Chapter 5
Bag-Based Classification Methods

Abstract In bag-based multi-instance methods, the main learning process occurs at the level of bags. In this chapter, we analyze two important subcategories of bag-based MIL classifiers. On the one hand, in Sect. 5.2, we examine classifiers that define a distance or similarity measure between bags to work directly in the original bag space. On the other hand, Sect. 5.3 is devoted to mapping-based classifiers that transform each bag to a single-instance representation such that the learner can train any single-instance classifier to label new bags.

5.1 Introduction

As opposed to instance-based classification methods, the learning process of bag-based methods occurs at bag level. The main feature that distinguishes bag-based from instance-based classifiers is that the former can predict the label of a new bag considering each training bag as a whole entity, without the need to discover any hidden instance labels. Instance-based classification methods need to construct an instance classifier that is as accurate as possible, but this is not a requirement for bag-based methods. Although some types of bag-based classifiers do train an instance-level learning model, it is only used as a rough guide to the main bag-level learning process. Moreover, the MI assumption of bag-based methods need typically not be as precise as is the case for instance-based methods, but can be more flexible and general. We discuss the following two important subcategories of bag-based methods:

- **Bag-based methods that work in the original bag space**: these methods rely on a metric function defined over bags. The metric is used in a distance-based classification algorithm, e.g., a nearest neighbor algorithm. By introducing the bag-wise distance measure, the learner is effectively upgraded to a full-fledged MI classification algorithm. We refer to these methods as *original bag space classification methods* (original-BS methods, for short) and discuss them in more depth in Sect. 5.2.
- **Bag-based methods that work in a mapped space**: these methods transform the multi-instance data into a single-instance representation and train a single-instance

F. Herrera et al., *Multiple Instance Learning*, DOI 10.1007/978-3-319-47759-6_5

classifier on the transformed data. The same transformation is applied to an unseen bag and its class label is predicted by the single-instance classifier learned in the mapped space. We refer to these methods as *mapped bag space classification methods* (mapped-BS methods, for short). They are discussed in Sect. 5.3.

5.2 Original Bag Space Methods

In single-instance learning, each instance is interpreted as a point in a multidimensional space determined by the features of the problem at hand. Many traditional single-instance learning algorithms rely on a distance function between points of this space to determine separating boundaries between classes. In MIL, bags can be understood as regions in the instance space and a bag-wise distance function is required to evaluate similarity relations between them. Using such a bag-wise distance function in a traditional distance-based learning algorithm, it becomes a multi-instance algorithm able to locate bag class boundaries. The two main design options of any bag-distance-based classification method are

- **A distance-based classification method**: we describe two distance-based methods: nearest neighbor methods (Sect. 5.2.1) and kernel methods (Sect. 5.2.2).
- **A bag-wise distance/similarity function**: recall that similarity functions can be used instead of distance functions by inverting the objective function of the learner. Both types of comparison measures are complementary and using one or the other depends on the definition of the bag label prediction method. In Sect. 3.5, we listed several distance and similarity functions that can be used in these algorithms.

5.2.1 Nearest Neighbor Methods

The CitationKNN algorithm was proposed in [20] and extends the traditional single-instance k-nearest neighbors method (KNN) to the level of bags. To classify a new bag X, CitationKNN uses a distance function between bags to determine which training bags are closest to X. Inspired by the concept of citations in the field of information science, this algorithm extends the set of nearest neighbors to consider not only the r bags closest to X (references, Fig. 5.1), but also the bags for which X is among the c closest bags (citers, Fig. 5.2). A voting scheme uses the class labels of both references and citers to determine the class label of X.

Any bag-wise distance function can be used in CitationKNN (see Sect. 3.5). In particular, the study of [20] uses the minimal Hausdorff distance (3.18), maximal Hausdorff distance ((3.19), (3.20)) and k-th ranked Hausdorff distance (3.22).

The distance function employed in CitationKNN has a major impact on its performance [2]. Each application domain can benefit more from a certain distance function than from others and some applications may require the selection of a less

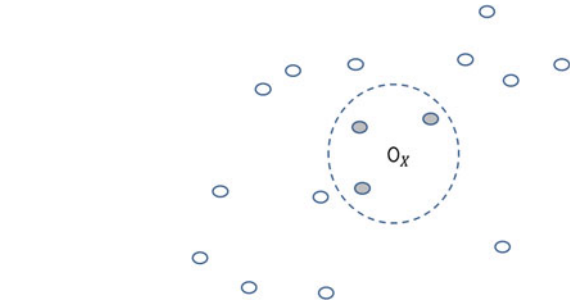


Fig. 5.1 References. The *circle* encompasses the nearest 3-references to X (*filled balls*). The closest references correspond to the (traditional) nearest neighbors

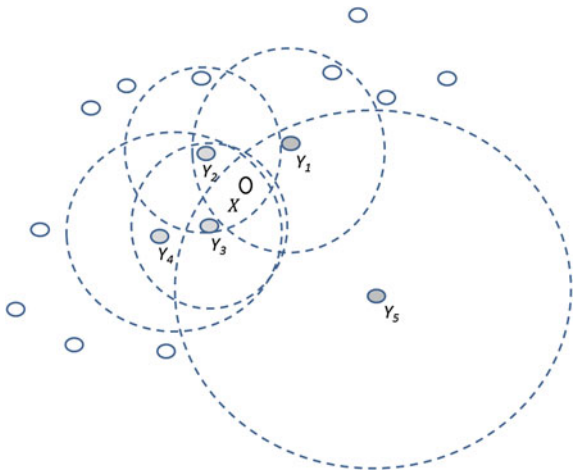


Fig. 5.2 Citers. The 3-citers nearest to X (*filled balls*) are those whose three nearest neighbors include X. Each *circle* contains the three nearest neighbors of the sample located at its *center*. For clarity, we have only represented *circles* including X. These are the 3-nearest citers to X

conventional metric. For example, the work of [27] on a web mining application adapts CitationKNN for text data represented by sets of terms, rather than the traditional attribute-value vector representation suffering from the so-called curse of dimensionality. They represent an instance x by a set of textual terms $\{t_1, t_2, \ldots, t_n\}$, where t_i $(i = 1, \ldots, n)$ is one of the n more frequent terms in the text fragment corresponding to x. They use the minimal Hausdorff distance variant, i.e., $k = 1$ in (3.22), and define a distance function between two instances $a = \{a_1, a_2, \ldots, a_n\}$ and $b = \{b_1, b_2, \ldots, b_n\}$ as

$$\|a - b\| = 1 - \sum_{\substack{i,\, j = 1 \\ a_i = b_j}}^{n} \frac{1}{n},$$

based on the idea that the fewer common terms two instances share, the greater the distance between them.

CitationKNN has been extended to regression tasks [8], clustering [13] and multi-label classification [24]. It has been used successfully in several application domains, such as textual classification [21] and anomaly detection [23].

5.2.2 Bag-Level SVM

Bag-level kernels are used to measure the similarity between two bags in a transformed representation space. They operate on whole bags and return a single number assessing how close the two bags are. As stated in Sect. 3.5, the similarity is inversely related to the distance. Kernel-based methods, as well as distance-based ones, rely on space metrics to find the separating class boundaries. When a bag-level kernel is used in a standard SVM, the latter becomes able to optimize the margin between bag classes without any modification to the SVM itself. One of the first bag-level kernels was presented by Gärtner et al. [11]. They define the set kernel between two bags A and B as

$$k_{MI}(A, B) = \sum_{a \in A, b \in B} k_I^p(a, b),$$

where k_I is a kernel defined at the instance level. Theoretically, for sufficiently large values of p, this kernel ensures the separability of the training set. Because of the computational cost involved in the MI kernel above, [11] defines a minimax kernel based on the minimum and maximum attribute values of instances in each bag, namely

$$k(A, B) = (\langle s(A), s(B) \rangle + 1)^p,$$

where $s(\cdot)$ defines the attribute transformation

$$s(X) = \left\langle \min_{x \in X} x_1, \ldots, \min_{x \in X} x_m, \max_{x \in X} x_1, \ldots, \max_{x \in X} x_m \right\rangle.$$

In the MI kernels proposed by Gärtner et al. [11], all attributes are treated with equal weight. On the other hand, Blaschko et al. [3] propose conformal kernels which can locally reduce or expand each attribute dimension based on the discriminative importance of each attribute, while preserving the angles between vectors in the transformed space.

