Neural Processing Letters (2006) 23:1–26
DOI 10.1007/s11063-005-2192-z

Adapting RBF Neural Networks to Multi-Instance Learning

MIN-LING ZHANG and ZHI-HUA ZHOU*
National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China. e-mail: zhangml@lamda.nju.edu.cn, zhouzh@nju.edu.cn

Abstract. In multi-instance learning, the training examples are *bags* composed of instances without labels, and the task is to predict the labels of unseen bags through analyzing the training bags with known labels. A bag is positive if it contains at least one positive instance, while it is negative if it contains no positive instance. In this paper, a neural network based multi-instance learning algorithm named RBF-MIP is presented, which is derived from the popular radial basis function (RBF) methods. Briefly, the first layer of an RBF-MIP neural network is composed of *clusters of bags* formed by merging training bags agglomeratively, where Hausdorff metric is utilized to measure distances between bags and between clusters. Weights of second layer of the RBF-MIP neural network are optimized by minimizing a sum-of-squares error function and worked out through singular value decomposition (SVD). Experiments on real-world multi-instance benchmark data, artificial multi-instance benchmark data and natural scene image database retrieval are carried out. The experimental results show that RBF-MIP is among the several best learning algorithms on multi-instance problems.

Key words. content-based image retrieval, Hausdorff distance, machine learning, multi-instance learning, neural networks, principle component analysis, radial basis function, singular value decomposition

1. Introduction

At present, roughly speaking, there are three frameworks for learning from examples [23]. That is, *supervised learning, unsupervised learning* and *reinforcement learning*. Supervised learning attempts to learn a concept for correctly labeling unseen examples, where the training examples are with labels. Unsupervised learning attempts to learn the structure of the underlying sources of examples, where the training examples are with no labels. Reinforcement learning attempts to learn a mapping from states to actions, where the examples are with no labels but with delayed rewards that could be viewed as delayed labels.

The notion of *multi-instance learning* was proposed by Dietterich et al. [12] in their investigation of drug activity prediction. In multi-instance learning, the training set is composed of many *bags* each containing many instances. If a bag contains at least one positive instance then it is labeled as a positive bag. Otherwise it is labeled as a negative bag. The labels of the training bags are known, but those

*Corresponding author.

of the training instances are unknown. The task is to learn something from the training set for correctly labeling unseen bags. Dietterich et al. [12] showed that learning methods ignoring the characteristics of multi-instance learning could not work well in this scenario. Due to its unique characteristics and extensive applicability, multi-instance learning has been regarded as a new learning framework parallel to supervised learning, unsupervised learning, and reinforcement learning [23].

When the notion of multi-instance learning was proposed, Dietterich et al. [12] indicated that a particular interesting issue in this area is to design multi-instance modifications for neural networks. In this paper, this problem is addressed in the way that a multi-instance neural network algorithm named RBF-MIP, i.e. Radial Basis Function (RBF) for Multi-Instance Problems, is proposed. As its name implied, RBF-MIP is derived from the popular RBF methods [6]. Briefly, the first layer of an RBF-MIP neural network is composed of *clusters of bags* formed by merging training bags agglomeratively, where Hausdorff metric [14, 31] is utilized to measure distances between bags and between clusters. Second layer weights of the RBF-MIP neural network are optimized by minimizing a sum-of-squares error function and worked out through singular value decomposition (SVD) [26]. Experiments on the drug activity prediction data, which is the only real-world benchmark test data for multi-instance learning at present, some artificial benchmark multi-instance data and natural scene image database retrieval, show that RBF-MIP is among the several top-ranked multi-instance learning methods. Furthermore, the performance of RBF-MIP is significantly better than that of BP-MIP [37], which is another neural network based multi-instance learner derived from the traditional Backpropagation algorithm [29].

The rest of this paper is organized as follows. In Section 2, the drug activity prediction problem is briefly introduced. In Section 3, previous works on multi-instance learning are reviewed. In Section 4, RBF-MIP is presented. In Section 5, the experimental results on various multi-instance learning problems are reported. Finally in Section 6, the main contribution of this paper is summarized and several issues for future work are indicated.

2. Drug Activity Prediction

Most drugs are small molecules working by binding to larger protein molecules such as enzymes and cell-surface receptors. The potency of a drug is determined by the degree of binding. For molecules qualified to make a drug, one of its low-energy shapes could tightly bind to the target area. While for molecules unqualified to make a drug, none of its low-energy shapes could tightly bind to the target area.

In the middle of 1990's, Dietterich et al. [12] investigated the problem of drug activity prediction. The goal was to endow learning systems with the ability of predicting if a new molecule was qualified to make some drug, through analyzing a collection of known molecules. As illustrated in Figure 1, the main difficulty of this problem is that each molecule may have many alternative low-energy shapes. However, biochemists

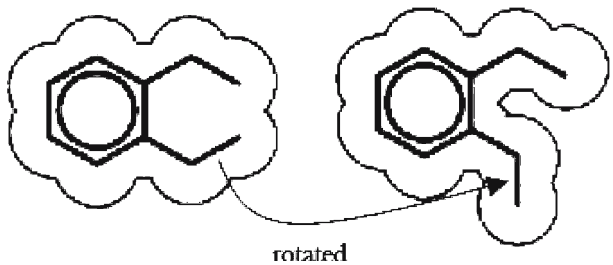

Figure 1. The shape of a molecule changes as it rotates an internal bond.

only know if a molecule is qualified to make a drug or not, instead of knowing which of its alternative low-energy shapes is responsible for the qualification.

