))$$

- Let $\mathcal{V}$ be the training set in the resulting embedding space, i.e., $\mathcal{V} = \{(\vec{v}_1, L_1), \ldots, (\vec{v}_M, L_M)\}$, where $\vec{v}_i = \mathcal{M}(X_i)$.
- $\mathcal{G} \leftarrow Train\_SVM\_Classifier(\mathcal{V})$
- Return the bag-level classifier $F(X)$ defined as $F(X) = \mathcal{G}(\mathcal{M}(X))$

Fig. 8. Main algorithm of the proposed method.

of the K clusters of instances found in *negative* bags. This way, we obtain a symmetrical bag representation $\mathcal{M}(X) = (H_1(X), \ldots, H_{2KT}(X))$ that considers the responses from classifiers adapted to clusters found in both positive and negative bags. This tends to increase the accuracy of the final bag classifier $F(X)$.

3.2. Discussion: training the cluster-adapted classifiers

As we explained before, the k-th instance-level classifier $h_k(\vec{x})$ is *initialized* using the k-th cluster of instances $\mathcal{C}_k$ and, as a consequence, $h_k(\vec{x})$ *tends* to be adapted to the cluster $\mathcal{C}_k$. However, in order to train the classifier $h_k(\vec{x})$, all the positive instances are considered, instead of using only the instances within the cluster $\mathcal{C}_k$. We explain here the motivation for doing this.

The basic idea is to avoid forcing the k-th classifier $h_k(\vec{x})$ to be adapted to a *contaminated* cluster $\mathcal{C}_k$. We say that a cluster is contaminated if it lies in a region of the instance space that not only contains positive instances but also many negative ones [3]. In Fig.9(a) we show a toy example where $\mathcal{C}_1$ is a cluster that is contaminated with many negative instances. In this example, if we force a classifier $h_1(\vec{x})$ to be adapted to the cluster $\mathcal{C}_1$, we obtain a poor classifier. On the contrary, in Fig.9 we show what happens if we initialize the classifier $h_1(\vec{x})$

[3] In order to simplify the discussion, we consider only clusters of positive instances in this explanation. However, the same idea can be used in clusters of negative instances. In this case, a cluster is contaminated if not only contains negative instances but it also contains many positive ones.

with the cluster $\mathcal{C}_1$ and (at the same time) let this classifier be trained with all the positive instances, not only the ones in the cluster $\mathcal{C}_1$. Let us explain this figure in detail.

In Fig. 9(a), we have two clusters of positive instances (the red points): $\mathcal{C}_1$ is the cluster on the top of the image, and $\mathcal{C}_2$ is the one on the bottom. In addition to this, there is one cluster of negative instances (the blue points) that is located in the same region than $\mathcal{C}_1$. As a consequence, the region of $\mathcal{C}_1$ is populated with both positive and negative points.

Now, given this data, we want to estimate one cluster-adapted classifier $h_k(\vec{x})$ for each cluster of positive instances. Thus, we train a classifier $h_1(\vec{x})$ for the cluster $\mathcal{C}_1$ and a classifier $h_2(\vec{x})$ for $\mathcal{C}_2$. Let us consider the estimation of $h_1(\vec{x})$, which is the problematic case. Fig. 9(b) shows the probability map of the obtained Gaussian model. In order to estimate this Gaussian, we have used the center of the cluster $\mathcal{C}_1$ (the black cross in Fig.9(a)) as initial center of the Gaussian $\vec{t}_0$. Despite the fact that the initial center is in $\mathcal{C}_1$, the EM-DD algorithm detects that this region is densely populated with negative instances, and it pulls the center $\vec{t}$ towards the low part of the image (the green cross symbolizes the resulting center $\vec{t}$). As we can see, the estimated center of the Gaussian $\vec{t}$ is *both* close to positive instances and, *at the same time*, far away from negative ones.

Now let us see what would happen if we trained the classifier $h_1(\vec{x})$ using a training set that contains only the positive instances of the cluster $\mathcal{C}_1$, where the objective is to discriminate between these positive instances and all the negative instances. In this case, if we train the classifier $h_1(\vec{x})$ with only the positive instances of $\mathcal{C}_1$, the estimated center cannot move away from the region defined by $\mathcal{C}_1$. This is because the EM-DD algorithm forces the solution to be close to positive instances, and in this case the only positive instances that EM-DD is aware of are the ones of $\mathcal{C}_1$, so that, necessarily, the solution must lie in this cluster.

