

A Two-level Learning Method for Generalized Multi-instance Problems

Nils Weidmann^{1,2}, Eibe Frank², and Bernhard Pfahringer²

¹Department of Computer Science
University of Freiburg
Freiburg, Germany

²Department of Computer Science
University of Waikato
Hamilton, New Zealand

weidmann@informatik.uni-freiburg.de {eibe,bernhard}@cs.waikato.ac.nz

Abstract. In traditional multi-instance (MI) learning, a single positive instance in a bag produces a positive class label. Hence, the learner knows how the bag’s class label depends on the labels of the instances in the bag and can explicitly use this information to solve the learning task. In this paper we investigate a generalized view of the MI problem where this simple assumption no longer holds. We assume that an “interaction” between instances in a bag determines the class label. Our two-level learning method for this type of problem transforms an MI bag into a single meta-instance that can be learned by a standard propositional method. The meta-instance indicates which regions in the instance space are covered by instances of the bag. Results on both artificial and real-world data show that this two-level classification approach is well suited for generalized MI problems.

1 Introduction

Multi-instance (MI) learning has received a significant amount of attention over the last few years. In MI learning, each training example is a bag of instances with a single class label, and one assumes that the instances in a bag have individual, but unknown, class labels. The bag’s class label depends in some way on the unknown classifications of its instances. The assumption of how the instances’ classifications determine their bag’s class label is called the *multi-instance assumption*. In existing approaches to MI learning, a bag is assumed to be positive if and only if it contains at least one positive instance. This assumption was introduced because it seemed to be adequate in the MI datasets used so far [1]. We refer to it as the *standard* MI assumption.

In this paper we explore a generalization of the standard assumption by extending the process that combines labels of instances to form a bag label. In the standard MI case, one instance that is positive w.r.t. an underlying propositional concept makes a bag positive. Instead of a single underlying concept, we use a set of underlying concepts and require a positive bag to have a certain number of instances in each of them.

We introduce three different generalized MI concepts. In *presence-based* MI datasets, a bag is labeled positive if it contains at least one instance in each of the

underlying concepts; a *threshold-based* MI dataset requires a concept-dependent minimum number of instances of each concept; and in a *count-based* MI dataset, the number of instances per concept is bounded by an upper as well as a lower limit. These three types of MI concepts form a hierarchy, i.e. *presence-based* \subset *threshold-based* \subset *count-based*. Note that the standard MI problem is a special case of a presence-based MI concept with just one underlying concept. Consequently any learner able to solve our generalized MI problem should perform well on standard MI data.

In generalized MI problems, the learner has much less prior knowledge about the way the class label is determined by the instances in a bag, making this type of problem more difficult. We introduce the idea of *two-level-classification* (TLC) to tackle generalized MI problems. In the first step, this method constructs a single instance from a bag. This so-called *meta-instance* represents regions in the instance space and has an attribute for each of these regions. Each attribute indicates the number of instances in the bag that can be found in the corresponding region. Together with the bag’s class label, the meta-instance can be passed to a standard propositional learner in order to learn the influence of the regions on a bag’s classification.

This paper is structured as follows. Section 2 gives an overview over the standard multi-instance problem and introduces notational conventions. In Section 3, we give definitions for our three generalizations of the MI problem. The two-level classification method is outlined in Section 4, and experiments with the algorithm on artificial data and the Musk problems are described in Section 5. We summarize our findings in Section 6.

2 The Standard Multi-Instance Setting

In traditional supervised learning, each learning example consists of a fixed number of attributes and a class label. However, sometimes only a collection of instances can be labeled. For these cases, Dietterich et al. [1] introduced the notion of a multi-instance problem, where a “bag” of instances is given a class label. The motivating task was to predict whether a certain molecule is active or not. This is determined by its chemical binding properties, which again depend on the shape of the molecule. A molecule occurs in different shapes (conformations), because some of its internal bonds can be rotated. If at least one of the conformations of the molecule binds well to certain receptors, the molecule expresses a “musky” smell and is therefore considered active. In the Musk problems, a bag corresponds to a molecule, and the instances are its conformations.