Kwok and Cheung [14] present marginalized kernels, that assume that the data are generated by a latent variable model. The observed variable is the bag and the hidden variable is its label. In particular, let $Z_1 = (X_1, \ell_1)$ and $Z_2 = (X_2, \ell_2)$ be two bags with their respective class labels. A joint kernel is defined as

$$k_Z(Z_1, Z_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} k_\ell(\ell_{1i}, \ell_{1j}) k_x(x_{1i}, x_{2j}),$$

where $k_\ell(\cdot,\cdot)$ is a kernel defined over the instance labels and $k_x(\cdot,\cdot)$ is a kernel defined over the instance space. The marginalized kernel, defined over two observed variables X_1 and X_2, is obtained by taking the expectation of the joint kernel with respect to the hidden variables ℓ_1 and ℓ_2, that is,

$$k(X_1, X_2) = \sum_{\ell_1 \in \mathbb{L}} \sum_{\ell_2 \in \mathbb{L}} P(\ell_1, X_1)\, P(\ell_2, X_2)\, k_Z(Z_1, Z_2)\,. \tag{5.1}$$

It is possible to calculate this marginalized kernel in polynomial time. The posterior distribution of ℓ_1 and ℓ_2 is obtained from a probabilistic model $P(\ell_i|X_i)$ estimated from the data.

Bag-level kernels make an implicit transformation of bags into a single-instance representation such that standard SVMs can be directly applied to multi-instance data. In Sect. 5.3, we show that an explicit transformation of the bags can also be set up to obtain a single-instance dataset on which any single-instance learner can be trained and used to predict bag labels.

5.3 Mapped Bag Space Methods

The multi-instance classification algorithms described in Chap. 4 and Sect. 5.2 are based on single-instance classifiers that have been modified to function in the MIL setting. Although good results have been reported in many applications for these multi-instance algorithms, the high cost of developing new algorithms today limits the applicability of this approach. There is only a small number of multi-instance algorithms compared to the large number of methods and algorithmic variants that have been developed for single-instance learning.

In this section, we examine another approach to solving multi-instance classification problems. Instead of using a modified single-instance classifier, a transformation is applied to the multi-instance data resulting in a single-instance representation of bags. In this new data representation, it is possible to construct a classification model using any traditional single-instance algorithm, effectively solving the multi-instance classification problem. The single-instance representation of multi-instance data not only allows the use of any single-instance classifier, but also the application of data preprocessing techniques, such as editing, cleaning, and dimensionality reduction, which have been well studied in single-instance learning.

In map based methods, the learning process occurs at bag level, but always relies on a mapping process. These methods transform the original multi-instance representation, in which each bag is a set of points (instances) in the attribute space, into another form of representation in which each bag is represented as a single point of the induced space. The multi-instance problem effectively becomes a single-instance problem to which any traditional learning algorithm can be applied.

Map based methods differ among each other in their specific mapping processes. In general, the following procedure is used. The methods are based on a function

$\mathscr{M} : \mathbb{N}^{\mathbb{X}} \rightarrow \langle a_1, \ldots, a_d \rangle$ that transforms the multi-instance representation of a bag X into a single vector $\mathscr{M}(X) = \langle a_1, \ldots, a_d \rangle$. The multi-instance training set is transformed in a single-instance training set by applying the function $\mathscr{M}$ to each training bag. Any suitable single-instance classifier is built on the new training set. To classify a new bag, it is first converted to the new space using $\mathscr{M}$ and is then fed to the classifier which predicts a label class. By representing a bag as a point in the new space, some of the inherent ambiguity and imprecision of the multi-instance dataset can be reduced. However, as it is practically impossible to eliminate it completely, some of the original ambiguity remains encoded in the attribute values of each vector. The amount of the ambiguity reduction depends on the design of the mapping function $\mathscr{M}$.

Mapping-based classification algorithms differ primarily in the design of $\mathscr{M}$ and the mapping process. Below, we examine each of these mapping strategies and the classifiers that use them. For a better understanding, we have made a division in four categories, considering the meaning of the attributes in the new representation space

- **Mapping methods based on bag statistics (Sect. 5.3.1)**: each attribute of the new mapping space is the value of a statistic that is applied to the set of values of the corresponding attribute in the original representation space.
- **Mapping methods based on representative instance concatenation (Sect. 5.3.2)**: each vector of the new mapping space is the concatenation of N instances of the bag, where each instance is a representative of one pattern in the instance space.
- **Mapping methods based on counting (Sect. 5.3.3)**: each attribute of the new mapping space indicates presence, amount or frequency of instances of the bag in a specific region of the instance space.
- **Mapping methods based on distance (Sect. 5.3.4)**: each attribute of the new mapping space represents the distance (or similarity) of the bag to a specific region of the instance space.

5.3.1 Mapping Methods Based on Bag Statistics

Bag statistics-based methods seek to represent each bag by a single attribute vector that summarizes the statistical information of the bag. Consider a bag $X = \{x_1, \ldots, x_n\}$ in which each instance is described by d attributes, i.e., $x_i = \left\langle x_i^1, \ldots, x_i^d \right\rangle$, $\forall i \in [1, \ldots, n]$. The bag can be seen as a set of d random variables with unknown probability distribution, for which we have a sample of size n. Several statistics can be used to characterize the probability distribution of these random variables. In the new attribute space, in which the multi-instance examples are mapped, each attribute of the original space is represented by one or more statistic values, that attempt to capture the shape of the probability distribution of the original variable within the bag. We list some examples of the kind of transformation performed on the bags

- **Average mapping**: $M(X) = \langle m_1, \ldots, m_d \rangle$, where m_j is the mean value of the jth attribute over all the instances of X. This transformation is used by the SimpleMI algorithm described in [7] and included in the experiments of Sect. 5.4.
- **Min-Max mapping**: $M(X) = \langle a_1, \ldots, a_d, b_1, \ldots, b_d \rangle$, where $a_j = \min_i (x_i^j)$ and $b_j = \max_i (x_i^j)$ are the minimum and maximum values of the jth attribute over all the instances in X. This transformation is used by the Min-Max kernel proposed by [11].
- **Moments mapping**: $M(X) = \langle m_1, \ldots, m_d, v_1, \ldots, v_d, s_1, \ldots, s_d, k_1, \ldots, k_d, \rangle$. The values m_j, v_j, s_j and k_j represent the first to fourth statistical moment (i.e., mean, variance, skewness, and kurtosis) of the jth attribute of the instances in X.

The dimension of the new mapped space is the number of dimensions of the original space multiplied by the number of statistics used to describe each variable.

Stratified Bag Statistics

The methods described above are limited to summarize statistical information of *all* instances inside the bag and do not consider that within the same bag different patterns can coexist. In different instance patterns, one or more attributes can have different probability distributions. If all instances of the bag are treated as if they belonged to the same pattern, the statistics will be unable to adequately describe the mixture of distributions. A more sophisticated mapping method can try to discover patterns or classes of instances in the data and represent each bag in the embedded space with the statistics values of each original attribute for each instance pattern separately. We call *stratified bag statistics-based mapping*.

The most common way to discover instance patterns in the data is to use unsupervised methods, since instance class labels are unknown. Unsupervised methods allow to find groups of instances with shared characteristics. These groups can be considered as different instance classes. We can also use supervised methods, assuming that instances are assigned to the same class labels of their bags. Clearly, this assumption can cause a certain proportion of mislabeled instances, but the goal is to obtain a first approximation of the underlying instance-level patterns. From this first approximation, a learning algorithm can be trained to obtain a more accurate instance-level classifier.

Learning methods based on stratified bag statistics represent each bag by a single attribute vector with statistical information of the different patterns or instance classes contained in the bag. The new attribute values related to each instance pattern are concatenated in the vector describing the bag. Let $C_1, \ldots, C_k$ be instance patterns found in the data and $\theta : \mathbb{N}^{\mathscr{A}} \rightarrow \mathbb{R}$ a statistic (e.g., average, minimum, maximum, or moments) applicable to the d attributes of a set of instances. The stratified bag statistics based mapping is defined as

$$M(X) \mapsto \langle \theta_{11}, \ldots, \theta_{1d}, \theta_{21}, \ldots, \theta_{2d}, \ldots, \theta_{k1}, \ldots, \theta_{kd} \rangle , \tag{5.2}$$

where θ_{ij} represents the statistic value applied to the jth original attribute of the instance subset in the bag belonging to the ith pattern. Equation 5.2 represents the

case where each attribute probability distribution is described by a single statistic, but in general several statistics can be used for each attribute. The dimension of the new embedded space is $d \times k \times q$, where d is the number of dimensions of the original space, k is the number of patterns or classes of instances discovered in the data and q is the number of statistics used to describe each original attribute distribution (e.g., in the Min-Max mapping two statistics are used, so $q = 2$).