This figure has shown just an extreme case in order to show the idea very clearly. In practice, the idea is to let the classifier $h_k(\vec{x})$ be *aware* of all the positive instances in order to make it flexible enough to move it to near parts of the space in the case where the initial region is densely populated with negative instances. At the same time, the estimated Gaussian tends to be adapted to the given cluster $\mathcal{C}_k$ (see examples in Fig. 6) whenever this cluster is free from negative instances. In this sense, we must note that the estimation is done by gradient-descent, which makes it find a local minima $\vec{t}$ near the initial position $\vec{t}_0$. As a result, the estimated center $\vec{t}$ is usually located in positions that are near the initial one and that lie in a region free of negative instances and, at the same time, densely populated with positive ones.

3.3. Comparison between MILDE, EM-DD and MILES

Let us summarize the differences between MILDE and the baselines EM-DD and MILES.

The differences between EM-DD and MILDE can be summarized as follows. First, the EM-DD method assumes that the instances belong to only two classes: the positive and the negative class. Based on this assumption, the EM-DD method estimates a single Gaussian model in order to classify the instances as either positive or negative. This type of method performs poorly when the instances are distributed into several classes. For example, in Fig. 6, the instances

[htb]

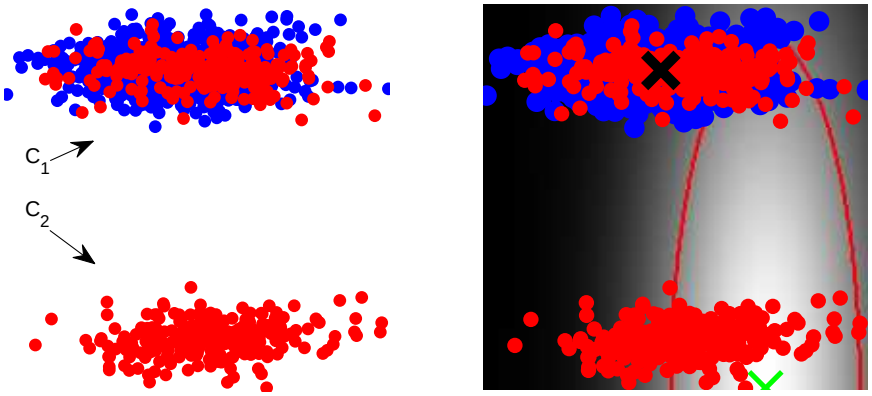


Fig. 9. (a) Example where training the instance-level classifiers $h_k(\vec{x})$ with *all* the positive instances is beneficial. Red and blue points symbolize positive and negative instances, respectively. (b) Probability map obtained by the Gaussian obtained by EM-DD (the higher the intensity the higher the probability). The black cross symbolizes the initial target point $\vec{t}_0$, and the green cross symbolizes the final target point $\vec{t}$, which is the center of the estimated Gaussian.

in the positive bags belong to two different clusters, and trying to fit a single Gaussian that is close to instances of both clusters (and, at the same time, far away from instances in the negative clusters) is not possible. An example extracted from real data where something similar happens is shown in Fig. 11 and discussed in section 5.2.

In contrast, MILDE learns multiple Gaussian models, each one adapted to a different cluster of the instance space. As a result, we obtain a series of Gaussian-based classifiers $H_1, \ldots, H_K$, where $H_k(X)$ provides a high score if any of the instances in X is close to the k-th Gaussian class. A fundamental issue is how the information provided by these models is learned by the method. We do so by defining a mapping function $\mathcal{M}(X) = \vec{v} \in \mathbb{R}^K$, as explained in section 3, which indicates the degree of matching of each Gaussian model with the instances of the bag X. Using this mapping $\mathcal{M}(X)$ we obtain a single feature vector $\vec{v}$ which summarizes the content of the bag X, and this is used to learn bag-level information with a second discriminant learner $\mathcal{G}$. In this sense, the present work evaluates the importance of learning bag-level information for solving MIL problems, versus learning only instance-level information H_k and aggregating this information through simple rules (e.g., the sum and product rules evaluated in section 5).

Now let us see the difference between MILES and MILDE. The MILES method makes use of multiple isotropic Gaussian models, where each one is centered at one positive instance of the training set. If we look at Fig. 6(a), the MILES method would define as many Gaussian models as red points in the figure, all of them with the same scale σ^2 (parameter estimated by cross-validation) and with isotropic shape. If there are R positive instances in the training set,

MILES maps the bag X to a R-dimensional feature vector $\vec{v}$, and this vector is introduced into a standard discriminant learner $\mathcal{G}$, where the authors propose to use the L1-regularized linear SVM. This effectively allows to weight each Gaussian component according to its discriminant power and to remove those components that receive a zero weighting.