In this paper, we follow the notation of Gärtner et al. [2]. \mathcal{X} denotes the instance space, Ω is the set of class labels. In MI learning, the class is assumed to be binary, so $\Omega = \{\top, \perp\}$. A MI concept is a function $\nu_{MI} : 2^{\mathcal{X}} \rightarrow \Omega$. In the standard MI case, this function is defined as

$$\nu_{MI}(X) \Leftrightarrow \exists x \in X : c_I(x)$$

where $c_I \in \mathcal{C}$ is a concept from a concept space \mathcal{C} (usually called the “underlying concept”), and $X \subseteq \mathcal{X}$ is a set¹ of instances. In this type of problem, a learner can be sure that every instance that is encountered in a negative bag is also a negative instance w.r.t. the underlying concept. Thus, it can focus on identifying positive instances using axis-parallel rectangles [1], neural networks [3], or the Diverse Density algorithm [4]. However, in this paper we are interested in a more generalized problem that leads to a harder learning task.

3 Generalized Multi-Instance Problems

We extend the assumption of how a bag’s label is determined by the classifications of its instances. In the standard MI case, a single instance that is positive in the underlying concept causes the bag label to be positive. In our case, we do no longer assume a single concept. Instead, a set of underlying concepts is used, each of which contributes to the classification. We assume that criteria based on the number of instances in each concept determine the bag’s class label. Below we introduce generalizations of the standard MI concept that are based on three different types of criteria.

These more general problems cannot be solved simply by identifying positive instances that never occur in negative bags, because an instance in a negative bag can still be positive w.r.t. one of the underlying concepts. In other words, a positive instance does not necessarily cause a bag to be positive. Only if a sufficient number of instances of other concepts is present in the bag, the bag’s label is positive. Thus a learning method for this task must take all instances in the bag into account.

To formalize our generalized view on MI learning, we redefine the MI concept function ν_{MI} introduced in the last paragraph. The data representation is unchanged, which means that we are given bags $X \subseteq \mathcal{X}$ with class labels in $\{\top, \perp\}$. The generalized MI concept function operates on a *set* of underlying concepts $C \subset \mathcal{C}$. We also need a counting function $\Delta : 2^{\mathcal{X}} \times \mathcal{C} \rightarrow \mathbb{N}$, which counts the members of a given concept in a bag.

3.1 Presence-based MI Concepts

Our first generalization is defined in terms of the presence of instances of each concept in a bag. For example, an MI concept of this category is “only if instances of concept c_1 and instances of concept c_2 are present in the bag, the class is positive”. Formally, a presence-based MI concept is a function $\nu_{PB} : 2^{\mathcal{X}} \rightarrow \Omega$, where for a given set of concepts $C \subset \mathcal{C}$,

$$\nu_{PB}(X) \Leftrightarrow \forall c \in C : \Delta(X, c) \geq 1$$

The following example (introduced in [5]) illustrates a presence-based MI concept. Assume we are given a set of bunches of keys. A bunch corresponds to a

¹ For notational convenience, we assume that all the instances in a bag are distinct.

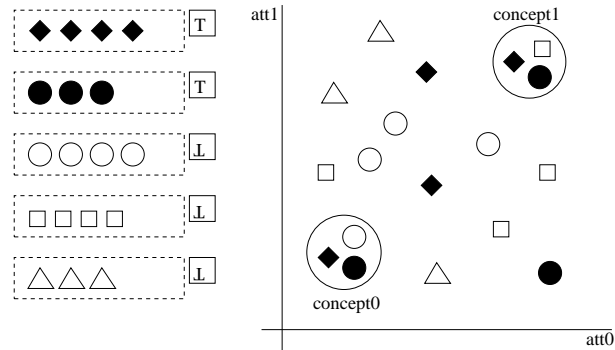


Fig. 1. A multi-instance dataset with a presence-based MI concept. Instances in a two-dimensional instance space (attribute `att0` and attribute `att1`) are assigned to five bags (\blacklozenge , \bullet , \circ , \square and \triangle). Two concepts (`concept0` and `concept1`) are given as circles. The instance in a circle are part of the corresponding concept. \blacklozenge and \bullet are positive bags, because they have instances in both `concept0` and `concept1`.