5.3.2 Mapping Methods Based on Prototype Concatenation

This approach was introduced by Boughorbel et al. [4]. They look for k instance patterns in the data and characterize each pattern C_i through its center p_i. However, instead of using statistics operating on individual attributes, they use a function $\varphi(X, p_i) : \mathbb{N}^{\mathbb{X}} \times \mathbb{X} \rightarrow \mathbb{X}$ to select the instance in the bag closest to the center p_i of the ith pattern and use that instance as the pattern representative. The mapping by Boughorbel et al. can be defined as $M(X) \mapsto \langle v_1, v_2, \ldots, v_k \rangle$, where v_i is the instance from X that is closest to the center p_i of the ith pattern. The authors use this transformation to construct an SVM with an ad hoc kernel. However, as with all mapping methods described in this chapter, any other single-instance learning algorithm can be applied to the mapped data as well.

This method can be generalized so that an aggregation of all instances of the bag is used to represent the matching degree between the bag and the instance pattern. Let $S(x, C) \in [0, 1]$ be a function that measures the matching degree between an instance x and a pattern C. A natural way of defining $S(x, C_i)$ is as a similarity measure between instance x and the center of the ith pattern p_i. The vector v_i can be calculated as

$$v_i = \frac{\sum\limits_{x \in X} x \cdot S(x, C_i)}{\sum\limits_{x \in X} S(x, C_i)}, \tag{5.3}$$

which represents the average of the instances weighted by their matching degree with the pattern. This method is related to the stratified statistic mapping method described in Sect. 5.3.1. When the matching function $S(x, C)$ is binary, so that $S(x, C)$ equals 1 if the similarity between x and C is above a given threshold and $S(x, C)$ equals 0 otherwise, we can use (5.2) to compute v_i using the average as the only statistic. In the other case, if the matching function takes on continuous values in the interval $[0, 1]$, we have a generalization of (5.2), where the value of each attribute is weighted with a matching degree.

5.3.3 Mapping Methods Based on Counting

This group of methods represent each bag as a single vector, where each attribute is the number of instances of the bag that are found in a specific region of the

instance space. In other words, they describe the relationship between the bag label and instance classes covered by different regions of the instance space.

Multi-instance classifiers using a counting-based mapping are strongly inspired by the MI assumptions hierarchy of Weidmann et al. (Sect. 3.4.2). Some algorithms create binary attributes in the mapping process, where the ith attribute indicates the presence or absence of instances of the bag in the ith region. These algorithms allow to model the presence-based assumption, including the standard MI assumption. Other algorithms create attributes that take on positive numeric values representing absolute or relative frequencies of the instances belonging to the bag and lying inside the corresponding region. These algorithms allow to model the threshold and counting-based MI assumptions.

We can further divide this group into two major categories the acquisition of the MI assumption into account. On the one side are those algorithms for which the designers decide in advance which MI assumption is used. This category is examined in Sect. 5.3.3.1. On the other side we consider the algorithms for which no MI assumption has been specified. They learn the hypothesis from the data during execution. Section 5.3.3.2 is devoted to these methods.

5.3.3.1 Using an a Priori Count-Based MI Assumption

The best known algorithm using a count-based MI assumption is GMIL, which first appeared in [17]. GMIL stands for Generalized Multiple Instance Learning and is indeed a generalization of the standard MI assumption. The presence-based MI assumption of the Weidmann hierarchy is generalized by GMIL as well. However, it cannot represent learning problems obeying the threshold or counting based MI assumption, because the attributes constructed in the mapping are binary.

Like all algorithms using count-based assumptions, GMIL first identifies regions of the instance space that will be used in a second step to map bag attributes. Regions are identified systematically and exhaustively. All possible axis-parallel boxes in the instance space are explicitly enumerated. As an illustration, consider a discrete d-dimensional instance space $\mathbb{X} = \{1, \ldots, v\}^d$ in a two-class classification problem. In a one-dimensional space ($d = 1$), if the attribute has two possible values ($v = 2$), there are three possible axis-parallel boxes as shown in Fig. 5.3. If the space has two dimensions and each dimension can take one of two possible values, there are nine possible axis-parallel boxes as shown in Fig. 5.4. If the space has three dimensions, each with two possible values, there are 27 possible axis-parallel boxes as shown in Fig. 5.5. In general, there are $N = (v\,(v+1)\,/2)^d$ possible axis-parallel boxes in a d-dimensional space. The reason why the regions have axis-parallel box shapes is because the infinite norm is used to determine distances in the instance space. This norm defines the length of a d-dimensional vector x as $\|x\|_\infty = \max\,\{|x_1|, \ldots, |x_d|\}$, the largest absolute value of its components. GMIL creates two Boolean attributes for each box, indicating whether a bag contains an instance within that box. To reduce the number of attributes, boxes containing the same set of points are grouped together and only one representative box for each group is used.

Fig. 5.3 There are three axis-parallel boxes when $d = 1$

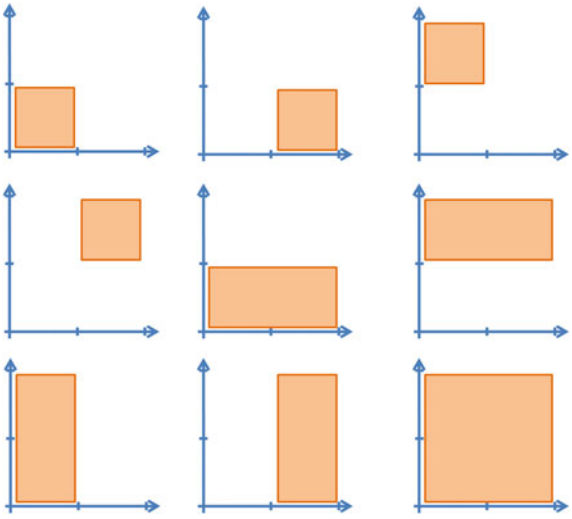


Fig. 5.4 There are nine axis-parallel boxes when $d = 2$

Concretely, GMIL maps a bag X to $M(X) = \langle a_1, \ldots, a_N, \overline{a}_1, \ldots, \overline{a}_N \rangle$. The algorithm sets a_i to 1 if any point of X is contained by the ith box and sets it to 0 otherwise. It sets $\overline{a}_i = 1 - a_i$, $\forall i \in [1, \ldots, N]$. All information is encoded by the N first attributes. This would be sufficient for many learning algorithms. However, GMIL was originally designed to learn monotone disjunctions using the Winnow classifier [15]. Since Winnow generates formulas generated only containing disjunctions of the input variables, the negations of the first N attributes must also be supplied such that any logical combination of the initial variables can be formed.

Once the bags have been mapped to Boolean attributes, the algorithm tries to learn the target concept using a specific MI assumption based on theoretical results from geometric pattern recognition [12]. In the standard MI assumption, a single positive instance inside a bag determines that the bag belongs to the positive concept. Instance labels are typically determined by the proximity of the instance to a single target point, but GMIL can represent more general concepts. It represents a concept by a set of target points, more specifically, a set of attraction points, which can be seen as instances from an ideal positive bag. GMIL can also include a set of repulsion points, which can be seen as instances from an ideal negative bag. In this setting, a bag is positive if and only if it is sufficiently close to attraction points and sufficiently far from repulsion points.