In our case, we do not restrict the standard learner $\mathcal{G}$ to any particular choice, it could be the L1-regularized linear SVM or any other. The important difference between MILDE and MILES is in the definition of the Gaussian components. The first important difference is that MILDE produces a series of cluster-adapted models, while MILES is based on a set of isotropic Gaussian models with fixed size and shape. We must note that this is also an important difference between MILDE and EM-DD, where the latter is not aware of any clustering in the data and is based on *the* single Gaussian that obtains the maximum Diverse Density.

The second important difference between MILDE and MILES is the fact that, in MILDE, the Gaussian models are estimated through a *discriminative* learning process. As a result, the Gaussian models in MILDE are located in regions of the space that are close to positive instances and, at the same time, far away from negative ones. This can be observed in Fig. 6(a), where the initial center of the Gaussian (the black cross) is located in the middle of negative instances, and the discriminative learning process *pulls* it towards a near area that is close to positive instances and far from negative ones. As a result, we obtain a richer mapping function, where the components of the vector $\mathcal{M}(X)$ tend to be more discriminative and relevant for the classification task.

Let us conclude by commenting the differences between MILDE and *both* EM-DD and MILES. The first important difference is that neither the EM-DD nor the MILES method are aware of the different clusters existing in the instance space, and they do not describe the content of the bag in terms of how its instances lie close or far from these clusters. In contrast, MILDE learns a series of *cluster-adapted* discriminant models, and use them to define a new mapping that describes how the content of the bag fits *relevant* clusters of the instance space. The second important difference of the proposed work is the fact that it proposes a two-layer architecture for learning *both* instance-level and bag-level information in a discriminative way.

Finally, we must note that the proposed framework is generic and can be used with many instance-level learners, although we make use of EM-DD in this work. Instead of using EM-DD, a similar and interesting alternative is to use other learners such as MI-SVM. The latter algorithm has an initialization and iterative EM-like strategy that is similar to the one of EM-DD, and that works by iterating between learning the model and labelling the instances with the learned model. Using MI-SVM inside our framework would be done in a very similar way to what we do now. In particular, we would learn several MI-SVM models, where the k-th one would be initialized with the k-th cluster in the data and then would be refined in order to discriminate between positive and negative instances. This way we would obtain a series of discriminant models where each one is biased towards a particular region of the instance space, just as we do with EM-DD.

4. Experimental set up

4.1. Parameters of the method

We scaled each component of the instance vectors $\vec{x}$ to the range $[0, 1]$ so that all the dimensions weight the same, as such a scaling is necessary for obtaining good results with the EM-DD algorithm. Regarding the discriminant classifier $\mathcal{G}$, any classifier can be used, and in this work we chose SVM as it is usually one of the best performers. In particular, we used an RBF kernel $K(\vec{x}, \vec{y}) = \exp\left(-\frac{1}{\gamma} D(\vec{x}, \vec{y})\right)$, where $D(\vec{x}, \vec{y})$ is the Euclidean distance. The parameter γ and the penalty cost C of SVM where selected through 5-fold cross-validation as in (Hsu et al, 2003). In particular, (Hsu et al, 2003) suggests to select C among the range of values $2^{-}5, 2^{-3}, \ldots, 2^{15}$ and select γ among the range of values $2^{-15}, 2^{-13}, \ldots, 2^{3}$.

The MILDE method has three parameters: the number of clusters K, the set of scales $\mathcal{S} = \{\sigma_1, \ldots, \sigma_T\}$, and the number of iterations *niter* used in the EM-DD algorithm (see Fig. 5). The values of these parameters were fixed a priori using as criteria both the computational cost and the accuracy. Regarding the computational cost, it is dominated by the cost of the optimization process present in the EM-DD algorithm (in the maximization step of Fig. 5). Let c be the cost of this maximization step, the total cost of the method is $O(K \times T \times niter \times c)$, where the cost of estimating each individual classifier is $O(niter \times c)$, and the number of classifiers is $O(K \times T)$. Given the high cost c of the optimization process, the rest of parameters $K, T, niter$ were chosen as low as possible so as to obtain a moderate total cost.