bag, the keys are its instances. The task is to predict if an unseen bunch can be used to open a locked door. If this door has only one lock, every bunch with at least one key for that lock would be positive. This corresponds to a standard MI learning problem. However, assume there are n different locks that need to be unlocked before the door can be opened. This is an example for a presence-based MI concept, because we need at least one instance (one key) of each of the n concepts (the locks) to classify a bag as positive. Thus the standard MI problem is a special case of this presence-based concept with $|C| = 1$.

Figure 1 visualizes a presence-based MI concept in a two-dimensional instance space. Note that presence-based MI problems have been introduced before as “multi-tuple problems” in an ILP-based setting [6]. The instances are all part of the same instance space, thus the number of underlying database relations is 1. In the ILP definition of standard MI learning, a hypothesis consists of a single rule using only one tuple variable. This rule corresponds to what we call the underlying concept. Multi-tuple problems are the relaxation of this definition, where an arbitrary number of rules can be used. These correspond to our set of concepts. Thus, because presence-based MI problems can be embedded into the ILP framework, they can, at least in principle, be solved by ILP learners. Another generalization of the MI problem called “multi-part problem” has been introduced in an ILP-based setting [5]. In this type of problem no explicit assumption is made about how the instances in a bag contribute to the bag’s classification.

3.2 Threshold-based MI Concepts

Instead of the mere presence of certain concepts in bag, one can require a certain number of instances of each concept to be present simultaneously. If for

each concept, the corresponding number of instances of that concept exceeds a given threshold (which can be different for each concept), the bag is labeled positive and negative otherwise. Formally, we define a threshold-based MI concept function ν_{TB} as

$$\nu_{TB}(X) \Leftrightarrow \forall c_i \in C : \Delta(X, c) \geq t_i$$

where $t_i \in \mathbb{N}$ is called the “lower threshold for concept i ”. An extension of the “key-and-lock” example mentioned above illustrates a count-based MI concept. Assume the door has more than just one lock of each type, and that the keys have to be used simultaneously in order to open the door, i.e. we need one key for each lock. Here, we need at least as many keys for each type of lock as there are locks of this type in the door. In a such a dataset, each positive bag (a bunch of keys) has to have a minimum number of keys (the instances) of each type of lock (the concepts).

3.3 Count-based MI Concepts

The most general concepts in our hierarchy are count-based MI concepts. These require a maximum as well as a minimum number of instances of a certain concept in a bag. This can be formalized as

$$\nu_{CB}(X) \Leftrightarrow \forall c_i \in C : t_i \leq \Delta(X, c) \leq z_i$$

where $t_i \in \mathbb{N}$ is again the lower threshold and $z_i \in \mathbb{N}$ is called the “upper threshold for concept i ”. Imagine the following learning example for this type of problem. We are given daily statistics of the orders processed by an company. An order is usually assigned to one the company’s departments. We want to predict if the company’s workload is within an optimal range, where none of the departments processes too few orders (because its efficiency would be low), or gets too many orders (because it would be overloaded). In a MI representation of this problem, bags are collections of orders (the instances) of a certain day, and the class label indicates whether the company was within an effective workload on that day. Each of the underlying concepts C assigns an order to a department. In order for the company to perform efficiently on a certain day, each of the departments has to work within its optimal range, i.e. the number of instances of this concept must be bounded from below and above. These bounds can be different for each concept.

4 Two-level Classification

Although ILP-based learners can be used for presence-based MI problems, there appears to be no method capable of dealing with threshold- and count-based MI concepts. In particular, learners relying on the standard MI assumption are doomed to fail, because they aim at identifying positive instances that are not present in a negative bag. In our generalized view, this is usually not the case.