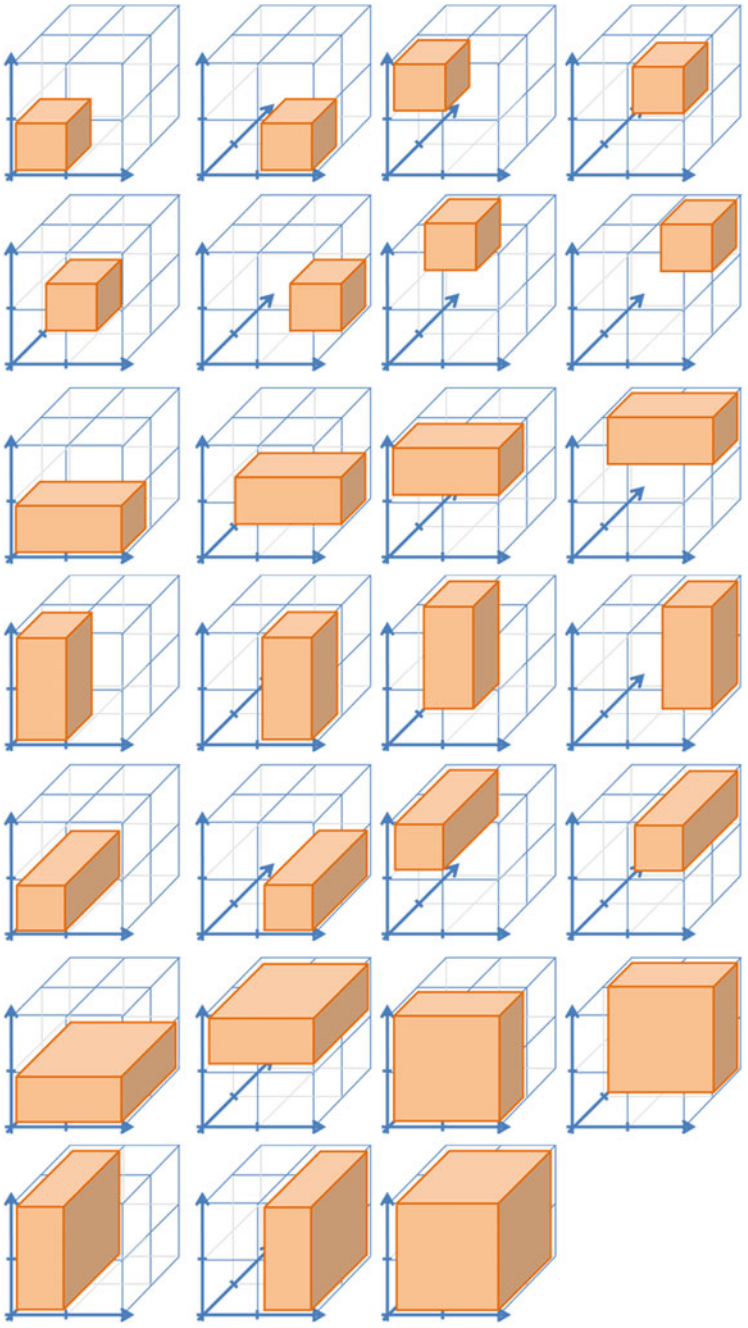


Fig. 5.5 There are 27 axis-parallel boxes when $d = 3$

GMIL's notion of distance between bags is based on the Hausdorff distance. Recall from Sect. 3.5 that the Hausdorff distance between two sets of points P and Q is defined as the largest distance from either a point in P to its nearest neighbor in Q or from a point in Q to its nearest neighbor in P. Due to its use of the max operator, the Hausdorff distance is sensitive to outlier points. To improve the robustness against noise, Scott et al. use the *ranked full-Hausdorff* distance

$$\max\left\{\max_{p\in P}^{s}\left\{\min_{q\in Q}\left\{\|p-q\|_\infty\right\}\right\}, \max_{q\in Q}^{s}\left\{\min_{p\in P}\left\{\|p-q\|_\infty\right\}\right\}\right\}, \tag{5.4}$$

in which instead of using the largest distance, the sth largest distance is used. In (5.4), $\max^s$ denotes the sth largest value, P represents the pattern and Q is the model. Positive bags are within a ranked full-Hausdorff distance of some threshold γ from the ideal positive bag and at least a ranked full-Hausdorff distance of γ' away from the ideal negative bag. Let $Q = \{q_1, \ldots, q_k\}$ be the set of attraction points and $\overline{Q} = \{\bar{q}_1, \ldots, \bar{q}_{k'}\}$ the set of repulsion points representing the target concept. The concept can be modeled as a set of k axis-parallel attraction boxes and a set of k' axis-parallel repulsion boxes. A bag is positive if and only if it contains points within at least $r = k - s$ of the k attraction boxes and contains points within at most s of the k' repulsion boxes.

The Winnow algorithm is used in [17] to implement the GMIL assumption. Winnow is a linear-threshold algorithm that learns r-of-k threshold functions. It assigns nonnegative real-valued weights w_a to each attribute a. Weights are iteratively modified to find a hyperplane

$$\sum_{i=1}^{N} a_i w_{a_i} + \overline{a}_i w_{\overline{a}_i} = \theta,$$

which separates both classes, where θ is the threshold determined by the algorithm. The $k + k'$ more weighted attributes are selected at the end of training. The values of the selected attributes correspond to the k attractions plus k' repulsion points identified by the algorithm. In the classification stage, a bag is labeled positive if $a_{i_1} + \cdots + a_{i_k} + \overline{a}_{i_1} + \cdots + \overline{a}_{i_{k'}} \geq r$.

Scott et al. also presented a GMIL variant using the *ranked half-Hausdorff distance*. Using this distance they assume that the model is accurate and compute the distance from the bag to the model, but not vice versa. According to this variant, positive bags are within a distance

$$\max_{q\in Q}^{s}\left\{\min_{p\in P}\left\{\|p-q\|_\infty\right\}\right\} \tag{5.5}$$

of some threshold γ to the ideal positive bag and, including repulsion points, beyond a distance

$$\min_{q\in Q}^{s'}\left\{\min_{p\in P}\left\{\|p-q\|_\infty\right\}\right\} \tag{5.6}$$

of another threshold γ' to the ideal negative bag. As before, the concept is a set of k axis-parallel attraction boxes and a set of k' axis-parallel repulsion boxes. A bag is positive if and only if it contains points within at least $r = k - s$ of the k attraction boxes and contains points within at most s' of the k' repulsion boxes. Note that, in contrast to the full-Hausdorff distance model, the number of points s which are tolerated to not fall in attraction boxes can be different to the number of points s' which are tolerated to fall in repulsion boxes. Though it was theorized that the half-Hausdorff variant should be more robust against noise and more able to avoid overfitting, empirical results show a higher generalization ability of the full-Hausdorff variant on data from several domains.

GMIL has a theoretically sound foundation. However, it is not a practical learning method, since it has a very high time complexity. In the sequence of Figs. 5.3, 5.4 and 5.5 it can be seen that the number of boxes grows exponentially as d increase. A real application, with a moderate number of attributes, like Musk, is unfeasible to be solved by GMIL. The strategy of using a reduced number of instances to build the learning model [18] fails because, when the dimension is not trivially small, in order to significantly reduce the computational cost, the number of instances must be so small that it becomes insufficient to build an accurate model. A kernel-based reformulation is another strategy used to improve the efficiency of GMIL. The kernel performs the feature mapping implicitly and allows a support vector machine to be applied directly to the data. However, computing the kernel on two bags requires counting the number of boxes that contain at least one instance from each of both bags, which again leads to severe scalability issues and quickly renders the problem intractable as the problem size increases. To address this issue, a fully polynomial randomized approximation scheme (FPRAS) was presented in [19], reducing the time complexity from exponential to polynomial.

5.3.3.2 Learning a Count-Based MI Assumption

In Chap. 4, we showed that instance-based methods make strong assumptions regarding the MI hypothesis. Each instance-based algorithm implements a specific MI assumption: some algorithms are based on the standard MI assumption, others on the collective assumption, and so on. The GMIL algorithm discussed in Sect. 5.3.3.1 has a MI assumption wired in its design as well. In these methods, the MI assumption is not only used in the classification stage to determine the bag label, but also in the training stage to impose restrictions to help determine the class likelihood of instances. If the imposed MI assumption does not conform reasonably well to a given dataset, then the algorithm cannot build an appropriate learning model for it. Each algorithm is only appropriate for those problems which conform to the applied MI assumption.

Unlike instance-based methods and methods like GMIL that have a specific MI assumption embedded in their design, mapping-based algorithms described in this section do not assume a priori the existence of a specific relationship between the

labels of each bag and of its instances. This relationship is learned from the data instead, in the form of a count-based MI assumption. Training occurs in two steps

1. The method tries to identify regions in the instance space using either supervised or unsupervised methods. These regions appear as a result of the instance space structure.
2. The underlying MI assumption is learned. This is the relationship between the bag labels and the instance space regions identified in the first step. To this end, a new representation space is built, in which each attribute corresponds to one of the regions. Each bag is mapped into this new space, such that the value of the ith attribute indicates the presence or frequency of instances of the bag in the ith region. Any single-instance learning model can be built on this new single-instance training set.

Methods in this category differ fundamentally in the way they identify instance patterns, i.e., in the first step described above. In the Two-Level Classification (TLC) algorithm [22], a standard decision tree is used for this purpose. The tree is built on all instances of all training bags. Each instance is assigned to its bag's class label. Instances are weighted such that all training bags have equal weight in the construction of the learning model. Each node in the tree represents a region of the instance space. In the second step, each bag is mapped to a new representation in which each attribute contains the number of instances of the bag that have reached the corresponding node in the tree.

Constructive Clustering Ensemble (CCE) [25] uses a clustering algorithm to determine the regions. The k-means algorithm is used to obtain a number of groups whose centers are stored. In the second step, each bag is mapped to a new representation in which each attribute indicates the presence of instances of the bag in the corresponding group. An instance belongs to a group g if its distance to the center of g is less than its distance to the center of the other groups. As it is not possible to determine the optimal number of groups in advance, CCE generates many classifiers, each obtained from a number of different groups, and then combines their predictions in a majority vote.

Since these algorithms do not make a priori assumptions about the nature of the relationship between bags and instances underlying the data, they can learn a wider variety of problems. For example, all algorithms based on the standard MI assumption take for granted that there are two classes of instances (positive and negative). Algorithms learning the MI assumption during training can find an arbitrary number of classes in the instance space and can discover relationships between bags and instances that best fit the training data.

