Chapter 8
Data Reduction

Abstract An increase in dataset dimensionality and size implies a large computational complexity and possible estimation errors. In this context, data reduction methods try to construct a new and more compact data subset. This subset should maintain the most representative information and remove redundant, irrelevant, and/or noisy information. The inherent uncertainty of MIL renders the data reduction process more difficult. Each positive bag is composed of several instances, of which only a part approximate the positive concept. Information on which instances are positive is not available. In this chapter, we first provide an introduction to data reduction. Next, two main strategies to reduce MIL data are considered. Section 8.2 describes the main concepts of feature selection as well as methods that try to reduce the number of features in MIL problems. Section 8.3 considers bag prototype selection and analyzes the corresponding multi-instance methods.

8.1 Introduction

Data reduction is a complex problem in any machine learning framework. Different techniques can be applied to obtain a reduced representation of the data, closely maintaining the original integrity. The reduced dataset should be more efficient to process and produce the same or similar analytical results.

As the dimensionality of a problem (number of features) increases, the well-known *curse of dimensionality* manifests itself. This concept was introduced by Bellman [1] to show that the number of samples required to estimate a function with a given level of accuracy grows exponentially with the problem dimension. For a given sample size, there is a maximum number of features above which the performance of an algorithm will degrade rather than improve. In fact, many data mining algorithms fail when the dimensionality is high, since data points become sparse and far apart from each other.

The large amount of data produced in any current application has resulted in datasets composed of thousands of objects represented by hundreds or even thousands of features. The dimensionality is a serious obstacle for the efficiency of most algorithms. Its reduction has become a necessary hot topic to ensure that algorithms

F. Herrera et al., *Multiple Instance Learning*, DOI 10.1007/978-3-319-47759-6_8

can perform appropriately. Data reduction techniques have been studied extensively in many domains. Section 1.2 gave a brief introduction to data reduction, grouping these methods as follows:

- **Feature selection**: the dimensionality is reduced by removing irrelevant or redundant features [17]. The goal of feature selection is to find a minimum set of attributes, such that the resulting probability distribution of the output is as close as possible to the original distribution using all features. These methods facilitate the understanding of the extracted knowledge and increase the speed of the learning stage.
- **Instance selection**: the number of observations is reduced by removing less relevant or noisy instances [16]. In MIL, these methods can be divided into two categories, since data samples are bags composed of one or more instances. On the one hand, *bag prototype selection* aims to reduce the number of training samples. The removal of a bag implies the elimination of all instances contained in it. On the other hand, *instance prototype selection* reduces the number of instances inside each bag. The number of bags remains the same, while less representative instances within them are eliminated. Both methods reduce the size of the data and therefore the computational complexity.
- **Feature extraction**: this is an extension of feature selection that allows the modification of the internal values representing each attribute [15]. In feature extraction, apart from the removal of attributes, feature subsets can be merged or can contribute to the creation of artificial substitute features.
- **Instance generation**: this process extends instance selection by allowing the modification of samples [22]. These methods create or adjust artificial substitute examples that could better represent the decision boundaries in supervised learning.

In this chapter, data reduction in MIL is considered. We describe the two most common techniques, feature selection, and bag prototype selection. All included concepts and methods show the interaction of data reduction with classification.

8.2 Multiple Instance Methods for Feature Selection

Datasets may contain many irrelevant or redundant features. To give an example, if we focus on the identification of a particular illness in several patients, attributes such as the patient's telephone number or surname are likely to be irrelevant, unlike attributes such as age or blood pressure. Irrelevant features imply a huge amount of data that results in an increase in processing time of a learner and a reduction of its performance. Manually evaluating the features (e.g., by a domain expert) becomes intractable when the number of attributes is high and the data behavior is not well-known. Automatic feature selection methods have been extensively used to replace the original data volume by an alternative, more reduced representation.

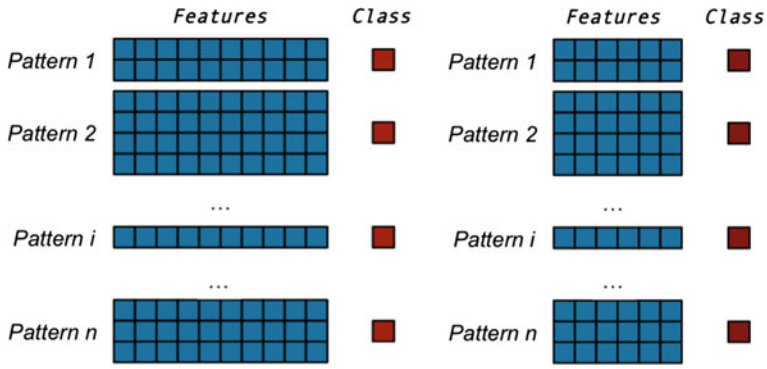
In this section, we first give an introduction to feature selection, providing a brief background and the main taxonomy. We analyze the particularities of multi-instance feature selection and discuss the representative methods.

8.2.1 Introduction to Feature Selection

Feature Selection (FS) methods select a subset of features from the initial dataset according to certain criteria. These methods should remove noisy and irrelevant features and maintain only the most informative ones. In this way, the feature subset should be capable of producing results equal to or better than the full set. Reducing the feature set has several benefits, such as minimizing the computational cost of algorithms, improving the accuracy of the final result, and making the data mining results easier to understand.

FS can be defined as a search problem to find a subset of features optimizing a particular criterion. Formally, let A be the original feature set with cardinality M and B the subset of desired features ($B \subset A$) with cardinality m ($m << M$) [10]. FS tries to find an optimal subset B that minimizes the criterion function $\mathscr{F}$. The main steps of FS methods are the following [10]:

1. **The generation of different subsets**: the feature space is explored for the best subset.
2. **The evaluation of feature subsets**: an evaluation function is used to test the fitness of a feature subset. This function corresponds to the criteria that the FS method tries to fulfill.
3. **The stopping criterion**: a stopping criterion to halt the search needs to be specified.