An intuitive solution is to use supervised algorithms by regarding all the low-energy shapes of the molecules qualified to make the drug as positive training examples, while regarding all the low-energy shapes of the molecules unqualified to make the drug as negative training examples. However, as shown by Dietterich et al. [12], such a method can hardly work because there may be a great many of false positive examples.

In order to solve this problem, Dietterich et al. [12] regarded each molecule as a bag, and regarded the alternative low-energy shapes of the molecule as the instances in the bag, thereby formulated multi-instance learning. In order to represent the shapes, the molecules were placed in a standard position and orientation and then a set of 162 rays emanating from the origin was constructed so that the molecular surface was sampled approximately uniformly, as illustrated in Figure 2. There were also four features that represented the position of an oxygen atom on the molecular surface. Therefore each instance in the bags was represented by a 166-dimensional numerical feature vector.

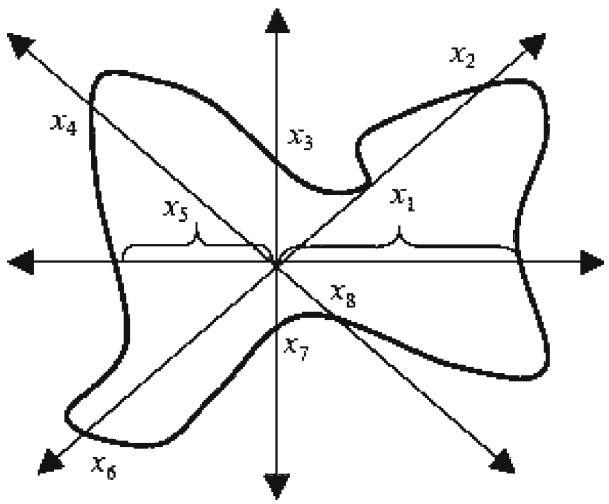

Figure 2. The ray-based representation of the molecular shape.

Based on such a representation, Dietterich et al. [12] proposed three Axis-Parallel Rectangle (abbreviated as APR) algorithms, which attempt to search for appropriate axis-parallel rectangles constructed by the conjunction of the features. Their experiments showed that the iterated-discrim APR algorithm achieves the best result on the *Musk* data, which is the only real-world benchmark test data for multi-instance learning until now, while the performance of popular supervised learning algorithms such as C4.5 decision tree and Backpropagation neural network is very poor. Note that Dietterich et al. [12] indicated that since the APR algorithms were optimized to the *Musk* data, the performance of iterated-discrim APR might be the upper bound for this data.

It should be mentioned that multi-instance problems are not unique to drug activity prediction. In fact, they reveal themselves in many real-world applications [21, 30]. But unfortunately, machine learning community had not paid special attention to such kinds of problems until Dietterich et al. [12].

3. Previous Work

Long and Tan [22] initiated the investigation of the PAC-learnability of axis-parallel rectangles under the multi-instance learning framework. Resorting to P-concept [20], they described a high-order polynomial-time theoretical algorithm and showed that if the instances in the bags are independently drawn from product distribution, then the APR is PAC-learnable. Auer et al. [5] showed that if the instances in the bags are not independent then APR learning under the multi-instance learning framework is NP-hard. Moreover, they also presented a theoretical algorithm without requiring product distribution and with reduced time and sample complexity than that of Long and Tan's algorithm. Later, this theoretical algorithm was transformed into a practical algorithm named MULTINST [4]. Blum and Kalai [8] gave a reduction from the problem of PAC-learning under the multi-instance learning framework to PAC-learning with one-sided or two-sided random classification noise. With the help of Statistical-Query Model [19], they also presented a theoretical algorithm with smaller sample complexity than that of Auer et al.'s algorithm. Goldman et al. [16] presented an efficient on-line agnostic multi-instance learning algorithm for learning the class of constant-dimension geometric patterns, which tolerated both noise and concept shift. Later, this algorithm was extended so that it could deal with real-valued output [17].

As reviewed above, theoretical machine learning community has contributed much to multi-instance learning. But since most of their results are obtained under the restrictive assumption that each bag must contain the *same* number of *independent* instances, which is usually not the case in real problems, those results are hard to be used directly in real-world applications.

Fortunately, some practical algorithms for multi-instance learning have been presented by the applied machine learning community. A representative practical multi-instance learning algorithm is Diverse Density proposed by Maron and

Lozano-Pérez [24], which has been applied to several applications including learning a simple description of a person from a series of images [24], stock prediction [24], natural scene classification [25], and content-based image retrieval (CBIR) [32, 34]. Wang and Zucker [31] extended k-nearest neighbor algorithm for multi-instance learning through adopting Hausdorff distance. Two algorithms, i.e. Bayesian-kNN and Citation-kNN, were presented. Bayesian-kNN labels a bag through analyzing its neighboring bags with Bayes theory. Citation-kNN borrows the notion of *citation* of science references, which labels a bag through analyzing not only its neighboring bags but also the bags that regard the concerned bag as a neighbor. Ruffo [28] presented a multi-instance version of C4.5 decision tree named Relic and applied it to data mining area. Later, Chevaleyre and Zucker [9] derived ID3-MI and RIPPER-MI, which are multi-instance versions of decision tree algorithm ID3 and rule learning algorithm RIPPER, where the key is a multi-instance entropy and a multiple-instance coverage function respectively. In 2002, Zhou and Zhang [37] developed a multi-instance neural network named BP-MIP, which extended the popular Backpropagation [29] algorithm with a global error function defined at the level of bags instead of at the level of instances. Gärtner et al. [15] developed a kernel on multi-instance data that can be shown to separate positive and negative bags under natural assumptions. Furthermore, they proposed another more efficient kernel that can easily deal with huge bag sizes. Zhang and Goldman [33] proposed EM-DD, which combines the EM [10] and Diverse Density algorithms to solve multi-instance problems. One year later, Andrews et al. [3] presented two new formulations of multi-instance learning as a maximum margin problem and applied them to applications such as drug activity prediction, automated image indexing and document categorization. Zhou and Zhang [38] utilized ensemble learning paradigms to solve multi-instance learning problems and obtained the best result up to now on a benchmark test.