5.3.4 Mapping Methods Based on Distance

In count-based mapping methods, the attribute values of each bag are defined by the location of instances of the bag inside a delimited region of instance space corre-

sponding to that attribute. The notion of an instance membership to a region is strict. It only accepts two extreme possibilities: the point either belongs or does not belong to the region, depending on which side of the border of the region the point is located. The fact that we can only have a vague idea of the borders of the instance regions is ignored. In many applications, perfectly delimited regions boundaries make no sense. For example, if tall people are an important region of the instance space, it is difficult to determine where we should start the region, at 1.70 m, 1.80 m, or 1.85 m? Any such value would be merely conventional. However, we can say that if an instance is near the center of a region we can have a great certainty that it falls within the region. The farther an instance is located from the center, the less likely it belongs to the region. This is the idea behind distance-based mapping methods: each attribute value in the output space is related to the distance from the bag to the center of a region.

These methods try to identify instance regions that are representative of the structure of the instance space. Regions can be obtained through a clustering or classification model constructed from training instances. A prototypical point is recorded at the center of each region. In some cases, prototypes of only one class (usually the positive class) are used. In other cases, they are determined for each class. Each attribute of the induced space corresponds to one of the prototypes found in the original space. The attribute value is a distance measure (or a similarity measure) between the bag and the prototype. Note that the bag contains many points (instances), while the prototype is a single point. Specific distance functions between bags and prototypes have to be used. Distance functions used in these cases are usually aggregations of distances between the instances of the bag and the prototype. Distance-based mapping methods differ in how instance prototypes are determined and in their definition of the distance function.

One of the first algorithms using this type of mapping was DD-SVM [5]. This algorithm selects instance prototypes for both classes based on the values of the diverse density (DD) function. Under the diverse density framework, a prototype for class C is a point of the instance space with a high probability of being found in bags of class C. Prototypes are local extrema of the DD function, where the positive prototypes are maxima and the negative prototypes minima. To locate the prototypes, gradient descent methods are used over the DD function. To find the positive prototypes, optimization processes are started from each instance of the positive bags, while for negative prototypes, searches start from each instance of the negative bags. The located prototypes are used to map each bag to the new representation space. Using T prototypes, a bag X is transformed as

$$\mathscr{M}(X) = \langle S(t_1, X), \ldots S(t_T, X) \rangle, \tag{5.7}$$

where t_i represents the ith prototype and $S(t_i, X)$ is a distance measure between the bag X and t_i. Specifically, in [5] an absolute distance measure $S(t, X) = \min_j \left\| x_j - t \right\|$ is used. The authors apply an SVM to the bags represented in the mapped space to obtain a bag classification model. In general, as with all mapping

methods, any single-instance learning algorithm can be used to build this model as well.

The MILES algorithm [6] was introduced by the same authors as DD-SVM. Instead of looking for class prototypes in each bag, MILES uses all training instances as reference points to construct the new bag space. In other words, each instance is treated as a prototype. The new representation space has as many attributes as the total number of instances in the training set. More formally, let $\mathbf{X} = \{X_1, \ldots X_m\}$ be the training bag set. We align the instances inside the bags and renumber them to get the set of instances $\{t_k | \exists X_i \in \mathbf{X} : t_k \in X_i\}$, $k = 1, \ldots, T$, where $T = \sum_{i=1}^{m} n_i$. We use (5.7) to map a bag X to the output space. To calculate the value of the ith attribute, MILES uses the Gaussian similarity function given by

$$S(t, X) = \max_j \exp\left(-\frac{\|x_j - t\|^2}{\sigma^2}\right), \tag{5.8}$$

where σ is a parameter to scale the attributes.

MILES is more computationally efficient than DD-SVM, because it avoids the expensive optimization procedure over the diverse density function, which DD-SVM must perform for every instance. Chen et al. [6] have shown that MILES is as good as and sometimes superior to DD-SVM in generalization accuracy and it is also more robust with respect to label noise.

The MILES mapping can be seen as a method for determining the weight of each instance. Indeed, the SVM applied to the mapping space calculates a weight for each attribute which is normally used for feature selection. The attributes of the mapped space are precisely the instances of the training bags, which allows to determine the influence of different parts of the instance space. However, MILES does not create a well-defined weight function over the instance space, because the max operator, used in (5.8) that determines the value of each attribute, only takes into account the influence of the nearest instance of the bag to the target point, resulting in a bag-dependent weight function [9]. Foulds et al. [9, 10] proposed the YARDS algorithm, which is similar to MILES in almost everything except in that YARDS can find a true weight function over the instance space. By replacing the max operator with the sum operator, that is, by setting

$$S(t, X) = \sum_j \exp\left(-\frac{\|x_j - t\|^2}{\sigma^2}\right),$$

the bag-dependence in the similarity function is removed. In YARDS, each instance of the bag has an influence on the bag-level classification and that influence only depends on the attributes of the instance and not on the rest of the bag.

5.3.5 *Bag-Level Distance Mapping Methods*

In the mapping methods included in Sects. 5.3.3 and 5.3.4, bags are described by their relationships with instance-level spatial structures. This mapping relates instance space regions with bag classes. Another way to transform a multi-instance problem into a single-instance one is by describing each bag through the spatial relationship it has with the other bags of the training set. In this case, the mapping is done at the bag level, but instance space regions are ultimately related to bag classes, since bags are represented by multiple vectors in the instance space. However, in this mapping each instance maintains the relationship with its bag, making it a more informative mapping than that which only includes instance-level relations.

The idea of bag-level distance mapping methods has been developed by Zhang and Zhou [26] with their BARTMIP algorithm. The work scheme of BARTMIP is shown in Fig. 5.6. A multi-instance clustering model is built on the training bags, dividing them in k groups. Each group is represented by its medoid, i.e., the most central bag. Each bag is mapped to a vector of k attributes, one for each group of bags. The ith attribute value of a bag is the distance from the bag to the ith medoid. All training bags are mapped with this form of representation. It results in a single-instance training set on which a single-instance classification algorithm is trained. In the prediction step, the new bag is mapped in the same way to a vector of k attributes and processed by the single-instance classification model.

The components of this algorithm can be selected from a wide variety of choices. BATRMIP can train any single-instance classification algorithm and use any multi-instance clustering algorithm. Multi-instance clustering algorithms are described in Chap. 7. Specifically, in [26], BARTMIP uses a multi-instance clustering algorithm called BAMIC (Sect. 7.1.4.1), which is an adaptation of the single-instance k-medoids clustering algorithm to the multi-instance setting. Many multi-instance clustering methods depend on a bag-level distance function which in turn uses an instance level distance function. Distance functions at bag and instance levels are other components of the model that should be chosen. The optimal number of groups to be generated in the clustering step can be determined by cross-validation. An alternative is to build several clustering models, each with a different number of groups, and train a classifier model from each grouping. The ensemble prediction is obtained by majority vote.

5.4 Experimental Analysis

In this section, we empirically compare the performance of some representative bag-based MIC methods. We show experimental results for both original-BS methods and mapped-BS methods and compare the two strategies. These experiments are only intended for illustration purposes and cannot be taken as a rigorous comparison

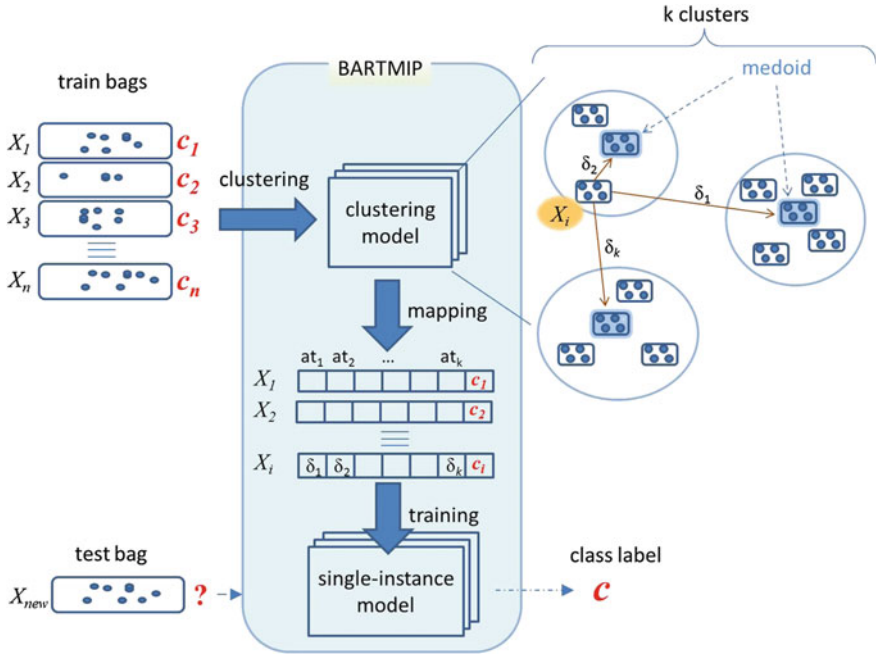


Fig. 5.6 BARTMIP algorithm

among classifiers. The experimental setup is specified in Sect. 5.4.1, while Sect. 5.4.2 presents the results.