(a) Initial dataset (10 features) (b) Reduced dataset (6 features)

Fig. 8.1 MI feature selection method

Figure 8.1 shows the process of feature selection in a MIL context. Figure 8.1a depicts the initial dataset, where the samples are composed of several instances, that is, several feature vectors. Figure 8.1b describes the dataset after application of FS, where the number of features describing each instance has been reduced. FS in MIL is more challenging than its counterpart in single-instance learning. In MIL, each observation is represented by a number of instances. Instance labels are not available and it is possible to find both positive and negative instances inside a positive bag. Different methods assume that positive bags consist of mainly positive instances. The negative instances in positive bags may limit the discriminative power of FS methods. As a consequence, determining the significance of each attribute becomes more difficult.

Before describing the most relevant proposals of FS in MIL, we specify the FS taxonomy. This categorization is probably the most known and employed in FS methods over the years [18]. It is based on how the methods combine the FS search with the construction of the classification model:

- **Filter methods**: features are selected based on a performance measure independent of the classification algorithm. In this case, FS is an independent preprocessing step before the application of a particular classifier. The optimal feature subset or a relevance ranking of the features is returned. The advantages are that these methods are computationally fast, scalable, and independent of the classifier. On the other hand, their main shortcomings are the fact that they ignore feature dependencies and interactions with the classifier.
- **Wrapper methods**: a learner is used to measure the quality of feature subsets without incorporating any information about the specific structure of the classification or regression function. These methods can be combined with any learner.
- **Embedded methods**: FS is performed during the modeling phase of the classifier. The FS method is embedded in the classifier. These methods have the advantage that they include the interaction with the classification model. They are often far less computationally intensive than wrapper methods. They directly return the final classifier.
- **Hybrid methods**: these algorithms combine the best properties of filter and wrapper methods. First, a filter method is used to reduce the dimension of the feature space and obtain several candidate feature subsets. Next, a wrapper is employed to find the best candidate subset among them.

FS has become an apparent need in many learning paradigms. Numerous studies show that the reduction can not only reduce computational complexity, but also improve validation results and enhance semantic interpretability. Below, we describe the most relevant FS methods in MIL. Table 8.1 provides an overview of the main contributions in this area grouped as filter, wrapper, embedded, and hybrid methods.

Table 8.1 Features selection methods in multi-instance learning

Filter methods	
ReliefF-MI algorithm [28]	
Reliability-based algorithm [9]	
Embedded methods	
Boosting based methods	*Kernel based methods*
MI-AdaBoost algorithm [27]	MILES algorithm [5]
MCMI-AdaBoost algorithm [32]	Ngiam et al. [20]
EBMIL algorithm [25]	MIO algorithm [14]
BEL algorithm [31]	MIL-MFS algorithm [11]
Online MIL adaboost algorithm [3]	FSPO algorithm [19]
Hybrid methods	
HyDR-MI algorithm [29]	

8.2.2 Filter Methods

To select a feature subset, these algorithms take advantage of general characteristics of the data, like distances or statistical dependencies between classes. They are faster than other approaches, because they act independently of the induction algorithm. However, they tend to select subsets with a high number of features. It is also difficult to fix an internal threshold above which a feature is important enough to be selected.

Two multi-instance FS methods can be listed in this group. ReliefF-MI [28] is based on the principles on the single-instance ReliefF algorithm [21]. The second included proposal describes a multi-instance FS algorithm based on information aggregation using a data reliability measure. Both methods assign a weight to each feature to determine its relevance and specify a threshold on these weights to determine the final feature subset. As any filter method, they can be used as a preprocessing step before the application of any classifier.

8.2.2.1 ReliefF-MI Algorithm

This method was proposed by Zafra et al. [28] and estimates the quality of features based on how well their values distinguish between bags that are near each other. A description of its main steps is given in Algorithm 11. First, bags are randomly selected from the training data. For each sampled bag R, its k nearest neighbors from the same class (nearest hits) are found as well as its k nearest neighbors of the opposite classes (nearest misses). Based on these neighbors, the weight of each feature is updated. These weights reflect the ability to distinguish class labels. A high weight indicates that this feature differs among bags from different classes and is the same in bags of the same class. Features are ranked by weight and those that exceed a user-specified threshold are selected to form the final subset. The calculation of

the nearest neighbors and the definition of the $diff_{bag}$ function applied to the bags is carried out with different variants of Hausdorff distance. The authors of [28] also proposed the adapted Hausdorff distance (Sect. 3.5).

Experimental Study

The study of Zafra et al. [28] considers five benchmark real world datasets: Musk1, Musk2, Elephant, Tiger, and Fox. A description of any of these is given in Sect. 3.6. ReliefF-MI provides a reduced dataset in a preprocessing step independent of a classifier. To show the efficiency of the method, the reduced dataset is used in 17 multi-instance classification algorithms. The results are evaluated on both accuracy and execution time. They confirm the utility and efficiency of ReliefF-MI as a preprocessing step for all included algorithms. The classifiers statistically improve both their accuracy and execution time.

The experimental study also includes a comparison of different distance measures used by ReliefF-MI, namely the maximal, minimal, average, and adapted Hausdorff distances (Sect. 3.5). The results show that the adapted Hausdorff distance performs statistically best, obtaining a better dimensionality reduction, and better classifier accuracy. Its advantage lies with the different ways to measure the distance depending on the specific information available in each bag.