In the early years of the research of multi-instance learning, most works are on multi-instance classification with discrete-valued outputs. Recently, multi-instance regression with real-valued outputs begins to attract the attention of some researchers. Ray and Page [27] showed that the general formulation of multi-instance regression is NP-hard, and proposed an EM-based multi-instance regression algorithm. Dooly et al. [13] confirmed the conclusion drawn by Ray and Page and further proved that learning from real-valued multi-instance examples is as hard as learning DNF. Amar et al. [2] extended Diverse Density and Citation-kNN for multi-instance regression. Moreover, they designed some method for artificially generating data sets for multi-instance regression. Their data sets are available from http://www.cs.wustl.edu/∼sg/multi-inst-data/.

Multi-instance learning has even attracted the attention of the Inductive Logic Programming community. De Raedt [11] showed that multi-instance problems could be regarded as a bias on inductive logic programming. He also suggested that the multi-instance paradigm could be the key between the propositional and relational representations, being more expressive than the former, and much easier

to learn than the latter. Zucker and Ganascia [39, 40] presented REPEAT, an ILP system based on an ingenious bias which firstly reformulate the relational examples in a multi-instance database, and then induces the final hypothesis with a multi-instance learner. Recently, Alphonse and Matwin [1] successfully employed multi-instance learning to help relational learning. At first, the original relational learning problem is approximated by a multi-instance problem. The resulting data is fed to feature selection techniques adapted from propositional representations. Then the filtered data is transformed back to relational representation for a relational learner. In this way, the expressive power of relational representation and the ease of feature selection on propositional representation are gracefully combined. This work confirms that multi-instance learning could act as a bridge between propositional and relational learning.

It is worth noting that when Dietterich et al. [12] coined the term *multi-instance learning*, they indicated that a particular interesting issue in this area is to design multi-instance modifications for decision trees, neural networks, and other popular machine learning algorithms. During recent years, multi-instance version of decision trees [9, 28], rule learning algorithms [9], lazy learning algorithms [31], and support vector machines [3, 15], have already been presented. Especially, a neural network based multi-instance learner named BP-MIP [37] derived from the traditional Backpropagation [29] neural network has also been proposed. But unfortunately, although BP-MIP performs comparably to many existing multi-instance learners, it is not so good as several algorithms derived from other popular machine learning algorithms, such as multi-instance decision tree named Relic [28], multi-instance lazy learning algorithm named Citation-kNN and multi-instance support vector machine named MI SVM [15], etc. Based on the above observation, another multi-instance neural network named RBF-MIP, which is derived from the popular RBF neural network [6], is proposed in the following section.

4. RBF-MIP

RBF is a popular neural network learning algorithm where the activation of a hidden unit is determined by the distance between the input vector and a prototype vector. Usually, a two-stage training procedure is used to train an RBF neural network, i.e. the parameters governing the basis functions (corresponding to hidden units) are determined using unsupervised methods while the final-layer weights are obtained by the solution of a linear problem. Detailed description and theoretical foundations of RBF neural networks can be found in the literature [6].

Suppose the training set is composed of N bags, i.e. $\{B_1, B_2, \ldots, B_N\}$, the ith bag is composed of M_i instances, i.e. $\{B_{i1}, B_{i2}, \ldots, B_{iMi}\}$, each is a p-dimensional feature vector, e.g. the jth instance of the ith bag is $[B_{ij1}, B_{ij2}, \ldots, B_{ijp}]^{\mathrm{T}}$. The desired output of a positive training bag is 1, while that of a negative training bag is 0.

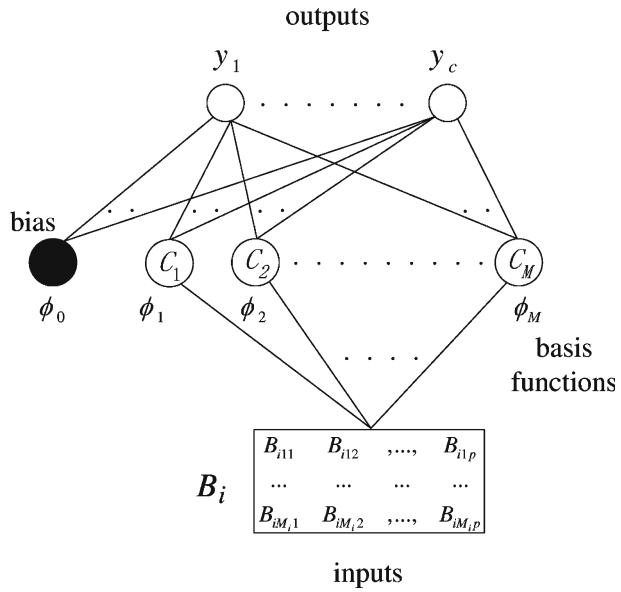

Figure 3. Typical architecture of an RBF-MIP neural network.