Taking a closer look at the way MI data are created, there are two functions that determine a bag’s class label. First, there is a function that assigns an instance in a bag to a concept, a mono-instance concept function. Second, there is the MI concept function that computes a class label from the instances in a bag, given their concept membership by the first function. Thus, a two-level approach to learning seems appropriate. At the first level, we try to learn the structure of the instance space \mathcal{X} , and at the second level we try to discover the interaction that leads to a bag’s class label. Let us discuss the second level first.

4.1 Second Level: Exploring Interactions

Assume we are given the concept membership functions c_i for all concepts $c_i \in C$. Then we can transform a bag X into a meta-instance with $|C|$ numerical attributes whose values indicate the number of instances in the bag that are members of the respective concept. Assigning the bag’s class label to this newly created instance, we end up with a single instance appropriate for propositional learning. Processing all the MI bags in this way results in a propositional dataset that can be used as input for a propositional learner. However, we are not given the concept membership functions, and inducing each c_i directly is not possible, because we neither know the number of concepts that are used, nor the instances’ individual class labels. Instead, we are only given a class label for the bag as a whole. However, it turns out that a decomposition of the instance space into “candidate” regions for each concept is possible, enabling the learner to recover the true MI concept at the second level. This is described in the next section.

4.2 First Level: Structuring the Instance Space

In the first level of our classification procedure, we construct a single instance from an MI bag. The attributes in this instance represent regions of the instance space, and an attribute’s value is simply the number of instances in the bag that pertain to the corresponding region. One possible approach for identifying the regions would be clustering. However, this discards information given by the bag label. If we label each instance with its bag’s label, regions with a high observed proportion of positive instances will be candidate components for concepts (see Figure 1). Hence, the “clustering” method should be sensitive to changes in the class distribution.

Consequently, we use a standard decision tree for imposing a structure on the instance space because it is able to detect these changes. The decision tree is built on the set of all instances contained in all bags, labeled with their bag’s class label. The weight of each instance in a bag X is set to $\frac{1}{|X|} \cdot \frac{N}{b}$, where N denotes the sum of all the bag sizes and b the number of bags in the dataset. This gives bags of different size the same weight and makes the total weight equal to the number of instances. Information gain is used as the test selection measure. A node in the tree is not split further when the weight of its instances sums up to less than 2, and no other form of pruning is used. A unique identifier is

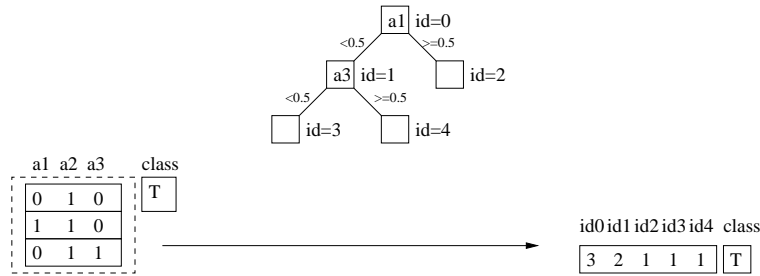


Fig. 2. Constructing a single instance from a bag with three instances and three attributes **a1**, **a2** and **a3**. A decision tree with five nodes is used to identify regions in the instance space. The constructed single instance (right) has an attribute for each node in the tree, that counts the number of instances in the bag pertaining to that node.

assigned to each node of the tree. Using this tree, we convert a bag into a single instance with one numerical attribute for each node in the tree. Each attribute counts how many instances in the bag are assigned to the corresponding node in the tree. See Figure 2 for an illustration.

4.3 Attribute Selection

Attribute selection (AS) can be applied to refine the classifier. If the decision tree is not able to find the region representing a concept, classification at the meta-level will fail. Attributes that do not contribute to the classification of individual instances could be picked as splitting attributes in the tree and thus cause an incorrect representation of the concept regions. Attribute selection tries to eliminate these attributes. In our experiments, we used the method proposed in [7], which evaluates a given subset of attributes by cross-validations on this subset. Note that the cross-validation is performed at the bag level using both levels of TLC. We used backward selection, which starts with all attributes and subsequently eliminates attributes that worsen the performance. Of course, this method is computationally expensive and increases the runtime of the two-level classifier considerably.