5.4.1 Setup

We use the same datasets as in the experimental study of Chap. 4, described in Table 4.1. The algorithms included in the study are named in the first column of Table 5.1. The second column describes the method type. CitationKNN and MISMO are representative algorithms that work on the original bag space. The other algorithms are mapping methods, one of each type described in Sect. 5.3 with the exception of prototype concatenation discussed in Sect. 5.3.2. Prototype concatenation mapping methods have been excluded due to their high memory requirements. They are appropriate to use in small problems, but even for medium-sized datasets (as some are in these experiments) it is difficult to make comparative studies.

Unlike methods that work on the original bag space and construct an specific classifier, a mapping method can train any standard classification algorithm. Their performance depends on both the mapping method and the learner used as base classifier. In order to get a better idea of the mapping method qualities, we try each

Table 5.1 Bag-based classification algorithms to be compared

Algorithm	Category
CitationKNN	Original-BS distance-based methods
MISMO	Bag-level kernel methods
SimpleMI	Bag statistic mapping methods
MILES	Distance-based mapping methods
CCE	Count-based mapping methods
BARTMIP	Bag-level distance mapping methods

alternative with five popular classification algorithms: one nearest neighbor (1NN), C4.5, logistic regression (LR), an SVM and AdaBoost with C4.5 as base classifier (AdaBoost). We use Weka implementations for algorithms in the first four rows of Table 5.1, while the last two were implemented by us. A rough optimization was made for the most important parameters of each method looking for those yielding the best result across all the datasets. We use default parameter settings for each algorithm if not specified otherwise. We use the fivefold cross-validation procedure and evaluate the performance of the classifiers by means of their accuracy (Sect. 1.4).

5.4.2 Results and Discussion

In Sect. 5.4.2.1, we show empirical results of the selected original-BS methods. We compare typical mapped-BS methods with each another using several base classifiers in Sect. 5.4.2.2. Finally, we compare the original-BS and mapped-BS based classifiers in Sect. 5.4.2.3.

5.4.2.1 Original-BS Methods

Table 5.2 presents the experimental results of two original-BS methods, namely CitationKNN and a bag-level SVM. The latter is a standard SVM using the Gärtner et al. MI kernel described in Sect. 5.2.2 with the standard RBF instance-level kernel. The table lists the best results for each algorithm after a simple parameter adjustment was done looking for the best average result over all data. The results shown for CitationKNN were obtained with $C = 2$ and $R = 2$ and those for SVM with $C = 1.0$ and $\gamma = 0.5$. The last two rows of the table show the average accuracy and the standard deviation of each classifier over the nine datasets. The best accuracy is highlighted in bold for each dataset. SVM is the winner in six out of nine cases, while CitationKNN

Table 5.2 Classification accuracy for methods working in the original bag space

Dataset	CitationKNN	SVM
Musk1	**89.13**	88.04
Musk2	**84.16**	83.17
Atoms	70.21	**74.47**
Bonds	74.47	**85.11**
Chains	73.40	**84.57**
WIR	64.60	**69.91**
TREC	47.75	**74.75**
Beach	**82.00**	80.50
Fox	50.00	**61.00**
Average	70.64	**77.95**
SD	14.42	**8.70**

wins in three datasets. The higher average accuracy of SVM supports the idea that it has a significantly better performance than CitationKNN over the studied problem domains. The lower standard deviation of the SVM means that its good performance is more evenly distributed across all datasets than that of the CitationKNN, which instead obtains very good results in a few datasets, but poor results in many of them.

5.4.2.2 Mapped-BS Methods

Table 5.3 presents a summary of the experimental results of the selected mapped-BS methods using five base classifiers. The average accuracy computed over the nine datasets along with the confidence interval with a significance level $\alpha = 0.05$ is shown for each pair of mapping method and classifier. The algorithm parameters were set as follows: $\sigma = 250$ in MILES, 60 % of clustering in BARTMIP, five iterations in CCE and 10 iterations in Adaboost. The SVM in all mapping methods uses an RBF kernel with $C = 10.0$ and $\gamma = 0.5$. The most accurate mapped-BS method for each base classifier is highlighted in bold. SimpleMI obtains the best performance for three classifiers: C4.5, SVM, and Boosting. BARTMIP is the best performing mapped-BS method for the 1NN and LogReg classifiers. This suggests that SimpleMI and BARTMIP are two of the most accurate mapped-BS methods overall, since they achieve the highest quality predictions with several base classifiers over a range of datasets from different application domains.

Table 5.4 presents the detailed experimental results of each mapped-BS method executed with its best base classifier following the conclusions of Table 5.3. The highest accuracy for each dataset among the four methods is marked in bold. SimpleMI and BARTMIP are again the most outstanding algorithms, as each one wins in four datasets. With respect to the application domains, it seems that SimpleMI is best suited for molecular activity prediction, while BARTMIP looks like the leader

Table 5.3 Average classification accuracy of mapping methods using different base classifiers

Base classifier	SimpleMI	MILES	BARTMIP	CCE
1NN	74.99 ± 6.32	72.47 ± 5.77	75.24 ± 8.21	70.55 ± 6.79
C4.5	76.73 ± 5.41	68.54 ± 8.04	73.42 ± 6.17	67.19 ± 5.55
LogReg	70.07 ± 5.75	69.81 ± 4.34	78.59 ± 5.75	74.00 ± 5.54
SVM	79.18 ± 6.81	69.09 ± 7.84	78.58 ± 8.47	70.40 ± 6.31
Boosting	76.77 ± 5.19	70.16 ± 8.70	74.81 ± 6.78	69.32 ± 5.19

Table 5.4 Classification accuracy for best performing mapping method schemes

Dataset	SimpleMI	MILES	BARTMIP	CCE
Musk1	91.30	77.17	84.78	80.43
Musk2	91.09	66.34	85.15	74.26
Atoms	73.40	80.85	84.04	78.19
Bonds	84.57	79.79	82.45	79.26
Chains	85.64	79.79	86.17	77.66
WIR	62.83	61.06	63.72	73.45
TREC	75.75	68.25	71.50	68.25
Beach	82.50	80.00	83.00	80.50
Fox	65.50	59.00	66.50	54.00

in the image recognition domain. In the next section, we delve deeper into this topic when we compare all bag-based methods to each other.

5.4.2.3 Overall Comparison

In Sects. 5.4.2.1 and 5.4.2.2, we pointed out the most accurate classifiers of each type. We are now interested to make an overall comparison between original-BS and mapped-BS methods in order to discover their advantages and disadvantages. The best performing model of each type is taken into account in this comparison. Two original-BS methods and four mapped-BS methods are included.

To discover which method is the best option in each case, we first separate the results by application domain. In Fig. 5.7, we depict the accuracy of the methods on the biochemical applications. Note that the accuracy axis values start at 40 to better distinguish the differences between the methods. SimpleMI, BARTMIP, and MISMO dominate in almost all datasets, while CitationKNN and MILES are not stable in their results. It is remarkable that BARTMIP performs quite good in the five datasets.