Algorithm 11 ReliefF-MI algorithm

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_N\}$ $(X_i \subseteq \mathbb{X})$
Input: $m \leftarrow$ number of times that the process is repeated
Input: $k \leftarrow$ number of nearest examples of the same and different class considered in the process.
Input: $\varepsilon \leftarrow$ threshold to determine if the feature is added to subset
Output: $selectedSubset$, the selected feature subset

$selectedSubset = \emptyset$
W = 0 ▷ Feature weight vector
for i from 1 to m **do**
  $R_i \leftarrow$ bag randomly selected from X
  $H^k_{R_i} \leftarrow findKNearestNeighborSameClass(R_i, X)$ ▷ Get k nearest hits
  for each class C $\neq$ Class (R_i) **do**
    $M^k_{R_i} \leftarrow findKNearestNeighborDifferentClass(R_i, X)$ ▷ Get k nearest misses
  end for
  for A from 1 to numberFeatures **do**
    $$W[A] = W[A] + \frac{\sum_{j=1}^{k} diff_{bag}(A, R_i, H^j_{R_i})}{m \cdot k} + \sum_{C \neq Class(R_i)} \left[\frac{\frac{P(C)}{1-P(Class(R_i))} \sum_{j=1}^{k} diff_{bag}(A, R_i, M^j_{R_i}(C))}{m \cdot k} \right]$$
  end for
end for
for i from 1 to numberFeatures **do**
  if $W[i] > \varepsilon$ **then**
    selectedSubset $\leftarrow$ selectedSubset $\cup \{f_i\}$.
  end if
end for

8.2.2.2 Reliability Based Algorithm

Gan and Yin [9] proposed a method that ranks features based on the reliability measure of each attribute. The class labels are not used for evaluating features, such that this method can also be applied in unsupervised learning. Based on the OWA operator [24], a feature is considered reliable if its values are tightly grouped together. The main steps are shown in Algorithm 12. First, the reliability of each feature f_r for each example X_i using its k nearest neighbors $FR_{X_i,r}$ is determined. Second, the data reliability FR_r of each feature f_r is computed by combining the data reliability of all its values in all samples. From these values, the average data reliability of all features $FR_{average}$ can be derived. As a last step, the features whose FR_r is bigger than the average data reliability are selected. The calculation of the nearest neighbors is carried out using the minimum Hausdorff distance, such that the difference of feature f_r between two bags is equal to the minimum difference of feature f_r between instances from those bags.

This method as well as ReliefF-MI both use a k nearest neighbor approach to determine the relevance of each attribute. Both proposals use a Hausdorff distance to determine the distance between bags. This bag-wise similarity measure is very important in MIL, since it models the relevance and relationship of different instances inside one bag. Gan et al. [9] use the minimal Hausdorff distance, while Zafra et al. [28] uses the different variants presented in Sect. 3.5.

Experimental Study

The study of Gan and Yin [9] considers two datasets: Musk1 and Musk2 (Sect. 3.6). To show the efficiency of their method, the obtained reduced dataset is fed to four classifiers, whose accuracy is evaluated. The results show that the predictive accuracy of the learners was enhanced by using the reduced dataset. The experimental study also includes a comparative study using different values of the k parameter, which represents the number of neighbors. The results show that different values of this parameter scarcely affect the obtained results.

8.2.3 Embedded Methods

Embedded methods differ from other FS methods in the interaction of FS and learning. Wrapper and embedded methods are often confused. A wrapper method uses a learner to measure the quality of feature subsets without incorporating knowledge about the specific structure of the classification function. Embedded methods on the other hand cannot separate the learning and FS parts. These methods learn which features best contribute to the accuracy of the model while the model is being created.

In this section, we describe two embedded multi-instance FS methods, where FS is combined with the Adaboost or SVM methods. These models implicitly select important features and construct a classifier simultaneously.

Algorithm 12 Reliability based method

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_N\}$ ($X_i \subseteq \mathbb{X}$).
Input: $k \leftarrow$ number of neighbors
Input: $m \leftarrow$ number of features
Output: $selectedSubset$, the selected feature subset
1: $selectedSubset = \emptyset$
2: $FR_{average} = 0$
3: **for** r from 1 to m **do**
4: $maxFeatureDiff \leftarrow$ maxFeatureDifference(X, r). ▷ Calculate maximum difference of feature f_r between two bags in whole training set
5: $FR_r = 0$
6: **for** each $X_i \in X$ **do**
7: $N^k_{X_i,r} = findKNearestNeighbor(X, k, X_i, r)$ ▷ k nearest neighbor of X_i considering feature f_r
8: $D^k_{X_i,r} = averageFeatureDifference(N^k_{X_i,r}, k)$. ▷ Average difference of feature f_r of bag X_i with its k nearest neighbor
9: $FR^k_{X_i,r} = bagFeatureDataReliability(D^k_{X_i,r}, maxFeatureDiff)$. ▷ Data reliability of feature f_r in bag X_i
10: $FR_r = FR_r + FR^k_{X_i,r}$. ▷ Data reliability of each feature by combining all examples
11: **end for**
12: $FR_{average} = FR_{average} + FR_r$
13: **end for**
14: $FR_{average} = FR_{average}/m$ ▷ Average data reliability
15: **for** r from 1 to m **do**
16: **if** $FR_r > FR_{average}$ **then**
17: selectedSubset $\leftarrow$ selectedSubset $\cup\{f_r\}$.
18: **end if**
19: **end for**

8.2.3.1 Adaboost for Feature Selection

Boosting techniques have been extensively used for FS in the MIL scenario. The key idea behind AdaBoost is that a strong classifier can be created by combining many weak classifiers. These weak classifiers need only perform slightly better than random guessing. Given a set of training samples, AdaBoost maintains a weight w for each of them. The weights are initialized uniformly. At each iteration t, one weak classifier is selected and the training samples are provided using weights w_t. A weak classifier h_t is trained on these samples. The weights are updated to put more emphasis on misclassified samples. Samples that are correctly classified by h_t get lower weights, while misclassified samples are assigned higher weights. AdaBoost focuses on samples with higher weights, which seem to be harder to predict correctly. The process continues for T iterations. The final strong classifier is a combination of the weak classifiers.

AdaBoost can be used to select the bag features and build the classifier simultaneously. The core idea is that each feature corresponds to a single weak classifier, such that boosting can select some features out of the pool of all possible features F. In each iteration t, the algorithm selects one new feature and adds it (with the corresponding voting factor) to the ensemble. All features are evaluated and the best

one is selected to form the weak classifiers h_t. The sample weights are updated. In the last step, a strong classifier H is computed as a weighted linear combination of the weak classifiers. The number of iterations T is related to the number of dimensions with a sufficient differentiation ability in the whole feature vector. Algorithm 13 shows this process. Some differences with respect to the original AdaBoost method [7] are evident. In this schema, the sums of sample weights for positive and negative samples are always kept equal to 1/2. This maintains the balance between positive and negative bags. The weak classifier uses the weights in its real-valued prediction. The final strong classifier is a direct combination of the weak classifiers instead of a weighted combination.