The number of clusters K was determined as a fraction of the total number of instances N_{inst} of the database: $K = \frac{N_{inst}}{D}$. This is a standard procedure for making sure that the clusters are estimated robustly from the data, i.e., the higher the number of instances the higher the number of clusters that we can estimate robustly. D was chosen heuristically as $D = 16$, which produced good results in a hold-out set different from the test set. However, no attempts to tune this parameter were made, as the main criteria was to obtain a low computational cost with good performance. Regarding the scales, we saw experimentally that using more than one scale is clearly beneficial. We heuristically used $\mathcal{S} = \{1, 0.1\}$, where $\sigma_1 = 1$ is the scale used by the original authors of EM-DD (Zhang and Goldman, 2001) and we found that including a smaller scale $\sigma_2 < \sigma_1$ was beneficial for obtaining a more diverse set of classifiers H_k. We used $\sigma_2 = 0.1$, which produced good results in the hold out set. Finally, regarding the number of iterations *niter*, we fixed $niter = 1$ in order to keep the computational cost reasonable. The first iteration is the most important one in decreasing the energy functional, while subsequent iterations refine the result with a slow pace. Preliminary results did not show a clear accuracy improvement if we use more iterations, while the computational cost increased substantially in this case.

In order to compare MILDE against the most related methods of the literature (EM-DD (Zhang and Goldman, 2001) and MILES (Chen et al, 2006)), we tried to use a setup as similar as possible for all the methods. Let us explain this setup in detail. EM-DD is run by considering multiple initializations, where each initialization consists of one scale parameter $\vec{s}$ and an initial Gaussian center $\vec{t_0}$. In order to use the same conditions as with MILDE, we used the same set of initial scales $\mathcal{S}$ and the same seeds $\vec{t_0}$ (which are the centers of the clusters) in

EM-DD and MILDE. As a result, we run EM-DD with multiple initializations (one for each possible scale $\vec{s}$ and seed $\vec{t_0}$) and, at the end, the EM-DD algorithm outputs the solution (i.e., the Gaussian model) that reaches the highest Diverse Density. Regarding the MILES method, we used three variations:

1. *Baseline*. This is obtained by running the MILDE method with $niter = 0$, i.e., where the target points $\vec{t}$ are the initial center of the clusters, without further optimization, and the scale vectors $\vec{s}$ are set to all ones (i.e., no scaling). Using this baseline is almost equivalent to using MILES, except for two details: i) this baseline uses a non-linear SVM with RBF kernel, whereas the original MILES algorithm uses a L1-regularized linear SVM; ii) both MILDE and this baseline use an embedding vector with $2 \times K$ components, where K is the number of positive and negative clusters, whereas in MILES the embedding vector has R components, where R is the number of positive instances found in the training set, as explained in section 2.4.2. This baseline was used in order to compare the performance when the conditions are as similar as possible, including the learner and the number of components.
2. *Implemented MILES*. In addition to the previous baseline, we also implemented the MILES method using the setup of the original authors, i.e., with the L1-regularized (or 1-norm) linear SVM, and using all the positive instances of the training set (thus obtaining a feature vector with R components).
3. In addition to these implemented versions, we also report the results obtained by the original authors of both EM-DD and MILES, together with the results of other methods in the literature.

In the two implemented versions ("*Baseline*" and ("*Implemented MILES*") the algorithm has two parameters: σ^2, i.e., the scale of the Gaussian along each dimension and C, the cost used in the definition of linear SVM. We used 5-fold cross-validation in order to estimate them. In particular, C is selected among a range of values that is the same as the one suggested in (Hsu et al, 2003): $C = 2^{-}5, 2^{-3}, \ldots, 2^{15}$. Regarding the range of values for σ^2 we used $\sigma^2 = 2^{-2}, 2^{-1}, \ldots, 10^5$. In our experiments, this range of values provided better results than the one suggested in (Chen et al, 2006).

4.2. Databases

MILDE was tested using eight well-known databases whose characteristics are listed in table 1. These databases were created using problems from different disciplines, such as drug discovery (i.e., in medicine), classification of documents (in information retrieval), and classification of images (in computer vision). This is indicated in the second column of table 1, where DD means drug discovery, IR means information retrieval, and CV means computer vision. The task of the *Musk1* and *Musk2* databases is the classification of molecules as either musk or non-musk (Dietterich et al, 1997). In *Text1* and *Text2* the task is, given a text document, decide whether or not it belongs to a given category. These databases were proposed in (Andrews et al, 2003) and have been extensively used since then. In *Fox* the task is, given an image, decide whether or not it contains a fox animal. In *Tiger* and *Elephant* the task is similar except that the animal is tiger and elephant respectively. These databases were also proposed in (Andrews et al, 2003) and are well-known. Finally, the task of the *Corel* database is to classify

Database	Discipline	Number of bags	Number of instances per bag	Total number of instances	Number of dimensions	Classes
Musk1 (Dietterich et al, 1997)	DD	92	5	476	166	2
Musk2 (Dietterich et al, 1997)	DD	102	65	6598	166	2
Text1 (Andrews et al, 2003)	IR	400	8	3224	66552	2
Text2 (Andrews et al, 2003)	IR	400	8	3344	66552	2
Fox (Andrews et al, 2003)	CV	200	7	1320	6	2
Tiger (Andrews et al, 2003)	CV	200	6	1220	6	2
Elephant (Andrews et al, 2003)	CV	200	7	1391	7	2
Corel (Chen et al, 2006)	CV	2000	4	7947	9	20

Table 1. Databases used in the experiments.

an image into one of twenty categories. This database was proposed in (Chen et al, 2006) and is also widely known.