5 Experiments

We have evaluated the performance of TLC using both artificial and real-world data. The only publicly available real-world data stems from the Musk problem [1], and for this problem it is very likely that methods able to deal with the standard MI assumption are sufficient. We therefore focus on artificially created datasets, where the performance on different types of MI concepts can be shown, and where we know that the various properties of our three types of problem hold.

Assuming that the first-level classifier can identify concepts over the instance space properly, the second-level learner must be able to learn intervals in the meta-attributes that are responsible for a positive classification. In our experiments, we used Logit-boosted decision stumps [8] with 10 boosting iterations, because they are able to do so, are well-known to be accurate general-purpose classifiers and performed well in initial experiments.

We compare the results of TLC both with and without attribute selection to the Diverse Density (DD) algorithm [4] and the MI Support Vector Machine [2]. Both have been designed for standard MI problems, although the latter does not exploit the MI assumption in any way. In the DD algorithm, we used an initial scaling parameter of 1.0. The MI Support Vector Machine was based on an RBF kernel, a γ -parameter of 0.6 and a C -parameter of 1.0.

5.1 Artificial Datasets

For the artificial datasets in our experiments we used bitstrings of different length l as instances, i.e. $\mathcal{X} = \{0, 1\}^l$. The length l of the bitstring is the sum of the number of relevant attributes l_r and the number of irrelevant attributes l_i . Concepts c are bitstrings in $\{0, 1\}^{l_r}$, i.e. an instance is member of a concept c_i if and only if its first l_r attributes match the bit pattern c_i .

The construction of an artificial dataset of a certain type (either presence-, threshold-, or count-based) works as follows. We randomly select different bitstrings of length l_r as concepts. A positive bag is first filled with instances of the different concepts according to the type of data we want to create: for presence-based MI concepts, at least one instance of each concept is added to the positive bag; for a threshold-based MI concept at least t_i instances of each concept c_i are added; and for a count-based MI concept, we add at least t_i and at most z_i instances of concept c_i . Since we choose different bitstrings representing the concepts, instances cannot be member of two concepts at the same time. In a second step, a number of random instances that are not member of any concept are added to the bag. These are created by uniformly drawing instances from the instance space $\{0, 1\}^l$ and ensuring that they are not member of any of the concepts. These “irrelevant” instances are designed to make learning problem harder and more realistic.

Negative bags are constructed by first creating a positive bag in the way described above and then converting it into a negative bag. Since a negative bag must not satisfy the used MI concept, we need to negate at least one condition imposed on one of the concepts C . For a presence-based concept, we need to remove all the instances of at least one concept c_i , for a threshold-based concept the number of instances of at least one concept c_i must be less than t_i , and for a count-based concept, the number of instances of at least one concept must be increased or decreased so that it is either less than t_i or greater than z_i . Every possible subset of C except the empty set can be negated to create a negative bag. We choose uniformly from these $2^{|C|} - 1$ possibilities. Increasing/decreasing the number of instances of a concept in bag can be done by replacing random instances by instances of the respective concept or replacing some of the concept’s

Table 1. Results for presence-based MI data using only one underlying concept

	DD	MI SVM	TLC without AS	TLC with AS
1-5-0	100 ± 0	94.35 ± 0.74	100 ± 0	100 ± 0
1-5-5	100 ± 0	92.26 ± 0.95	100 ± 0	100 ± 0
1-5-10	100 ± 0	90.74 ± 0.76	100 ± 0	100 ± 0
1-10-0	99.57 ± 0.59	96.20 ± 0.85	97.46 ± 0.92	97.07 ± 0.3
1-10-5	99.41 ± 0.54	94.67 ± 1.35	97.57 ± 0.87	96.11 ± 1.57
1-10-10	99.8 ± 0.44	91.66 ± 2.3	97.85 ± 0.89	94.34 ± 2.66