Algorithm 13 AdaBoost feature selection Algorithm for MIL

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_N\}$ $(X_i \subseteq \mathbb{X})$
Output: H, the final strong classifier
1: Map X to a new bag-level feature space
2: $n^+ \leftarrow$ number of positive samples
3: $n^- \leftarrow$ number of negative samples
4: **if** $Class(X_i)$ is positive **then** ▷ Initialize the weight vector
5: $w(X_i) = \frac{1}{2}n^+$
6: **else**
7: $w(X_i) = \frac{1}{2}n^-$
8: **end if**
9: **for** t from 1 to T **do**
10: Train the weak classifier h_j for each feature j using w(X_i)
11: Calculate the training error e_j of h_j,

$$e_j = \sum_i w(X_i)|h_j(X_i) - Class(X_i)|$$

12: Choose $h_t = h_j$ with the lowest error and set $e_t = e_j$
13: Update weights for positive examples

$$w_{t+1}(X_i) = \frac{w_t(X_i)\exp^{(-Class(X_i)\cdot h_t(X_i))}}{Z_{tp}}$$

14: Update weights for negative examples

$$w_{t+1}(X_i) = \frac{w_t(X_i)\exp^{(-Class(X_i)\cdot h_t(X_i))}}{Z_{tn}}$$

▷ Z_{tp} and Z_{tn} are normalization factors to ensure that w_{t+1} is distribution and that the weight of positive and negative samples all sum up to 1/2
15: **end for**
16: $H(X) = sign[\sum_{t=1}^{T} h_t(X)]$

Depending on the employed weak classifiers, different proposals can be found regarding the weight distribution in their training phase and the way they are combined at the end of the algorithm. Different map-based algorithms are encountered.

As described in Sect. 5.3, these methods map each bag into a new feature vector and thereby transforms the multi-instance data to a single-instance representation. The MIL problem is converted to a standard single-instance problem, usually with a higher dimensionality such that it is convenient to perform a FS step. Any traditional single-instance FS method can be applied. We list proposals using AdaBoost with a linear weak classifier for selecting the bag features obtained by mapping and building the final classifier simultaneously (Algorithm 13).

The MI-AdaBoost algorithm was proposed by Yuan et al. [27]. It uses AdaBoost to select the bag features mapped by a certain set of instance prototypes. It considers two types of instance prototypes: instances from positive bags and the clustering centers of the instances from negative bags. The minimum Hausdorff distance is used to measure the distance between bags.

Zhu et al. [32] proposed the MCMI-AdaBoost method, an algorithm that uses AdaBoost to select the bag features by computing the Hausdorff distance to define a similarity measure between two bags. The bags are mapped to a new bag feature space based on this similarity. An AdaBoost algorithm is proposed to build a two-level classifier converting the multi-class classification problem to a series of two-class classification problems. The output of the first level indicates the possibility that a bag belongs to one class. The second level performs a two-class classification between the two classes with the highest possibility.

Based on their previous work, Yuan et al. developed the existence-based MIL called EBMIL in [25]. This method is able to select different feature modalities for each concept under MIL settings. As a step prior to AdaBoost, a mapping is applied based on points in the instance-level feature space, that hold potential information on the positive and opposite concepts. Positive instance prototypes and opposite instance prototypes are considered. The former are all the instances from positive bags, while the latter are the clustering centers of instances from negative bags.

More recently, Zhang et al. [31] proposed a boosted exemplar learning (BEL) approach for the computer vision field. Based on the learned exemplar, M candidate exemplars are obtained. Each action bag (e.g., an action in a video clip) is described as an M-dimensional vector of its similarity with the M exemplars. The AdaBoost algorithm integrates the FS and action modeling.

Ciliberto et al. [3] follow the philosophy of AdaBoost to design their online MIL algorithm. They include a mechanism for online FS based on Algorithm 13. In an online context, it is likely that useful and descriptive features (and hence potential centers for new weak classifiers) will not be available from the start, but may become available in a later stage. In the problem studied in [3], the object to be learned can rotate, revealing its previously hidden parts.

Experimental Study

The studies of Yuan et al. [25, 27] consider the Corel and Musk datasets (Sect. 3.6) to evaluate their proposals. The mean average precision and computation time are used as evaluation measures. The experimental results show that MI-AdaBoost [27] is much more efficient than the 1-norm SVM [5] and MI-Boosting [7]. Concretely, for the Corel dataset, MI-AdaBoost performs better than MI-Boosting, while its results

are comparable with the 1-norm SVM. In the Musk datasets, MI-AdaBoost performs better than both MI-Boosting and 1-norm SVM. The EBMIL algorithm [25] achieves promising experimental results on the Corel dataset compared with four other feature reduction methods, confirming its effectiveness.

Other models based on AdaBoost for feature selection described in this section are applied to solve particular problems, such as the study of Zhu et al. [32] that shows that their approach, the MCMI-AdaBoost method, is an effective solution for the lung cancer classification problem. In order to evaluate its performance, they compare the accuracy of their method with four other proposals, including classic algorithms such as Citation-kNN and an SVM based algorithm. The study of Zhang et al. [31] considers two available datasets of video action recognition, the KTH human motion dataset and Weizmann human action dataset. To demonstrate the validity and effectiveness of their BEL algorithm, they compare its results with four other MIL classification methods based on the accuracy. The results show that the BEL algorithm outperforms its competitors.

8.2.3.2 SVM for Feature Selection

SVMs have also been widely used for interweaving FS and classifier construction. They allow the incorporation of feature weighting in their kernel function to combine different features. The performance of these methods can be improved by providing information about the features during model generation.

Ngiam et al. [20] incorporate feature weighting into the kernel function based on the idea that different features work well with different concepts. In their proposal, the weight learning is carried out by a simple greedy algorithm (Algorithm 14). In order to obtain the final classifier, each concept is considered independently using an SVM with extended Gaussian kernels over the χ^2 distance:

$$K(X_i, X_j) = \sum_{f \in F} \frac{1}{\mu_f} \chi^2(f(X_i), f(X_j)),$$

where μ_f is the average χ^2 distance for a particular feature, used to normalize the distances across different features.