In Figs. 5.8 and 5.9, we show the accuracy of the methods on datasets from the textual and image domain, respectively. From these charts, we can not identify one algorithm that is superior to the others in any of these domains. The advantage

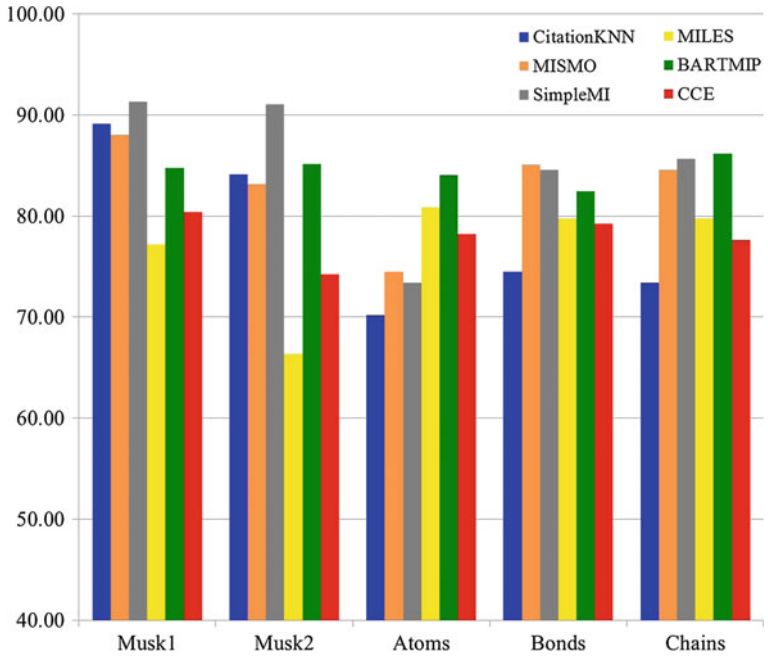
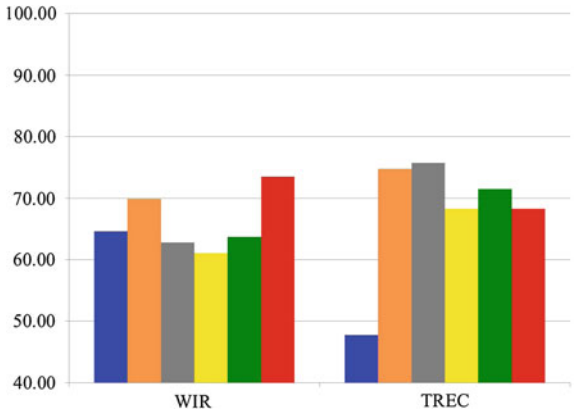


Fig. 5.7 Biochemistry domain

Fig. 5.8 Textual domain



(discussed in the previous section) of BARTMIP over SimpleMI on image datasets is negligible. We can only point out some general trends. CitationKNN and MILES have again poorer results compared to the other methods. SimpleMI, BARTMIP and MISMO excel in most datasets. Figure 5.10 shows the average accuracy of the six methods over the nine datasets and supports the above statement.

We are also interested in analyzing the training time of the models. Figure 5.11 shows the average training time of the six methods over the nine datasets. Note that

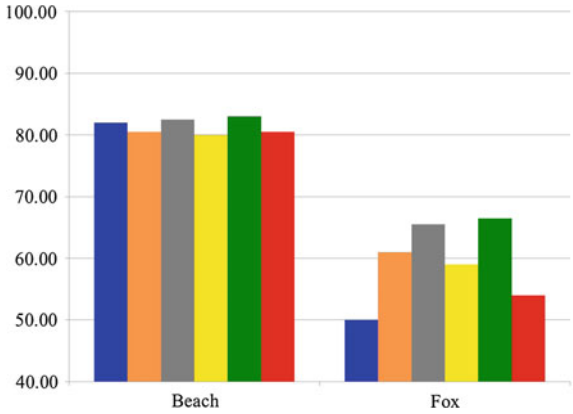


Fig. 5.9 Imaging domain

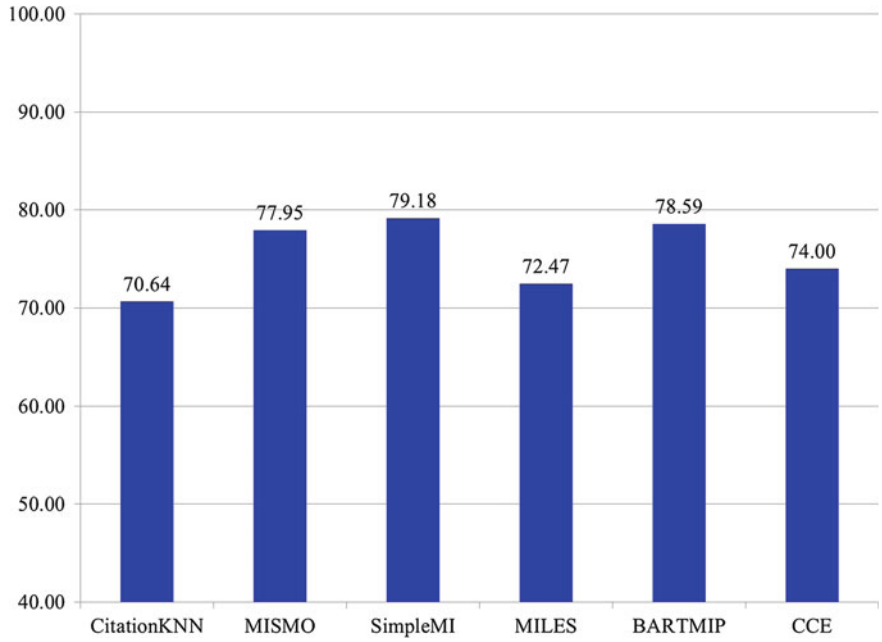


Fig. 5.10 Average accuracy of selected bag-based classification methods over the experimental datasets

a logarithmic scale is used to represent time intervals, such that differences between methods can be correctly perceived. Time values are given in seconds, but we are mostly interested in the relative time proportions of the different models. The training

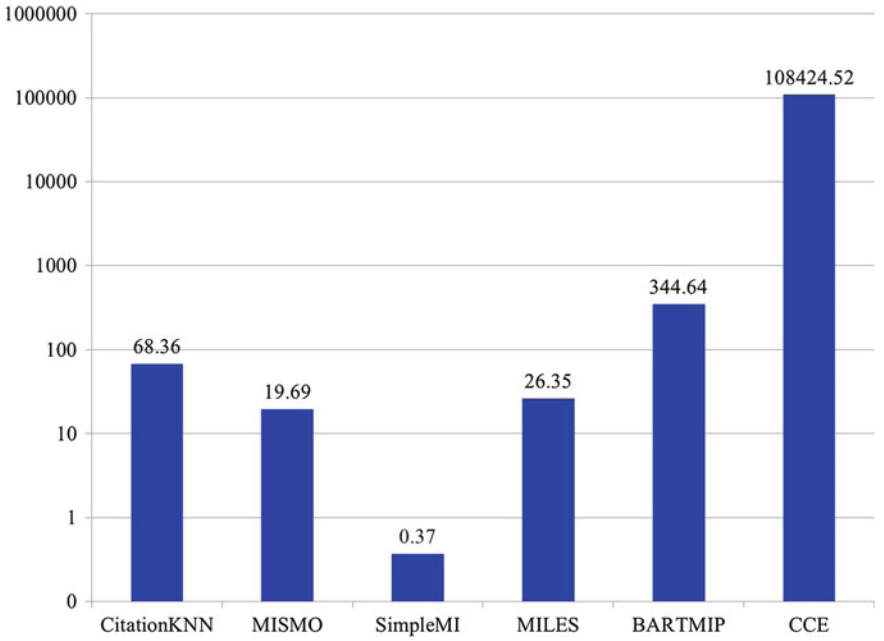


Fig. 5.11 Average training time of selected bag-based classification methods over the experimental datasets

time of CitationKNN is mainly devoted to the calculation of bag-level distances,[1] whereas kernel calculations made by MISMO are three times faster than the work of CitationKNN. The very small training time of SimpleMI is one of the most remarkable things in the figure. The key lies in the simplicity of its mapping method. MILES has a fair training time complexity, which is in line with its moderately simple mapping method. Conversely, BARTMIP training has a considerable time complexity. This method has a much more complex mapping method, that includes bag-level distance calculations and bag-level clustering. Finally, training CCE takes a long time. It includes instance-level distance calculations and instance-level clustering, that are much more time demanding than their bag-level relatives.

5.5 Comparing Instance-Based, Bag-Based, and Traditional Classification Methods

In Chap. 4 and this chapter, we discussed two classifier families that work very differently: one of them learns at the instance level, the other at the bag-level. In both

[1] The implementation used for CitationKNN calculates the neighbor list of each bag in the training step.

cases, we have presented comparative experiments on the performance of several representative members of each family. An obvious question is how these two families compare. This does not have an easy answer and has been the subject of study of some recent work [1, 2]. There is another more basic question that some researchers have put forward [1, 16], namely whether multi-instance classifiers outperform single-instance classifiers in all multi-instance datasets.

Ray and Craven [16] found that single-instance classification algorithms can perform well for several MIL problems, outperforming MI classifiers in some cases. This strongly impacts the MIL community, as occasional reports have shown that MI classifiers with good success records were beaten by simple single-instance models in some datasets.

Alpaydin et al. [1] designed artificial datasets with increasing complexity levels, corresponding to more and more complex dependencies between instances in a bag. They compare instance-based, bag-based, and single-instance classifiers on artificial datasets of different sizes and levels. Their conclusion was that, in general, single-instance classifiers can only handle the simplest MIL problems corresponding to the lowest complexity level, instance-based classifiers are good to solve problems from the first and second complexity levels and bag-based classifiers can solve problems from the first three levels. Datasets from the fourth complexity level require even more advanced classification methods. Alpaydin et al. also found that datasets where single-instance classifiers outperform multi-instance methods are those with the lowest complexity level and with a small number of bags, because there is not enough data to train the bag-level classifiers.