Figure 3 illustrates the typical architecture of an RBF-MIP neural network. As shown in Figure 3, there are two main architectural differences between the RBF-MIP and the standard RBF neural networks. Firstly, the input of the RBF-MIP neural network corresponds to a bag containing several vectors (instances) instead of a single vector of the standard RBF neural network. Secondly, for the RBF-MIP neural network, each node $C_i (1 \leqslant i \leqslant M)$ in the first layer corresponds to a cluster of training bags where $\bigcup_{i=1}^{M} C_i = \{B_1, B_2, \ldots, B_N\}$ and $C_i \bigcap_{i \neq j} C_j = \emptyset$, while that of the standard RBF neural network is a prototype vector determining the center of basis function ϕ_i. The second layer weights $w_{jk} (0 \leqslant j \leqslant M, 1 \leqslant k \leqslant c)$ are shown as lines from the basis functions to the output units, and the biases are shown as weights w_{0k} from an extra "basis function" ϕ_0 whose output is fixed at 1.

Similar to the training procedure used for traditional RBF neural network, a two-stage training procedure is also employed to train the RBF-MIP neural network. In the first stage, its first layer is automatically constituted by merging training bags agglomeratively, where Hausdorff metric is utilized to measure distances between bags and between clusters. In the second stage, weights of its second layer are optimized by minimizing a sum-of-squares error function and worked out through SVD. The above two stages are scrutinized in Sections 4.1 and 4.2 respectively.

4.1. FIRST LAYER CLUSTERING

As shown in the literature [6], for traditional RBF neural network, clustering algorithms such as *K-means or self-organizing feature map* are usually employed to

partition the training instances (vectors) into a number of disjoint subsets, i.e. clusters of instances. After that, the centers of their basis functions in the first layer are determined by the means of the training instances in each subset. While in multi-instance learning paradigm, the training set is composed of *bags* each containing many instances instead of individual instances in traditional supervised learning paradigm. Thus, an intuitive way to fit RBF neural network into multi-instance learning paradigm is to form *clusters of bags* instead of *clusters of instances* in the first layer.

In order to form clusters of bags in the first layer, some kinds of distance metrics should be utilized to measure the distances between bags and between clusters. In this paper, two types of Hausdorff metric, i.e. maximal Hausdorff distance [14] and minimal Hausdorff distance [31] are adopted to fulfill this objective, since Hausdorff metric has already shown its successful application in multi-instance learning paradigm [31]. Formally, given two sets of objects $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$, the maximal and minimal Hausdorff distances are defined as Equations (1) and (2):

$$\max \mathrm{H}(A, B) = \mathrm{Max}\left\{ \underset{a\in A}{\mathrm{Max}} \underset{b\in B}{\mathrm{Min}}\{\mathrm{dist}(a, b)\}, \underset{b\in B}{\mathrm{Max}} \underset{a\in A}{\mathrm{Min}}\{\mathrm{dist}(b, a)\} \right\}, \tag{1}$$

$$\min \mathrm{H}(A, B) = \underset{a\in A, b\in B}{\mathrm{Min}}\{\mathrm{dist}(a, b)\}. \tag{2}$$

As shown in the above equations, it is worth noting that both max H$(*, *)$ (maximal Hausdorff distance) and min H$(*, *)$ (minimal Hausdorff distance) are capable of measuring distances between bags (sets of instances) and between clusters (sets of bags). In detail, when both A and B are sets of numerical vectors and function dist$(*, *)$ is Euclidean distance, max H$(*, *)$ and min H$(*, *)$ can be used to measure distance between sets of instances (bags). Interestingly, on the other hand, when both A and B correspond to sets of bags and function dist$(*, *)$ is either max H$(*, *)$ or min H$(*, *)$, i.e. measuring distance between bags using maximal or minimal Hausdorff distance, Equations (1) and (2) can also be utilized to calculate distance between sets of bags (clusters).

For example, let $C_1 = \{B_1, B_2\}$ and $C_2 = \{B_3\}$ be two clusters of bags, where $B_1 = \{1, 3, 7\}$, $B_2 = \{4, 9\}$ and $B_3 = \{6, 7, 11\}$ are three different bags each containing several one dimensional instances. If minimal Hausdorff distance is used to measure distance between bags and function dist$(*, *)$ shown in Equation (2) is Euclidean distance, then the distance between bags B_1 and B_2 can be calculated as $\mathrm{bag\_dist}(B_1, B_2) = \min \mathrm{H}(B_1, B_2) = \mathrm{Min}_{a\in B_1, b\in B_2} |a - b| = \mathrm{Min}\{|1 - 4|, |1 - 9|, |3 - 4|, |3 - 9|, |7 - 4|, |7 - 9|\} = 1$. Similarly, $\mathrm{bag\_dist}(B_1, B_3) = 0$ and $\mathrm{bag\_dist}(B_2, B_3) = 2$ can also be verified by the readers. Furthermore, when maximal Hausdorff distance is used to measure distance between clusters and function dist$(*, *)$ shown in Equation (1) is minimal Hausdorff distance, the distance between clusters C_1 and C_2 can be calculated as clu_dist