There are others kernel-based methods that carry out FS using an SVM model to select important features and construct a classifier simultaneously. These methods transform MIL into a FS problem by embedding bags into a new feature space. According to the specific mapping function used in the transformation, the final purpose of a method can change. The MILES method of Chen et al. [5] (Sect. 5.3.4) falls in this category. MILES maps the bags via an instance similarity measure. It uses each instance as a candidate target point, such that the induced space has a high dimensionality. The authors use a $1-$norm SVM to select relevant features and build classifiers at the same time. At its core, this approach identifies relevant instances in the new bag feature space, since each feature is induced by an instance.

Algorithm 14 Greedy FS

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_N\}$ ($X_i \subseteq \mathbb{X}$), $Class(X_i) = 0$ for the negative examples and $Class(X_i) = 1$ for the positive examples
Input: $F \leftarrow$ all features
Output: $selectedSubset$, feature subset
1: **repeat**
2: **for** each feature $f \in F$ **do**
3: Compute error rate if f is removed
4: **end for**
5: Remove the feature which results in the highest improvement
6: **until** removing any feature results in worse performance
7: **repeat**
8: **for** each feature $f \in F$ **do**
9: Compute error rate if f is added
10: Compute error rate if f is removed
11: **end for**
12: $selectedSubset \leftarrow$ Add feature which give best improvement
13: **until** local optimum is reached

The Multiple Instance Online (MIO) method proposed by Li et al. [14] is an online MI learning algorithm that has an efficient online update procedure. Similar to MILES, it maps each bag to a feature space defined by all instances and then performs joint FS and classification by using the 1-norm SVM.

The MIL-MFS (Multiple-Instance Learning with Multiple Feature Selection) algorithm was proposed by Jhuo et al. [11] and uses multiple kernel learning. The authors use a similarity based feature representation, where each instance may be mapped into diverse feature spaces. It iteratively selects the fusing of multiple features for classifier training.

More recently, Mao et al. [19] proposed the FSPO (Feature Selection method for multivariate Performance measures Optimization) algorithm. They propose a generalized sparse regularizer for FS, based on which a unified FS framework is presented for general loss functions. Specifically, they propose a two-layer cutting plane algorithm including group feature generation and selection to solve this problem effectively and efficiently. Multiple kernel learning is proposed to deal with the exponential size of constraints induced by multivariate losses.

Experimental Study

The study of Jhuo et al. [11] considers the Corel dataset (Sect. 3.6). They compare their proposal, the MIL-MFS algorithm, with four multi-instance classifiers based on SVM. Their results show that their method achieves the best accuracy among the included algorithms.

Considering a proposal of online MI learning, the study of Li et al. [14] utilizes synthetic datasets and the Musk datasets (Sect. 3.6) to compare their MIO algorithm with the MILES method. The experimental results show that their proposal outperforms MILES with a small number of passes. Moreover, the average error of nine

multi-instance classifiers is used to validate the efficiency of MIO and the latter is shown to achieve competitive results.

Finally, a more recent study carried out by Mao et al. [19] considers the Corel and News datasets (Sect. 3.6). They compare their FSPO algorithm with four feature selection methods and evaluate the results using the F1 score measure [19]. Comparing with various feature selection methods, the FSPO algorithm is shown to be superior to the others.

8.2.4 Hybrid Method: HyDR-MI Algorithm

Hybrid FS methods combine the advantages of filter and wrapper methods. To determine the important properties of the feature space, a filter method assigns a score to each attribute. Features with very low scores are considered to be irrelevant and can be discarded. The reduced or ranked feature set is provided to a wrapper method, whose purpose is to select the best feature subset for a particular MIL algorithm.

Algorithm 15 HyDR-MI algorithm

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_N\}$ $(X_i \subseteq \mathbb{X})$
Output: subsetFeature, the most relevant feature subset
▷ ReliefF-MI Method
subsetFeature ← obtained with method ReliefF-MI
▷ Genetic Algorithm
$P_0 \leftarrow$ initial population, generated randomly
$P_0 \leftarrow$ evaluation of P_0 using a classification method
$t \leftarrow 0$
repeat
  $P_{parent} \leftarrow$ selectionParents (P_t)
  $P_{offspring} \leftarrow$ genetic operators crossover and mutation over P_{parent}
  $P_{offspring} \leftarrow$ evaluation of $P_{offspring}$ using a classification method
  $P_{t+1} \leftarrow$ update population using P_t and $P_{offspring}$
  $t \leftarrow t + 1$
until $t <$ maxGenerations
subsetFeature ← the best individual obtained in genetic algorithm

Adhering to this setup, Zafra et al. [29] developed HyDR-MI (Hybrid Dimensionality Reduction method for Multiple Instance learning), as shown in Algorithm 15. This method consists of a filter component based on the ReliefF-MI algorithm [28] and a wrapper component based on a genetic algorithm [23] that optimizes the search for the best feature subset from a reduced set of features obtained by filter. This combination benefits both sides. On the one hand, the main restriction of the ReliefF-MI algorithm is the necessity of setting a threshold which determines how many top scored features should be selected. It evaluates each feature individually and cannot handle the problem of feature redundancy appropriately. The genetic algorithm assists the search for the best feature subset and thereby solves these issues. It uses

the performance of a classifier as a fitness function and optimizes the FS by finding the most suitable subset for that classifier. On the other hand, the main limitation of a genetic algorithm is its computation time. ReliefF-MI helps to reduce the search space to achieve better results in less time.

Experimental Study

The study of Zafra et al. [29] considers five benchmark datasets: Musk1, Musk2, Elephant, Tiger, and Fox (Sect. 3.6). To show the efficiency of HyDR-MI compared to ReliefF-MI [28], the results of 17 multi-instance classification algorithms are compared based on the accuracy.