In all the databases (except for the *Corel* one) we used a ten-fold validation approach, as the majority of authors (Chen et al, 2006): each round we take 90% of the data for training and the remaining 10% for testing, and this is repeated ten times in order to test with all the bags. In the *Corel* database we used a two-fold validation approach as the authors do (Chen et al, 2006).

The *Text1* and *Text2* databases have a very large dimensionality (see table 1, column 6), which makes it infeasible to apply the EM-DD method. Furthermore, we could not use standard methods such as PCA in order to reduce the dimensionality, due to the fact that the very high dimensionality makes it not possible to build a covariance matrix due to memory issues. In order to reduce the dimensionality we used a simple heuristic that consists of selecting the first 3000 dimensions that have higher variance. Then, we applied PCA on the resulting data, and after this we selected the first 50 dimensions. We also tested higher dimensionalities, but the results were not better, so we selected a low dimensionality for computational efficiency. In all the cases, the same setting was used for the benchmark methods (EM-DD and MILES), in order to obtain a fair comparison.

5. Results

Table 2 presents the accuracy (classification hit rate) of MILDE for the eight databases. It also presents the results obtained with our implementation of MILES (Chen et al, 2006) and our implementation of EM-DD (Zhang and Goldman, 2001), which is based on the public implementation provided by Yang (Yang, 2005). In order to implement both EM-DD and MILES, we used the setup explained at the end of section 4. Regarding the MILES method, we used two variations, explained also at the end of section 4. The first one, called *implemented MILES* uses the 1-norm linear SVM learner and the same embedding used by the original authors (Chen et al, 2006). The second one, called *baseline* is obtained by running MILDE with $niter = 0$, i.e., where the target points $\vec{t}$

Method / DB	Musk1	Musk2	Text1	Text2	Fox	Tiger	Elephant	Corel
Implemented EMDD	85	81.4	74.7	76.5	57	70	77	43
Implemented MILES	**88.1**	84.7	91	71.75	54.5	76.5	78	69.7
Baseline	86.1	90.2	**94.5**	79.8	63.1	80.8	83.1	68.4
MILDE	87.1	**91**	94.2	**84.7**	**66.5**	**83**	**85**	**74.8**

Table 2. Comparison between MILDE and our implementation of EM-DD and MILES.

Method / DB	Musk1	Musk2	Text1	Text2	Fox	Tiger	Elephant	Corel
MILDE	87.11	**91**	**94.25**	**84.75**	**66.5**	**83**	**85**	**74.8**
EMDD	84.8	84.9	85.8	84	56.1	72.1	78.3	-
MILES	86.3	87.7	-	-	-	-	-	68.7
MI-SVM	77.9	84.3	93.7	76.4	58.8	66.6	73.1	54.6
mi-SVM	**87.4**	83.6	90.4	74.3	57.9	78.9	80	-

Table 3. Comparison between MILDE and related methods from the literature. MI-SVM and mi-SVM denote two different methods.

are the initial center of the clusters, without further optimization, and the scale vectors $\vec{s}$ are set to all ones (i.e., no scaling). This way we obtain a method that is very similar to MILES and that uses the same learner (non-linear SVM) and number of Gaussian components as our MILDE method, which makes it a good baseline. As can be seen in table 2, MILDE outperforms EM-DD in all the databases, it outperforms the baseline in all but one database, and also outperforms the implemented MILES in all but one database, most of the times by a wide margin. Note in particular the performance on the Corel database, proposed by the authors of the MILES method. In this database MILDE outperforms all the methods by a wide margin.

Table 3 presents the accuracy of different methods from the literature, including both EM-DD and MILES as reported by the original papers. The numbers shown for the MI-SVM and mi-SVM methods are based on the RBF kernel, which is the one used by the authors in (Andrews et al, 2003) for all the databases including Musk1 and Musk2 (they also report results with other kernels but not for all the databases), and which is the same kernel used in the implementation of MILDE. We can see that the proposed MILDE method outperforms all the others in all but one database.