$(C_1, C_2) = \max \mathrm{H}(C_1, C_2) = \mathrm{Max}\{\mathrm{Max}_{A \in C_1} \mathrm{Min}_{B \in C_2}\{\min \mathrm{H}(A, B)\}, \mathrm{Max}_{B \in C_2} \mathrm{Min}_{A \in C_1}\{\min \mathrm{H}(B, A)\}\} = \mathrm{Max}\{\mathrm{Max}\{0, 2\}, \mathrm{Max}\{0\}\} = \mathrm{Max}\{2, 0\} = 2.$

In the above example, the distance between bags is measured using minimal Hausdorff metric while the distance between clusters is measured using maximal Hausdorff metric. However, it is noteworthy that both Hausdorff metrics can be utilized either as distance measure between bags or as distance measure between clusters. Based on this, procedure for the first layer clustering of RBF-MIP is shown in Figure 4, which is in fact the well-known agglomerative clustering algorithm specifically adapted to the multi-instance learning framework.

4.2. SECOND LAYER OPTIMIZATION

When the above stage of first layer clustering is accomplished, second layer weights of an RBF-MIP neural network are obtained by the solution of a linear problem, where the involved procedure is very similar to the one used to train traditional RBF neural network [6]. In detail, the second layer weights of an RBF-MIP neural network are optimized by minimizing the following sum-of-squares error function:

$$E = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{c} \{y_k(B_n) - t_k^n\}^2, \tag{3}$$

where t_k^n is the target value for output unit k when the network is presented with input bag B_n. The corresponding actual output $y_k(B_n)$ is determined as follows:

FLC(M, S, clu_dist)

Inputs:

M – number of remaining clusters in the first layer;

S – training set $\{B_1, B_2, \ldots, B_N\}$;

clu_dist$(*, *)$ – distance measure between clusters of bags (which uses bag_dist$(*, *)$ to measure distance between bags);

Outputs:

C_i – clusters of training bags. $(1 \leq i \leq M)$

Process:

Begin with one cluster per training bag $(C_1 = \{B_1\}, \ldots, C_N = \{B_N\})$;

While there are more than M clusters

Merge the two clusters C_i, C_j which minimize clu_dist(C_i, C_j);

Figure 4. Procedure for the first layer clustering of RBF-MIP.

$$y_k(B_n) = \sum_{j=0}^{M} w_{jk}\phi_j(B_n), \tag{4}$$

where ϕ_0 is an extra 'basis function' with activation value fixed at 1. For the case of Gaussian basis functions we have:

$$\phi_j(B_n) = \exp\left(-\frac{(\text{clu\_dist}(\{B_n\}, C_j))^2}{2\sigma_j^2}\right) \quad (1 \leqslant j \leqslant M), \tag{5}$$

where clu_dist$(\{B_n\}, C_j)$ calculates the distance between bag B_n and cluster C_j by taking the input bag as a cluster of its own, while some form of bag_dist$(*, *)$ is used at the same time to measure distance between bags as shown in Figure 4. The standard deviation σ_j is a parameter whose value controls the smoothness property of the basis function ϕ_j. For traditional RBF neural network, one heuristic approach to determining the standard deviations is to choose all the σ_j to be equal and to be given by some multiple of the average distance between the basis function centers [6]. Thus, in order to fit this heuristic approach into multi-instance learning framework, each standard deviation used in the basis functions of RBF-MIP is set to take the same value σ determined by the average distance between every pair of clusters using Equation (6), where μ is a scaling factor.

$$\sigma = \mu \times \left(\frac{\sum_{i=1}^{M-1}\sum_{j=i+1}^{M} \text{clu\_dist}(C_i, C_j)}{M(M-1)/2}\right). \tag{6}$$

Note that the Gaussian basis functions in Equation (5) are not normalized, since any overall factors can be absorbed into the weights in Equation (4) without loss of generality. Substituting Equation (4) into Equation (3), the sum-of-squares error function can be rewritten as:

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left\{\sum_{j=0}^{M} w_{jk}\phi_j(B_n) - t_k^n\right\}^2. \tag{7}$$

Differentiating this expression with respect to w_{jk} and setting the derivative to zero gives the normal equations for the least-squares problem in the following form:

$$\sum_{n=1}^{N}\left\{\sum_{j'=0}^{M} w_{j'k}\phi_{j'}(B_n) - t_k^n\right\}\phi_j(B_n) = 0 \quad (0 \leqslant j \leqslant M, 1 \leqslant k \leqslant c). \tag{8}$$

In order to find a solution to Equation (8) it is convenient to write it in a matrix notation to give:

$$(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})\mathbf{W} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{T}. \tag{9}$$

Here $\mathbf{\Phi}$ has dimensions $N \times (M+1)$ and elements $\phi_j(B_n)$, $\mathbf{W}$ has dimensions $(M+1) \times c$ and elements w_{jk}, and $\mathbf{T}$ has dimensions $N \times c$ and elements t_k^n. The

matrix $\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$ in Equation (9) is a square matrix of dimensions $(M+1)\times(M+1)$. Provided that it is non-singular we may invert it to obtain a solution to Equation (9) which can be written in the form:

$$\mathbf{W} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{T}. \tag{10}$$

However, in practice, the direct solution of the normal equations can lead to numerical difficulties due to the possibility of $\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$ being singular or nearly singular. Fortunately, such problems can be conveniently resolved by using the technique of SVD [26] to find a solution for the weights. Thus, the second layer weights can be found by fast, linear matrix inversion techniques.