The results show the potential of HyDR-MI statistically improving the predictive performance of many classifiers compared to ReliefF-MI. This is achieved by the possibility to decide how many of the top ranked features are useful for each particular algorithm and the possibility to discard redundant attributes.

8.3 Multiple Instance Methods for Bag Prototype Selection

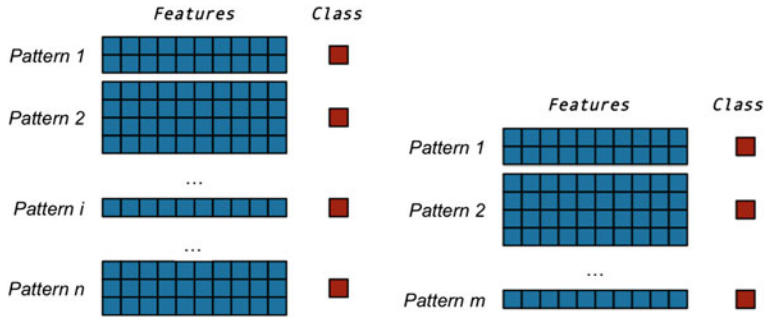
The second data reduction technique that we consider in this chapter is bag prototype selection. In Sect. 8.3.1, we introduce this concept. Section 8.3.2 describes several multi-instance methods implementing this procedure.

8.3.1 Introduction to Bag Prototype Selection

Bag prototype selection (BPS) methods reduce the dataset by selecting a subset of samples. The aim is to eliminate noisy and irrelevant bags and preserve only the most informative ones.

As multi-instance samples are bags composed of one or more instances, a different interpretation can be raised compared to the concept of instance selection in single-instance learning. BPS carries out a bag selection. From this perspective, the aim would be similar to traditional instance selection, that is, reducing the number of observations in the dataset. The distinctive feature is that the removal of a bag implies the elimination of all instances contained in it. An instance selection within each bag can be considered as well. This setting would eliminate individual instances, but it does not reflect the traditional aim of instance selection, which is the reduction of the number of data samples. The number of instances inside bags would be lower, but the total number of bags would be maintained.

Nevertheless, instance prototype selection has always concerned MIC algorithm developers. According to the standard MI assumption, negative bags contain only negative instances, while positive bags contain both positive and negative ones. Instance label ambiguity lies with the positive bags. Mislabeling negative instances in positive bags can limit the performance of multi-instance classifiers. Many methods have focused on selecting a subset of instances from positive bags to learn the



(a) Initial dataset (n bags) (b) Reduced dataset (m bags)

Fig. 8.2 MI bag selection method

classifier. For example, the EM-DD algorithm [30] chooses one instance that is most consistent with the current hypothesis in each positive bag to predict the label for a new bag. DD-SVM [4] depends on the DD concept to identify instance prototypes. Those instances corresponding to local maximizers of the DD function are chosen as instance prototypes, after which an SVM with a Gaussian kernel is learned in the embedded space. MILD [13] performs the instance selection based on a conditional probability model. The instance having the highest ability to distinguish between positive and negative training bags is chosen from each positive bag as an instance prototype. The method learns a standard SVM with a Gaussian kernel using bag-level features. MILIS [8] achieves the initial instance selection by modeling the distribution of the negative population with a Gaussian-kernel-based kernel density estimator. It depends on an iterative optimization framework to update instance prototypes and learn a linear SVM.

Many algorithms can be included here, as several aim to locate the most relevant instance(s) inside of a bag, as related in previous chapters. In this section, we focus on BPS that tries to reduce the number of training bags. These methods have emerged recently in MIL and have a similar finality as single-instance instance selection methods. Figure 8.2 visualizes the BPS process. Figure 8.2a shows the initial dataset, where each sample is composed of several instances. The preprocessed dataset is shown in Fig. 8.2b. The number of bags has been reduced with respect to the initial dataset.

BPS can be defined as a search problem to find a subset of bags which optimizes a particular criterion. Formally, let T be the original dataset with n bags and S a subset of m bags ($m << n$) [10]. BPS tries to find the optimal subset S that does not contain superfluous bags and with which the performance obtained by the classifier is similar to or better than that with the original set T.

Following a similar taxonomy as for FS (Sect. 8.2.1), different BPS methods can be divided in filter, wrapper, and embedded categories. To the best of our knowledge, only filter proposals have been presented for BPS. A summary of these methods is shown in Table 8.2.

Table 8.2 Bag prototype selection methods in multi-instance learning

Bag prototype selection methods
Filter methods
MICLONE algorithm [26]
MILNS algorithm [26]
MILSUP algorithm [26]

8.3.2 Filter Methods

Three proposals designed by Yuan et al. [26] can be classified as filter methods. Their MILCLONE, MILNS, and MILSUP methods are respectively based on clonal selection theory [2], the negative selection principle [12], and self-regulation and suppression mechanisms in natural immune systems [6]. The main features of these methods are described below.

8.3.2.1 MICLONE Algorithm

The MICLONE method [26] is based on clonal selection theory [2]. This theory explains the basic response of the adaptive immune system to an antigenic stimulus. Only those cells capable of recognizing an antigen proliferate, while those that do not are not selected. The main steps of MICLONE are shown in Algorithm 16. The training set is composed of antigens (training bags). One antigen is provided to the algorithm at a time, until all have gone through the entire process. After the initialization of memory cells, memory cell identification, generation of candidate memory cell and memory cell introduction, the output is a set of memory cells, and corresponding to bag prototypes. In more detail, the first stage initializes memory cells, where some antigens are selected to form the memory pool. Second, memory cell identification is carried out and candidate memory cell are generated. For a given antigen X, its closest (most stimulated) antigen with the same class is chosen as its bag prototype (memory cell) from the pool. A stimulation function, which can determine the closest bag, needs to be defined. The authors use the minimum Hausdorff distance. In the memory cell introduction stage, developed candidate memory cell are added to the existing set. The most stimulated memory cell is removed from the pool, if the affinity between the candidate memory cell and its parental instance from the most stimulated cell is less than the product of the affinity threshold AT_{CLONE} and the user-specified threshold ATS_{CLONE}. The former is calculated as

$$AT_{CLONE} = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} affinity(X_i, X_j)}{\frac{n(n-1)}{2}},$$

where n is the number of training antigens, X_i and X_j are the ith and jth training antigen and $affinity(X_i, X_j)$ returns the minimal Hausdorff distance between two bags.