Recently, some authors have proposed to add optimization layers on top of these instance-level methods such as mi-SVM and MI-SVM. In particular, Gehler and Chapelle improve these methods by using deterministic annealing optimization (Gehler and Chapelle, 2007). On top of this, Antic and Ommer (Antic and Ommer, 2013) show even further improvements when, in addition to deterministic annealing, the method makes use of multiple subsets of bags (called *Superbags*) in order to obtain a robust labelling of the instances. Table 4 shows the result of applying deterministic annealing plus Superbags (Antic and Ommer, 2013). Different rows of table 4 correspond to different configurations of

Method / DB	Musk1	Musk2	Tiger	Elephant	Fox
MILDE	87.11	**91**	83	**85**	66.5
AW-SVM + SuperBags	85.8	86.2	85.5	82.5	67
AL-SVM + SuperBags	86.9	82.6	83.5	82.5	67
ALP-SVM + SuperBags	**87.9**	86.6	**86**	84	**69**

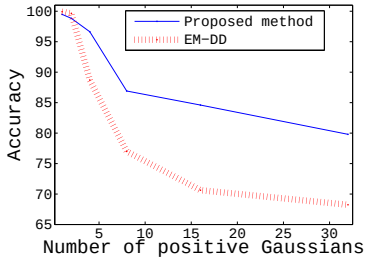
Table 4. Comparison between MILDE and SuperBags-based methods

Method / DB	Musk1	Musk2	Text1	Text2	Fox	Tiger	Elephant	Corel
Proposed method	**87.1**	**91**	**94.2**	**84.7**	**66.5**	**83**	**85**	**74.8**
Sum of models	73.1	66.6	47	53.2	45	48.5	55	0.3
Product of models	75.1	81.3	44.2	52.5	48	53.5	57.5	4.4

Table 5. Comparison of performance between the proposed embedded-based method (MILDE), and other aggregation-based methods

mi-SVM, MI-SVM and the deterministic annealing optimization (Gehler and Chapelle, 2007), and we only show results for the five databases of table 1 that are also used by the authors of these methods (Antic and Ommer, 2013). MILDE continues to rank among the best performers: it is either the best or the second best in four out of five databases: it is the best performer in *Musk2* and *Elephant*, it matches the performance of the best method in *Musk1* and has a similar performance than the one of the second best method in *Fox*. We must note that MILDE is based on estimating multiple instance-level models (and we use an adapted EM-DD at the core of this process). This EM-DD instance-level algorithm could also benefit from adding optimization layers such as SuperBags. The results shown in table 4 suggest that the final accuracy of MILDE would probably increase further if we added these optimization layers, indicating an interesting line of research. Nonetheless, MILDE still ranks among the top performers even without these optimizations.

In table 5 we evaluate to what extent it is necessary to use an embedded-based method such as ours. For this purpose, we compare the performance of MILDE against two other methods that simply aggregate the scores of the cluster-adapted classifiers. The first method makes use of a sum aggregation rule, $F(X) = \sum_k H_k(X)$, whereas the second one is based on a product rule, $F(X) = \prod_k H_k(X)$. These two methods fall into the traditional paradigm, as the discriminant learning occurs only at the instance level. At the bag level there is no learning, but simply an aggregation of scores. In contrast, MILDE is based on both instance-level and bag-level learning. In this sense, recall that $F(X)$ is computed as $F(X) = \mathcal{G}(\mathcal{M}(X))$, where $\mathcal{G}$ is trained using bag-level representations $\vec{v}_i = \mathcal{M}(X_i)$. Table 5 clearly shows the superiority of learning both at the instance and at the bag level. In particular, the aggregation-based methods are very poor in the image classification task, and they are also significantly worse in the other databases.



[htb] Fig. 10. Accuracy as a function of the number of classes of instances.

5.1. Analysis of performance under controlled conditions

We also used a synthetic data set in order to analyze the performance under controlled conditions. In this data set, the positive instances were generated by using N_p Gaussian components, and we studied the performance as a function of N_p (see Fig. 10) [4]. For example, let us consider the case $N_p = 1$. In this case we are reproducing a MIL problem where the positive instances come from a single Gaussian class. This is the classical scenario assumed by the methods of the traditional paradigm (see section 2.4.1), including the EM-DD method. For values $N_p > 1$, the positive instances fall into several classes, and the assumption used by the traditional paradigm is no longer valid. This is what we discussed in section 2.3.2 for the image classification problem, where the positive instances fall in more than one class ("sand" and "sea"), i.e., $N_p > 1$. In this type of scenario, the traditional paradigm tends to fail. Furthermore, the higher the number of Gaussian classes, the larger the decrease in performance of this paradigm.