It is worth noting that the appropriate parameter configuration of an RBF-MIP neural network, i.e. M (the number of remaining clusters in the first layer) and μ (the scaling factor), could be chosen based on the training data. For instance, given a set of candidate configurations, the performance of each configuration could be evaluated through performing 10-fold cross validation on the training data or be estimated on a validation data set separated from the training data, where the configuration with the best performance is chosen and an RBF-MIP neural network is then trained on the entire training set using this configuration to predict the labels of unseen bags.

5. Experiments

5.1. MUSK DATA SETS

The *Musk* data is the only real-world benchmark test data for multi-instance learning at present. The data is generated by Dietterich et al. in the way described in Section 2. There are two data sets, i.e. *Musk1* and *Musk2*, both of which are publicly available from the UCI Machine Learning Repository [7]. *Musk1* contains 47 positive bags and 45 negative bags, and the number of instances contained in each bag ranges from 2 to 40. *Musk2* contains 39 positive bags and 63 negative bags, and the number of instances contained in each bag ranges from 1 to 1,044. Detailed information on the *Musk* data is tabulated in Table 1.

Leave-one-out test is performed on both data sets. In detail, for N bags, one bag is used to test while the others are used to train an RBF-MIP neural network in a loop of N iterations. In each iteration, in order to automatically determine the

Table 1. The *Musk* data (72 molecules are shared in both data sets).

		Bags				Instances per bag		
Data set	Dim.	Total	Musk	Non-musk	Instances	Min	Max	Ave.
Musk1	166	92	47	45	476	2	40	5.17
Musk2	166	102	39	63	6,598	1	1,044	64.69

parameter configuration of the algorithm, i.e. M (the number of remaining clusters in the first layer) and μ (the scaling factor), the original training set (denoted as *ori_set*) is further divided into two portions. Concretely, a fraction of *ori_set* is randomly selected to form the validation set (denoted as *vali_set*) while the remaining portion of *ori_set* (denoted as *train_set*) is used for training. Thus, given a set of candidate parameter configurations, RBF-MIP neural networks with one output unit are trained using each candidate parameter configuration on *train_set* according to the two-stage procedure described in Section 4 and then tested on the *vali_set*. After that, the parameter configuration with which the RBF-MIP neural network achieves the highest predictive accuracy on *vali_set* is selected and then employed to train a new RBF-MIP neural network on the original training set *ori_set*. For *Musk1*, 20% random fraction of *ori_set* is used to form the *vali_set* while a parameter candidate set with 28 different configurations is used, i.e. M ranges from 40 to 70 with an interval of 5 and μ ranges from 0.3 to 0.6 with an interval of 0.1. For *Musk2*, 30% random fraction of *ori_set* is used to form the *vali_set* while the parameter candidate set is the same as that used in *Musk1*. The iterations are repeated in the way that each bag in the data set has been used as the test bag once. When a trained RBF-MIP network is used in prediction, a bag is positively labeled if and only if the output of the network is not less than 0.5. At the end of the loop, the final predictive accuracy is calculated as the total number of correctly labeled test bags divided by N.

On both *Musk1* and *Musk2*, 10 times of leave-one-out tests are conducted for each distance metric configuration of $\langle \text{bag\_dist}(*,*), \text{clu\_dist}(*,*) \rangle$. The corresponding average predictive accuracy and standard deviation is reported in Table 2.

As shown by Table 2, it is obvious that when the distance metric between clusters is fixed (1st line vs. 3rd line, 2nd line vs. 4th line), using min $H(*,*)$ (minimal Hausdorff distance) to measure distance between bags will result in better performance than using max $H(*,*)$ (maximal Hausdorff distance). On the other hand, when the distance metric between bags is fixed (1st line vs. 2nd line, 3rd line vs. 4th line), using max $H(*,*)$ to measure distance between clusters will lead to better results than using min $H(*,*)$. RBF-MIP achieves highest predictive accuracy on both *Musk1* and *Musk2* when min $H(*,*)$ and max $H(*,*)$ are utilized respectively to measure distances between bags and between clusters. In the rest of this

Table 2. The performance of RBF-MIP (% correct±std. deviation) on the *Musk* data.

Distance metric	*Musk 1*	*Musk 2*
$\langle$ min $H(*,*)$, min $H(*,*)$ $\rangle$	80.9 ± 3.0	85.0 ± 2.5
$\langle$ min $H(*,*)$, max $H(*,*)$ $\rangle$	90.2 ± 2.6	88.0 ± 3.5
$\langle$ max $H(*,*)$, min $H(*,*)$ $\rangle$	73.2 ± 1.9	80.5 ± 2.6
$\langle$ max $H(*,*)$, max $H(*,*)$ $\rangle$	77.3 ± 1.5	82.1 ± 1.6

Table 3. Comparison of the performance (% correct ± std. deviation) on the *Musk1* data.

Algorithm	*Musk1*	Evaluation
MI SVM [15]	92.4	LOO
Iterated-discrim APR [12]	92.4	10CV
Citation-*k*NN [31]	92.4	LOO
RBF-MIP-PCA	91.3 ± 1.6	LOO
GFS elim-kde APR [12]	91.3	10CV
RBF-MIP	90.2 ± 2.6	LOO
GFS elim-count APR [12]	90.2	10CV
Bayesian-*k*NN [31]	90.2	LOO
Diverse Density [24]	88.9	10CV
BP-MIP-PCA [35]	88.0	LOO
RIPPER-MI [9]	88.0	N/A
mi-SVM [3]	87.4	10CV
EM-DD [33]	84.8	10CV
BP-MIP [37]	83.7	LOO
Relic [28]	83.7	10CV
MI-SVM [3]	77.9	10CV
MULTINST [4]	76.7 ± 4.3	10CV
Backpropagation [12]	75.0	10CV
C4.5 [12]	68.5	10CV

paper, all the reported experimental results of RBF-MIP were obtained with this type of distance metric configuration.