Table 2. Results for presence-based MI data using two or three underlying concepts

	MI SVM	TLC without AS	TLC with AS
2-5-0	80.96 ± 1.9	100 ± 0	100 ± 0
2-5-5	81.17 ± 1.79	88.38 ± 11.91	100 ± 0
2-5-10	79.21 ± 1.66	78.64 ± 13.15	100 ± 0
2-10-0	84.18 ± 0.52	99.01 ± 1.32	97.56 ± 1.38
2-10-5	82.0 ± 1.53	85.18 ± 10.07	96.67 ± 1.58
2-10-10	80.74 ± 0.79	86.63 ± 8.69	94.45 ± 4.54
3-5-0	82.0 ± 2.13	100 ± 0	100 ± 0
3-5-5	82.12 ± 0.98	81.93 ± 2.9	99.98 ± 0.04
3-5-10	81.43 ± 0.96	86.32 ± 6.48	98.49 ± 1.74
3-10-0	84.39 ± 1.25	95.68 ± 3.78	94.73 ± 2.91
3-10-5	84.27 ± 1.44	78.07 ± 0.91	87.41 ± 6.24

instances by random instances. After negation, the bag has the same size as before, thus the average bag size of positive and negative bags is the same.

Presence-based MI Datasets We created presence-based datasets with concepts of length 5 and 10, hence $l_r = 5$ or $l_r = 10$, respectively. Some of our datasets required two concepts to be present in the positive bags ($|C| = 2$), in some we used three concepts ($|C| = 3$). To generate a positive bag, the number of instances in a concept was chosen randomly from $\{1, \dots, 10\}$ for each concept. The number of random instances was selected with equal probability from $\{10|C|, \dots, 10|C|+10\}$. Hence the minimal bag size in this dataset was $|C| + 10|C|$ and the maximal bag size $20|C| + 10$. We trained the classifiers on five different training sets with 50 positive and 50 negative bags each. Tables 1 and 2 show the average accuracy on a test set with 5,000 positive and 5,000 negative bags and the standard deviation of the five runs. The parameters of the dataset are given as $\langle |C| \rangle - \langle l_r \rangle - \langle l_i \rangle$, e.g. 3-10-5 has three concepts, 10 relevant and 5 irrelevant attributes.

The results in Table 1 show that each of the MI learners can deal well with presence-based MI concepts using only one underlying concept (corresponding to the standard MI assumption). DD cannot deal with more than one underlying concept and its performance is not competitive, therefore it is not included in Table 2. The MI SVM does not make explicit use of the standard MI assumption and surprisingly, its similarity measure enables it to perform well on these presence-based datasets (Table 2). The TLC method discovers the underlying

Table 3. Results for threshold-based MI data

	MI SVM	TLC without AS	TLC with AS
42-5-0	84.35 \pm 3.07	100 \pm 0	100 \pm 0
42-5-5	81.54 \pm 2.24	95.93 \pm 9.1	100 \pm 0
42-5-10	81.59 \pm 0.4	84.67 \pm 14.31	100 \pm 0
42-10-0	86.28 \pm 1.33	99.35 \pm 0.45	97.02 \pm 1.65
42-10-5	85.36 \pm 0.92	88.65 \pm 10.12	96.58 \pm 1.91
42-10-10	83.93 \pm 0.36	84.59 \pm 8.08	95.89 \pm 2.05
275-5-0	84.75 \pm 1.03	97.2 \pm 2.78	97.2 \pm 2.78
275-5-5	83.9 \pm 1.29	90.62 \pm 6.57	97.94 \pm 2.83
275-5-10	82.73 \pm 0.85	86.42 \pm 5.39	93.75 \pm 6.63
275-10-0	88.66 \pm 1.12	95.44 \pm 1.21	97.68 \pm 1.57
275-10-5	87.05 \pm 0.75	86.92 \pm 6.56	90.44 \pm 4.63

MI concept perfectly in most cases, if no irrelevant attributes are used. Irrelevant attributes make it hard for the decision tree to represent the true underlying concepts, which in turn worsens the performance of the classifier (e.g. for dataset 2-5-10). However, the attribute selection method eliminates the irrelevant attributes, enabling the classifier to give good results even on datasets with a high ratio of irrelevant attributes (dataset 2-5-10 and 3-5-5).