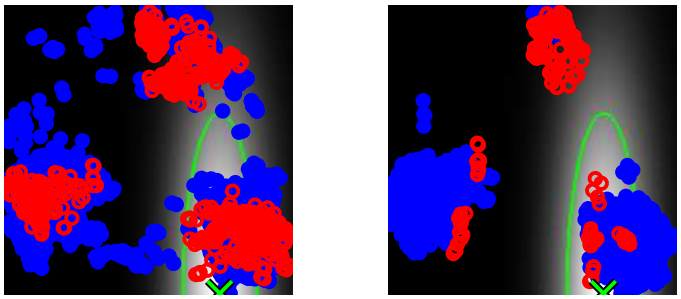
This is shown empirically in Fig. 10. This figure shows the accuracy obtained as a function of N_p, where we evaluated both EM-DD (representing the traditional paradigm) and the embedded-based MILDE method proposed in this work. We can see that the accuracy of both methods is similar when $N_p \approx 1$, i.e., when the assumption followed by the traditional methods holds. However, the difference between MILDE and EM-DD rapidly increases as N_p increases.

5.2. Illustration of performance in two dimensional data

In order to visualize the instance space, we projected the instances to two dimensions using PCA. This was done for the Musk1 and Musk2 databases, and we call the projected databases Musk1-2D and Musk2-2D. Fig. 11 shows the instance space for the Musk1-2D database, where we show the training data in Fig. 11(a), and the test data in Fig. 11(b). The difference between the training data and the test data is due to the fact that 90% of the data is used for training and only 10% is used for testing. The images also show the Gaussian model obtained with the EM-DD method, where the green cross represents the center of

[4] Regarding the negative instances, they were generated by using a constant number of Gaussian components, $N_n = 32$. We used a large number of components for the negative class in order to obtain realistic data. In this sense, note that the negative class is usually defined as the negation of the positive class, i.e., it groups all the classes of objects that are not positive, and thus forms an heterogeneous class.

[htb]



(a) (b)

Fig. 11. Instance space in projected Musk1 database, and EMDD probability map.

Method / DB	Musk1-2D	Musk2-2D
EMDD	54.6	57.6
Sum of models	51.1	41.9
Product of models	49.4	61.8
MILES	70.7	64.9
Proposed method	**78.1**	**67.9**

Table 6. Classification accuracy of methods in projected databases

the Gaussian, the green ellipse represents the decision boundary (the instances inside the ellipse are classified as positive), and the intensity of white represents the probability map. Note that the original EM-DD algorithm in (Zhang and Goldman, 2001) only obtains one Gaussian model, which is the one shown in the images.

If we look at the training data (Fig. 11(a)), we can see that the instances are distributed into several clusters, and there is a high degree of overlapping between positive and negative instances. For this type of data, using a single instance-level model is poor. It is much more robust to use a bag-level classifier based on multiple instance-level models, as we do in the MILDE method. Table 6 shows the classification accuracies of MILDE. The proposed method consistently outperforms both EM-DD and MILES in both databases. Note the large improvement of MILDE over MILES, especially for the *Musk1* database, which confirms that learning instance-level information is also important. Also, we can see that the good performance of MILDE is not due to the use of multiple Gaussian models. In this sense, simply aggregating multiple models leads to results that are similar to the EM-DD method, and clearly worse than *both* MILDE and MILES. This clearly demonstrates the importance of learning at the bag level.

6. Conclusions

In this work we proposed a new embedded-based Multiple Instance Learning method. The proposed method was designed based on an analysis of the existing paradigms: the traditional one that learns a discriminative model at the instance-level, and the embedded-based one that learns a discriminative model at the bag-level. Based on such an analysis we introduced a generic meta-algorithm that exploits the strengths of both types of approach. This is based on a new embedding that makes use of cluster-adapted discriminant instance-level classifiers. The proposed schema can be instantiated with any of a number of instance-level classifiers, and in particular in this work we make use of the EM-DD discriminative learner. This permits to obtain a set of Gaussian models where each one is adapted to one cluster of instances. At the same time, each model takes into account the separation between positive and negative instances in the data. In addition to this, we show that if we use EM-DD in our framework we obtain a generalization of the MILES method (Chen et al, 2006).

The proposed method was evaluated using eight well-known MIL databases and compared against related state-of-the-art approaches. The results showed that the proposed method consistently outperformed all of them in the large majority of the databases. In order to deepen the analysis, we also studied the behavior of the methods under controlled conditions, using synthetically generated MIL data. The results clearly show that an embedded-based approach such as ours becomes necessary when the positive instances are not concentrated in a single region of the space, and instead they are scattered across different regions.

Altogether, this paper presents both a comprehensible characterization of the existing paradigms and, based on this, a new framework that embraces new interesting possibilities. This includes new mechanisms for learning both at the instance and at the bag levels. In this sense one possibility would be to use stronger instance-level learners (in our implementation we use simple Gaussian models) which might incur into some over-fitting, and thus could be combined with a regularized learner at the second layer.