Tables 3 and 4 compare the performance of RBF-MIP on both *Musk* data sets with those reported in the literatures, where the value following "±" shows the available standard deviation. Unfortunately, only very few literatures have reported the standard deviations of their corresponding learning algorithms. The empirical results shown in the tables have either been obtained by multiple 10-fold cross-validation runs (10CV) or by leave-one-out estimation (LOO).[1]

Note that Zhou and Zhang [38] have showed that ensembles of multi-instance learners could achieve better results than single multi-instance learners. However, considering that RBF-MIP is a single multi-instance learner, the performance of ensembles of multi-instance learners are not included in the tables for fair comparison.

On the other hand, both Dietterich et al.'s APR algorithms and Maron and Lozano-Pérez's Diverse Density algorithm [24] employed some feature selection mechanisms. Furthermore, the technique of principle component analysis (PCA) [18] has also been used to improve the performance of BP-MIP [37], where an enhanced version of this algorithm named BP-MIP-PCA [35] is proposed.

[1]As what has been pointed out by Andrews et al. [3], the EM-DD algorithm described in [33] seems to use the test data to select the optimal solution obtained from multiple runs of the algorithm. Thus, the experimental results of EM-DD shown in Tables 3 and 4 are the results given by Andrews et al. [3].

Table 4. Comparison of the performance (% correct ± std. deviation) on the *Musk2* data.

Algorithm	*Musk2*	Evaluation
MI SVM [15]	92.2	LOO
RBF-MIP-PCA	90.1 ± 1.7	LOO
Iterated-discrim APR [12]	89.2	10CV
RBF-MIP	88.0 ± 3.5	LOO
Relic [28]	87.3	10CV
Citation-*k*NN [31]	86.3	LOO
EM-DD [33]	84.9	10CV
MI-SVM [3]	84.3	10CV
MULTINST [4]	84.0 ± 3.7	10CV
mi-SVM [3]	83.6	10CV
BP-MIP-PCA [35]	83.3	LOO
Diverse Density [24]	82.5	10CV
Bayesian-*k*NN [31]	82.4	LOO
BP-MIP [37]	80.4	LOO
GFS elim-kde APR [12]	80.4	10CV
RIPPER-MI [9]	77.0	N/A
GFS elim-count APR [12]	75.7	10CV
Backpropagation [12]	67.7	10CV
C4.5 [12]	58.8	10CV

In this paper, PCA is also embedded into RBF-MIP to yield better results on the *Musk* data sets. Briefly speaking, PCA is one of the most popular methods for irrelevant feature reduction, which is usually employed to discover the intrinsic dimensionality of a data set based on the covariance matrix **R** computed from the data. The q eigenvectors corresponding to the q largest eigenvalues of **R** define a linear transformation matrix **T**, which projects the original p-dimensional space into a q-dimensional space in which the features are uncorrelated. In this paper, for both *Musk1* and *Musk2*, the original 166-dimensional feature spaces are transformed into new 100-dimensional feature spaces by PCA, where the experimental setups are the same as used above. The performance of RBF-MIP combined with PCA (i.e. RBF-MIP-PCA) is also reported in Tables 3 and 4.

Tables 3 and 4 show that, RBF-MIP is among the top-ranked learning algorithms on both *Musk1* and *Musk2*. Especially, the performance of RBF-MIP is significantly better than that of BP-MIP [37], i.e. another neural network based multi-instance learner, on the *Musk* data. By incorporating the particular feature selection strategy of PCA, the performance of RBF-MIP is further improved and RBF-MIP-PCA also outperforms BP-MIP-PCA [35]. In addition, RBF-MIP has some advantages compared with other multi-instance learners. For example, Dietterich et al.'s APR algorithms [12] were specially designed for the *Musk* data, while RBF-MIP is a general algorithm so that its applicability is better than that of the APR algorithms. More important, through adopting the same notations and algorithm description in Section 4 and then converting the discrete-valued

output of each bag into the corresponding real-valued one, RBF-MIP is easy to be adapted for multi-instance regression problems.

Tables 3 and 4 also indicate that the performance of all the multi-instance learning methods is better than that of Backpropagation and C4.5, which is especially obvious on *Musk2* that is more difficult to learn than *Musk1*. This observation supports Dietterich et al.'s claim [12] that traditional supervised learning methods can hardly work well on multi-instance problems because they have not incorporated the characteristics of multi-instance learning.