8.3.2.2 MILNS Algorithm

MILNS [26] is based on the negative selection algorithm [12]. These methods describe the negative representation of information when negative examples are not available. For example, in many anomaly detection applications, only positive (normal) examples are available for training, while negative (abnormal) examples are not. The positive examples are used to obtain some representative negative examples known as detectors.

The main steps of this method are described in Algorithm 17. All positive bags are regarded as self samples (self cells) and all negative bags as the set of candidate detectors (antibodies) from which negative example prototypes will be selected. Given a candidate detector, the method scans the set of self samples for the one with the lowest affinity with the candidate detector. If the affinity approximated by means of the minimum Hausdorff distance is greater than the product of the affinity threshold AT_{NS} and the threshold ATS_{NS} provided by the user, the candidate detector is considered as a negative example prototype. The affinity threshold is the average affinity value over all self examples, which is calculated as

$$AT_{NS} = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} affinity(X_i, X_j)}{\frac{n(n-1)}{2}},$$

where n is the number of self samples, X_i and X_j are the ith and jth self samples and $affinity(X_i, X_j)$ returns the minimal Hausdorff distance between these two.

Algorithm 16 MICLONE Algorithm

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_N\}$ $(X_i \subseteq \mathbb{X})$, $Class(X_i) \in \{0, 1\}$
Output: $subsetBags \leftarrow$ more relevant bag subset.

```
memory = initializeRandomRepertoire()
for X_i ∈ memory do
    best = memoryCellIdentification(Class(X_i))
    fit = generationCandidateMemoryCell(best, X_i)
    Add fit to memory cell
    if affinity(fit, closest_best) < (AT_CLONE · ATS_CLONE) then      ▷ closest_best its the closest bag with the same class as X_i
        Eliminate best of memory cell
    end if
end for
```

8.3.2.3 MILSUP Algorithm

MILSUP [26] is based on an immune inspired suppressive algorithm [6]. It is inspired by the self-regulation and suppression mechanisms in the biological immune system. According to the self-regulation mechanism, those cells unable to neutralize danger tend to disappear from the organism (or be suppressed). By analogy, data not relevant to the classifier is eliminated from the training set.

The main steps of MILSUP are described in Algorithm 18. The affinity approximation between two bags is given by means of the minimal Hausdorff distance. The dataset is divided into two subsets, the first one representing the lymphocytes in the organism (training set) and the second one a set of pathogens (suppression set). The algorithm sets out with the idea that the system's model must identify the best subset of lymphocytes in order to recognize pathogens. Specifically, each pathogen in the suppression set is classified according to the closest lymphocytes in the training set. Those lymphocytes able to recognize pathogens are retained, while others are eliminated. The recognition ability is determined by comparing the label of the closest lymphocyte with that of the corresponding pathogen. If their labels are the same, the lymphocyte is considered to have the ability to recognize the corresponding pathogen, otherwise it is not.

Experimental Study

The study of Yuan et al. [26] considers five datasets: Musk1, Musk2, Elephant, Tiger, and Fox (Sect. 3.6). The experimental study includes a comparison between the three proposals (MILSUP, MICLONE, and MILNS). The reduced datasets provided by these methods are used by 20 classifiers for the Elephant, Tiger, and Fox datasets and by 12 classification algorithms for the Musk1 and Musk2 datasets. The accuracy and computation time are used as evaluation measures. The results show that MILCLONE and MILNS are competitive with each other in terms of their effect on the classification accuracy. They are superior to MILSUP. All methods considerably reduce the computation time of the classifiers.

Algorithm 17 MILNS Algorithm

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_N\}$ $(X_i \subseteq \mathbb{X})$
Output: $subsetBags \leftarrow$ more relevant bag subset.
1: $self \leftarrow$ set of all positive bags
2: $detector \leftarrow$ set of all negative bags
3: $P \leftarrow \emptyset$ ▷ P is the set of bag prototypes
4: **for** $X_d \in detector$ **do**
5: $X_p \leftarrow argmin_{X_s \in Self} affinity(X_s, X_d)$
6: **if** $affinity(X_p, X_d) > AT_{NS} \cdot AT S_{NS}$ **then**
7: Add X_p to P
8: **end if**
9: **end for**

Algorithm 18 MILSUP algorithm

Input: $X \leftarrow$ multi-instance training set $\{X_1, X_2, \ldots, X_n\}$, $(X_i \subseteq \mathbb{X})$, $Class(X_i) \in \{0, 1\}$
Input: Fraction $f \in [0, 1]$
Output: $bagSubset$, most relevant bag subset
1: $WBCs \leftarrow$ randomly assign $f \cdot n$ examples
2: $Pathogens \leftarrow$ examples not assigned to WBCs
3: **for** $X \in WBCs$ **do** ▷ Set a survival signal for every WBC and initialize it to be false
4: $Survival_X = false$
5: **end for**
6: **for** $X_p \in Pathogens$ **do**
7: $Nearest_{WBC} \leftarrow argmin_{w \in WBCs} affinity(X_p, X_w)$
8: **if** $Class(Nearest_{WBC}) = Class(X_p)$ **then**
9: $Survival_{Nearest_{WBC}} = true$
10: **end if**
11: **end for**
12: Eliminate those bags of $WBCs$ with the survival signal set to false
13: Add to $bagSubset$ those bags of $WBCs$ with survival signal set to true

8.4 Summarizing Comments

Data reduction in MIL is a critical challenge. The inherent data ambiguity, where instances in a positive bag may or may not approximate the positive concept, adds more complexity to the problem. In this chapter, we considered two important tasks to reduce the computational complexity and improve the performance of the subsequent learner: FS and BPS. In case of FS, different methods are described adhering to the well-known taxonomy based on filter, wrapper, and embedded methods. A similar study is carried out for BPS, which is the more recently addressed task.