This explanation clarifies the general relation that appears between algorithms and data complexity. Nevertheless, we should keep in mind that no classifier exists that can handle all different application domains. Faced with a new MIL problem, the best algorithm might be an instance-based, a bag-based or a traditional classifier.

5.6 Summarizing Comments

Bag-based classification algorithms are an important group of MIC methods. They predict the bag class mainly using information at the bag level. They do not strive to predict instance class labels and have more flexible and generals MI assumptions. Several bag-based methods have appeared in the literature. According to their main features, we organize them in a category system depicted in Fig. 5.12. There are two principal categories of bag-based classifiers: (i) methods that operate on the original bag space by relying on a distance, similarity or kernel function and (ii) methods that use a mapping function to transform the data to a single-instance representation, such that single-instance classifiers can be trained and used to predict bag labels. Several types of transformations have been developed. Some mapping functions are based on simple bag statistics. Others represent the new space by concatenating prototypes extracted from the training bags. Other mapping methods count the number

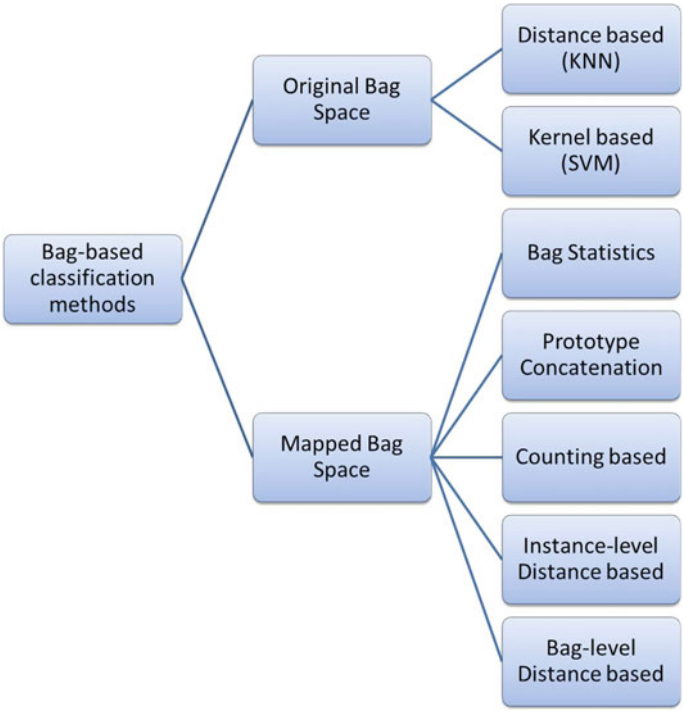


Fig. 5.12 Bag-based methods hierarchy

of instance of the bag falling in specific regions of the instance space and yet others compute the distance from the bag to the centers of these regions.

The experimental study shows that each of the discussed methods can attain high accuracy in some application domains. Nevertheless, we do not recommend CCE because of its large training time and uncertain performance. We advise the use of SimpleMI, because it often attains a very good accuracy and is very fast to train. BARTMIP is also a good option, because of its stable performance over several domains.

References

1. Alpaydin, E., Cheplygina, V., Loog, M., Tax, D.M.: Single-versus multiple-instance classification. Pattern Recognit. **48**, 2831–2838 (2015)
2. Amores, J.: Multiple instance classification: review, taxonomy and comparative study. Artif. Intell. **201**, 81–105 (2013)
3. Blaschko, M., Hofmann, T.: Conformal multi-instance kernels. In: Schölkopf, B., Platt, J.C., Hoffman. T. (eds.) Proceedings of the 19th Conference on Advances in Neural Information Processing Systems (NIPS 2006), pp. 1–6. MIT Press, Cambridge (2006)

4. Boughorbel, S., Tarel, J.P., Boujemaa, N.: The intermediate matching kernel for image local features. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2005), pp. 889–894. IEEE, Los Alamitos (2005)
5. Chen, Y., Wang, J.: Image categorization by learning and reasoning with regions. J. Mach. Learn. Res. **5**, 913–939 (2004)
6. Chen, Y., Bi, J., Wang, J.: MILES: multiple-instance learning via embedded instance selection. IEEE Trans. Pattern Anal. **28**, 1931–1947 (2006)
7. Dong, L.: A Comparison of Multi-instance Learning Algorithms. Master thesis, The University of Waikato, New Zealand (2006)
8. Dooly, D.R., Zhang, Q., Goldman, S.A., Amar, R.A.: Multiple-instance learning of real-valued data. J. Mach. Learn. Res. **3**, 651–678 (2002)
9. Foulds, J.: Learning instance weights in multi-instance learning. Master thesis, The University of Waikato, New Zealand (2008)
10. Foulds, J., Frank, E.: A review of multi-instance learning assumptions. Knowl. Eng. Rev. **25**(1), 1–25 (2010)
11. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.: Multi-instance kernels. In: Sammut, C., Hoffmann, A. (eds.) Proceedings of the 19th International Conference on Machine Learning (ICML 2002), pp. 179–186. Morgan Kaufmann Publishers, San Francisco (2002)
12. Goldman, S.A., Kwek, S.S., Scott, S.D.: Agnostic learning of geometric patterns. J. Comput. Syst. Sci. **62**(1), 123–151 (2001)
13. Henegar, C., Clément, K., Zucker, J.D.: Unsupervised multiple-instance learning for functional profiling of genomic data. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) Proceedings of the 16th European Conference on Machine Learning (ECML 2006), pp. 186–197. Springer, Berlin (2006)
14. Kwok, J., Cheung, P.: Marginalized multi-instance kernels. In: López, R., Veloso, M.M. (eds.) Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJACI 2007), pp. 901–906. IJCAI/AAAI Press, Hyderabad (2007)
15. Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Mach. Learn. **2**(4), 285–318 (1988)
16. Ray, S., Craven, M.: Supervised versus multiple instance learning: an empirical comparison. In: De Raedt, L., Wrobel, S. (eds.) Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 697–704. ACM, New York (2005)
17. Scott, S., Zhang, J., Brown, J.: On generalized multiple-instance learning. Int. J. Comput. Int. Sys. **5**(1), 21–35 (2005)
18. Tao, Q., Scott, S.: A faster algorithm for generalized multiple-instance learning. In: Valerie, B., Zdravko, M. (eds.) Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004), pp. 550–555. AAAI Press, Menlo Park (2004)
19. Tao, Q., Scott, S.D., Vinodchandran, N.V., Osugi, T.T., Mueller, B.: Kernels for generalized multiple-instance learning. IEEE T Pattern Anal. **30**(12), 2084–2098 (2008)
20. Wang, J., Zucker, J.: Solving the multiple-instance problem: a lazy learning approach. In: Langley, P. (ed.) Proceedings of the 17th International Conference on Machine Learning (ICML 2000), pp. 1119–1126. Morgan Kaufmann Publishers, San Francisco (2000)
21. Wei, H., Yu, W.: Text representation and classification based on multi-instance learning. In: Lan, H., Yang, Y.-H. (eds.) Proceedings of the 10th International Conference on Management Science and Engineering Management (ICMSE 2009), pp. 34–39. IEEE, Los Alamitos (2009)
22. Weidmann, N., Frank, E., Pfahringer, B.: A two-level learning method for generalized multi-instance problems. In: Lavrac, N., Gamberger, D., Blockeel, H., Todorovski, L. (eds.) Proceedings of the 14th European Conference on Machine Learning (ECML 2003), pp. 468–479. Springer, Heidelberg (2006)
23. Yang, W., Gao, Y., Cao, L.: TRASMIL: a local anomaly detection framework based on trajectory segmentation and multi-instance learning. Comput. Vis. Image Underst. **117**(10), 1273–1286 (2013)
24. Zhang, M.: A K-nearest neighbor based multi-instance multi-label learning algorithm. In: Grégoire, E. (ed.) Proceedings of the 22nd International Conference on Tools with Artificial Intelligence (ICTAI 2010), pp. 207–212. IEEE, Los Alamitos (2010)

25. Zhou, Z., Zhang, M.: Solving multi-instance problems with classifier ensemble based on constructive clustering. Knowl. Inf. Syst. **11**(2), 155–170 (2007)
26. Zhang, M., Zhou, Z.: Multi-instance clustering with applications to multi-instance prediction. Appl. Intell. **31**(1), 47–68 (2009)
27. Zhou, Z., Jiang, K., Li, M.: Multi-instance learning based web mining. Appl. Intell. **22**, 135–147 (2005)