Threshold-based MI Datasets We created threshold-based datasets using $l_r = 5$ or $l_r = 10$ and two or three concepts. We chose thresholds $t_1 = 4$ and $t_2 = 2$ for six datasets and $t_1 = 2$, $t_2 = 7$ and $t_3 = 5$ for another five datasets. For positive bags, the number of instances of concept c_i was chosen randomly from $\{t_i, \dots, 10\}$. To form a negative bag, we replaced at least $(\Delta(X, c_i) - t_i - 1)$ instances of a concept c_i in a positive bag X by random instances. The minimal bag size in this dataset is $\sum_i t_i + 10|C|$, the maximal size is $20|C| + 10$. Table 3 shows the results. The parameters of the dataset are given as $\langle t_1..t_n \rangle - \langle l_r \rangle - \langle l_i \rangle$. For example, 42-10-0 has at least 4 instances of the first concept and 2 instances of the second concept in a positive bag, with 10 relevant and 0 irrelevant attributes.

Even though threshold-based MI concepts are harder to learn than presence-based ones, the results for the MI SVM show that it can deal quite well with these datasets. However, TLC achieves better results, although the variance of the results can be high (datasets 42-5-10 and 275-5-5) if no attribute selection is performed. We did not apply attribute selection in conjunction with the MI SVM, because Table 3 shows that its performance is not greatly affected by irrelevant attributes.

Count-based MI Datasets Our count-based MI datasets are based on 5 or 10 relevant attributes. We used the same value for both thresholds t_i and z_i , because we considered this an interesting special case. In the following, we refer to this value as z_i . Hence, the number of instances of concept c_i is exactly z_i in a positive bag. For six datasets, we set $z_1 = 4$ and $z_2 = 4$, and for five other datasets, $z_1 = 2$, $z_2 = 7$ and $z_3 = 5$. A negative bag can be created by either

Table 4. Results for count-based MI data

	MI SVM	TLC without AS	TLC with AS
42-5-0	52.78 ± 2	99.55 ± 0.64	99.35 ± 0.92
42-5-5	52.7 ± 1.04	57.89 ± 11.09	85.22 ± 21.65
42-5-10	53.83 ± 1.46	57.63 ± 7.64	70.9 ± 25.94
42-10-0	55.21 ± 1.76	90.89 ± 6.25	92.76 ± 1.64
42-10-5	54.62 ± 0.5	57.8 ± 8.55	92.78 ± 2.45
42-10-10	55.59 ± 2.81	51.05 ± 1.6	65.1 ± 20.35
275-5-0	54.31 ± 2.07	95.15 ± 2.4	95.15 ± 2.4
275-5-5	51.6 ± 0.45	55.2 ± 6.13	83.87 ± 17.67
275-5-10	52.34 ± 0.5	50.33 ± 0.72	56.94 ± 11.64
275-10-0	54.52 ± 1.54	87.85 ± 4.26	89.86 ± 3.4
275-10-5	54.5 ± 1.81	54.11 ± 4.79	83.5 ± 17.88

increasing or decreasing the required number z_i of instances for a particular c_i . We chose a new number from $\{0, \dots, z_i - 1\} \cup \{z_i + 1, \dots, 10\}$ with equal probability. If this number was less than z_i , we replaced instances of concept c_i by random instances, if it was greater, we replaced random instances by instances of concept c_i . The minimal bag size in this dataset is $\sum_i z_i + 10|C|$, the maximal possible bag size is $20|C| + 10$. Accuracy results are given in Table 4. The parameters of the dataset are given as $\langle z_1..z_n \rangle - \langle l_r \rangle - \langle l_i \rangle$. For example dataset 42-5-5 requires exactly 4 instances of the first concept and 2 instances of the second concept in a positive bag, using 5 relevant and 5 irrelevant attributes.