References

1. Bellman, R.: Dynamic Programming and Lagrange Multipliers. Princeton University Press, Princeton (1957)
2. Burnet, S.F.M.: The Clonal Selection Theory of Acquired Immunity. Vanderbilt University Press, Nashville (1959)
3. Ciliberto, C., Smeraldi, F., Natale, L., Metta, G.: Online multiple instance learning applied to hand detection in a humanoid robot. In: De Luca, A. (ed.) Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS 2011), pp. 1526–1532. IEEE, San Francisco (2011)
4. Chen, Y., Wang, J.Z.: Image categorization by learning and reasoning with regions. J. Mach. Learn. Res. **5**, 803–821 (2004)
5. Chen, Y., Bi, J., Wang, J.Z.: MILES: multiple-instance learning via embedded instance selection. IEEE Trans. Pattern Anal. **28**(12), 1931–1947 (2006)
6. Figueredo, G.P., Ebecken, N.F., Augusto, D.A., Barbosa, H.J.: An immune-inspired instance selection mechanism for supervised classification. Memet. Comput. **4**(2), 135–147 (2012)
7. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann. Stat. **28**(2), 337–407 (2000)

8. Fu, Z., Robles-Kelly, A., Zhou, J.: MILIS: multiple instance learning with instance selection. IEEE Trans. Pattern Anal. **33**(5), 958–977 (2011)
9. Gan, R., Yin, J.: Feature selection in multi-instance learning. Neural Comput. Appl. **23**(3–4), 907–912 (2013)
10. García, S., Luengo, J., Sáez, J.A., López, V., Herrera, F.: A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Trans. Knowl. Data Eng. **25**(4), 734–750 (2013)
11. Jhuo, I.H., Lee, D.T.: Multiple-instance learning: multiple feature selection on instance representation. In: Proceedings of the 25th International Conference on Artificial Intelligence (AAAI 2011), pp. 1794–1795. Association for the Advancement of Artificial Intelligence, San Francisco (2011)
12. Ji, Z., Dasgupta, D.: V-detector: an efficient negative selection algorithm with "probably adequate" detector coverage. Inf. Sci. **179**(10), 1390–1406 (2009)
13. Li, W.J.: MILD: multiple-instance learning via disambiguation. IEEE Trans. Knowl. Data Eng. **22**(1), 76–89 (2010)
14. Li, M., Kwok, J.T., Lu, B.L.: Online multiple instance learning with no regret. In: Boykov, Y., Schmidt, F.R., Kahl, F., Lempitsky, V. (eds.) Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR 2010), pp. 1395–1401. IEEE, Los Alamitos (2010)
15. Liu, H., Motoda, H.: Feature Extraction, Construction and Selection: A Data Mining Perspective. Kluwer, Boston (1998)
16. Liu, H., Motoda, H.: Instance selection and construction for data mining. Kluwer Academic Publisher, Norwell (2001)
17. Liu, H., Motoda, H.: Computational Methods of Feature Selection. CRC Press, Boca Raton (2007)
18. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. Knowl. Data Eng. **17**(4), 491–502 (2005)
19. Mao, Q., Tsang, I.W.H.: A feature selection method for multivariate performance measures. IEEE Trans. Pattern Anal. **35**(9), 2051–2063 (2013)
20. Ngiam, J., Goh, H.: Learning global and regional features for photo annotation. In: Peters, C., Muller, H., Caputo, B., Gonzalo, J., Jones, G.J.F., Kalpathy-Cramer, J., Former, P., Giampiccolo, D. (eds.) Proceedings of 10th Workshop of Cross-Language Evaluation Forum for European Languages (CLEF 2009), pp. 287–290. Springer, Berlin (2009)
21. Robnikikonja, M., Kononenko, I.: Theoretical and empirical analysis of ReliefF and RReliefF. J. Mach. Learn. Res. **53**(1–2), 23–69 (2003)
22. Triguero, I., Derrac, J., García, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans. Syst. Man Cybern. C **42**(1), 86–100 (2012)
23. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994)
24. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Trans. Syst. Man Cybern. **18**(1), 183–190 (1988)
25. Yuan, X., Wang, M., Song, Y.: Concept-dependent image annotation via existence-based multiple-instance learning. In: Proceedings of the IEEE International Conference on Systems. Man and Cybernetics (SMC 2009), pp. 4112–4117. IEEE, Los Alamitos (2009)
26. Yuan, L., Liu, J., Tang, X.: Combining example selection with instance selection to speed up multiple-instance learning. Neurocomputing **129**, 504–515 (2014)
27. Yuan, X., Hua, X.S., Wang, M., Qi, G.J., Wu, X.Q.: A novel multiple instance learning approach for image retrieval based on adaboost feature selection. In: Yun-Qing, S., Liao, M., Hu, Y.H., Sheu, P., Ostermann, J. (eds.) Proceedings of the International Conference on Multimedia and Expo (ICME 2007), pp. 1491–1494. IEEE Service Center, Piscataway (2007)
28. Zafra, A., Pechenizkiy, M., Ventura, S.: ReliefF-MI: an extension of ReliefF to multiple instance learning. Neurocomputing **75**(1), 210–218 (2012)
29. Zafra, A., Pechenizkiy, M., Ventura, S.: HyDR-MI: a hybrid algorithm to reduce dimensionality in multiple instance learning. Inf. Sci. **222**, 282–301 (2013)

30. Zhang, Q., Goldman, S.: EM-DD: an improved multiple-instance learning technique. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Proceedings of the 17th Conference on Advances in Neural Information Processing Systems (NIPS 1998), pp. 1073–1080. MIT Press, Cambridge (1998)
31. Zhang, T., Liu, J., Liu, S., Xu, C., Lu, H.: Boosted exemplar learning for action recognition and annotation. IEEE Trans. Circ. Syst. Video **21**(7), 853–866 (2011)
32. Zhu, L., Zhao, B., Gao, Y.: Multi-class multi-instance learning for lung cancer image classification based on bag feature selection. In: Wang, L., Jin, Y. (eds.) Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2008). Lecture Notes in Artificial Intelligence, pp. 487–492. Springer, Berlin (2008)