Acknowledgements. We thank anonymous reviewers for their very useful comments and suggestions. This work was supported by the fellowship RYC-2008-03789 and the spanish project TRA2011-29454-C03-01.

References

T. G. Dietterich, R. H. Lathrop, T. Lozano-Perez, Solving the multiple-instance problem with axis-parallel rectangles, Artificial Intelligence 89 (1997) 31–71.

S. Kotsiantis, D. Kanellopoulos, Multi-instance learning for bankruptcy prediction, in: Proc. of Int. Conf. on Convergence and Hybrid Information Technology, 2008, pp. 1007–1012.

Y. Chen, J. Bi, J. Z. Wang, MILES: Multiple-instance learning via embedded instance selection, IEEE Trans. on Pattern Analysis and Machine Intelligence 28 (2006) 1931–1947.

D. A. Reynolds, T. F. Quatieri, R. B. Dunn, Speaker verification using adapted Gaussian mixture models, Digital Signal Processing 10 (2000) 19–41.

M. Oded, Learning from ambiguity, Ph.D. thesis, MIT (May 1998).

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

S. Andrews, I. Tsochantaridis, T. Hofmann, Support vector machines for multiple-instance learning, in: Proc. Neural Information Processing Systems, 2003, pp. 561–568.

X. Xu, E. Frank, Logistic regression and boosting for labeled bags of instances, in: Proc. Pacific Asia Conf. on Knowledge Discovery and Data Mining, 2004, pp. 272–281.

P. V. Gehler, O. Chapelle, Deterministic annealing for multiple-instance learning, in: Proc. Int. Conf. on Artificial Intelligence and Statistics, 2007, pp. 123–130.
B. Antic, B. Ommer, Robust multiple-instance learning with superbags, in: Lecture Notes in Computer Science, Vol. 7725, 2013, pp. 242–255.
S. Scott, J. Zhang, J. Brown, On generalized multiple-instance learning., Int. Journal of Computational Intelligence and Applications 5 (2005) 21–35.
Z.-H. Zhou, M.-L. Zhang, Solving multi-instance problems with classifier ensemble based on constructive clustering, Knowledge and Information Systems 11 (2007) 155–170.
T. Gärtner, P. A. Flach, A. Kowalczyk, A. J. Smola, Multi-instance kernels, in: Proc. Int. Conf. on Machine Learning, 2002, pp. 179–186.
J. Foulds, E. Frank, A review of multi-instance learning assumptions, The Knowledge Engineering Review 25 (2010) 1–25.
E. Nowak, F. Jurie, B. Triggs, Sampling strategies for bag-of-features image classification, in: Proc. European Conf. on Computer Vision, 2006, pp. 490–503.
K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. V. Gool, A comparison of affine region detectors, Int. Journal on Computer Vision 65 (2005) 43–72.
O. Maron, T. Lozano-Pérez, A framework for multiple-instance learning, in: Proc. Neural Information Processing Systems, 1998, pp. 570–576.
R. Bunescu, R. Mooney, Multiple instance learning for sparse positive bags, in: Proc. Int. Conf. on Machine Learning, 2007, pp. 105–112.
Z.-H. Zhou, Y.-Y. Sun, Y.-F. Li, Multi-instance learning by treating instances as non i.i.d. samples, in: Proc. Int. Conf. on Machine Learning, 2009, pp. 1249–1256.
M.-L. Zhang, Z.-H. Zhou, Multi-instance clustering with applications to multi-instance prediction, Applied Intelligence 31 (2009) 47–68.
R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, Liblinear: A library for large linear classification, Journal of Machine Learning Research 9(2008), 1871–1874.
C.-W. Hsu, C.-C. Chang, C.-J. Lin, A practical guide to support vector classification, Tech. rep., National Taiwan University (2003).
J. Yang, Mill: A multi-instance learning library, http://www-2.cs.cmu.edu/ juny/IR-lab/ (2005).

Author Biographies

Jaume Amores received the Ph.D. degree from the Universitat Autònoma de Barcelona (UAB), Bellaterra, Spain, in 2006. He has held positions in the Computer Vision Center, UAB, and in the Institut National de Recherche en Informatique et en Automatique (INRIA), in Rocquencourt-Paris. Currently, he holds a Ramon y Cajal fellowship as part of the Advanced Driver Assistance Systems Group. His current research interests include machine learning and pattern recognition, object recognition and detection, and medical imaging.

Correspondence and offprint requests to: Jaume Amores, Email: jaume@cvc.uab.es