The results for the count-based MI data (Table 4) show that the TLC method is able to learn this type of concept, even if a reasonable number of irrelevant attributes is involved. In datasets with a very high ratio of irrelevant attributes (dataset 275-5-10), even TLC fails, because the underlying concepts cannot be identified accurately enough. Attribute selection improves the performance of TLC, but only in some of the five runs, which leads to high variance. The results show that the MI SVM cannot learn count-based MI concepts; its performance is only slightly better than the default accuracy.

5.2 Musk Datasets

We have also evaluated the performance of TLC on the Musk datasets used by Dietterich et al. [1]. As described above, we used a boosted decision stump with 10 boosting iterations at the second level. At the first level, we used a standard pruned C4.5 tree [9]. However, performance improved after equal-width discretization based on ten intervals, representing the split points as binary attributes [10], and our results are based on the discretized data. We performed 10 runs of 10-fold cross-validation and give their average accuracy and standard deviation in Table 5. The results for the MI SVM are an average value of 1000 runs of randomly leaving out 10 bags and training the classifier on the remaining ones [2], using a γ -parameter of $10^{-5.5}$. The results for the DD algorithm were achieved by twenty runs of a 10-fold cross-validation [4]. Table 5 shows that TLC can successfully be applied to the Musk data.

Table 5. Results for musk-datasets

	DD	MI SVM	TLC without AS
musk1	88.9	86.4 \pm 1.1	88.69 \pm 1.64
musk2	82.5	88.0 \pm 1.0	83.13 \pm 3.23

6 Conclusions and Further Work

We have presented three different generalizations of the multi-instance assumption. Two-level classification is an elegant way to tackle these learning problems, and as our results show, it performs well on artificial data representing all three types of problems. A simple form of attribute selection increases the performance considerably in cases where a high ratio of irrelevant attributes makes it hard to discover the underlying instance-level concepts. On the Musk data, our algorithm performs comparably with state-of-the-art methods for this problem. Further work includes the application of a different clustering technique at the first level to structure the instance space and the application of our method to an image classification task.

Acknowledgments

Many thanks to Luc de Raedt and Geoff Holmes for enabling Nils to work on his MSc at Waikato University. Eibe Frank was supported by Marsden Grant 01-UOW-019. Xin Xu provided the implementation of the DD algorithm, and Thomas Gärtner the code of the MI-Kernel SVM.

References

1. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the Multiple Instance Problem with Axis-Parallel Rectangles. *Artificial Intelligence* **89** (1997) 31–71
2. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-Instance Kernels. In: Proc. 19th Int. Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2002) 179–186
3. Ramon, J., De Raedt, L.: Multi Instance Neural Networks. In: Proc. ICML Workshop on Attribute-Value and Relational Learning. (2000)
4. Maron, O., Lozano-Pérez, T.: A Framework for Multiple-Instance Learning. In: Advances in Neural Information Processing Systems. Volume 10., MIT Press (1998)
5. Zucker, J.D., Chevalyere, Y.: Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem. In: Internal Report, University of Paris 6. (2000)
6. De Raedt, L.: Attribute-Value Learning Versus Inductive Logic Programming: The Missing Links. In: Proc. 8th Int. Conf. on ILP, Springer (1998) 1–8
7. Kohavi, R., John, G.H.: Wrappers for Feature Subset Selection. *Artificial Intelligence* **97** (1997) 273–324
8. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. *Annals of Statistics* **28** (2000) 307–337
9. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
10. Frank, E., Witten, I.: Making Better Use of Global Discretization. In: Proc. 16th Int. Conf. on Machine Learning, Morgan Kaufmann (1999) 115–123