It is noteworthy that, in multi-instance learning area, it is always the best performance of each algorithm been reported when making comparison. For instance, the experimental results of MI SVM [15] and Iterated-discrim APR [12] reported in Tables 3 and 4 are the best performance of each algorithm out of 26 and more than 50 different parameter configurations respectively. While in Tables 3 and 4, the reported results of RBF-MIP are not obtained by evaluating many different parameter configurations and then picking the best one. However, the parameters used to train RBF-MIP neural network, i.e. M (the number of remaining clusters in the first layer) and μ (the scaling factor), are automatically determined using the training data. Therefore, to make a fair comparison, the performance of RBF-MIP is also evaluated under a number of (36) different parameter configurations on both data sets and the best performance of RBF-MIP is recorded. In detail, M ranges from 40 to 80 with an interval of 5 and μ ranges from 0.3 to 0.6 with an interval of 0.1. Figures 5 and 6 illustrate the predictive accuracy of RBF-MIP with each parameter configuration on *Musk1* and *Musk2* respectively, where leave-one-out test is used to evaluate the performance. The horizontal axis indicates the number of remaining clusters in the first layer. For *Musk1* (as shown in Figure 5), the best performance of the RBF-MIP neural network is 94.6% (the best performance on *Musk1* shown in Table 3 is 92.4%), which is obtained with 65 clusters remaining in the first layer and the value of μ set to be 0.4. For *Musk2* (as shown in Figure 6), the best performance of the RBF-MIP neural network is 92.2% (the best performance on *Musk2* shown in Table 4 is also 92.2%), which is obtained with 70 clusters remaining in the first layer and the value of μ set to be 0.3. These results suggest that the rank of RBF-MIP among multi-instance learning methods may be even higher than that was shown in Tables 3 and 4.

5.2. ARTIFICIAL DATA SETS

In 2001, Amar et al. [2] presented a method for creating artificial multi-instance data. This method generates an artificial receptor at first. Then, artificial molecules with several instances per bag are generated, with each feature value considered as the distance from the origin to the molecular surface when all molecules are in the same orientation. Each feature has a *scale factor* to represent its importance in the binding process. The binding energies between the artificial molecules and receptor are calculated based on the *Lennard–Jones potential* for intermolecular interactions.

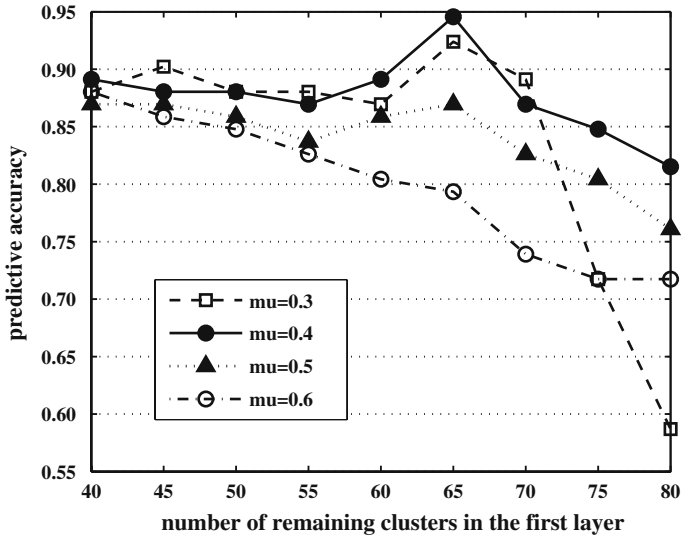

Figure 5. The predictive accuracy of RBF-MIP on *Musk1* changes as the number of remaining clusters increasing.

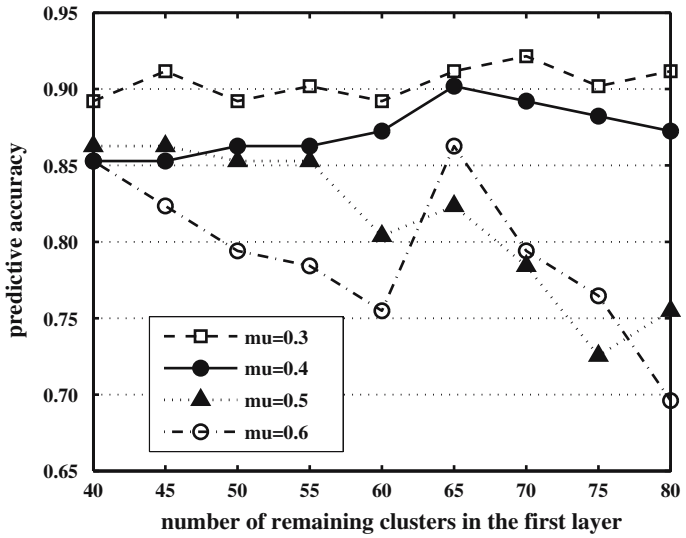

Figure 6. The predictive accuracy of RBF-MIP on *Musk 2* changes as the number of remaining clusters increasing.

The artificial data sets are named as LJ-$r.f.s$ where r is the number of relevant features, f is the number of features, and s is the number of different scale factors used for the relevant features. To partially mimic the *Musk* data, some data sets only use labels that are not near 1/2 (indicated by the 'S' suffix) and all scale factors for the relevant features are randomly selected between [0.9, 1]. Note that these data sets are mainly designed for multi-instance regression, but they can also

be used for multi-instance classification through rounding the real-valued label to 0 or 1.

Leave-one-out test is performed on four artificial data sets, i.e. LJ-160.166.1, LJ-160.166.1-S, LJ-80.166.1, and LJ-80.166.1-S. Each data set contains 92 bags. In detail, for N bags, one bag is used to test while the others are used to train an RBF-MIP neural network in a loop of N iterations. In each iteration, the same mechanism used in the experimental procedure of the *Musk* data is also used to automatically determine the parameter configuration. Concretely, for each artificial data set, 20% random fraction of the original training set is used to form the validation set while a parameter candidate set with 28 different configurations is used, i.e. M ranges from 40 to 70 with an interval of 5 and μ ranges from 0.3 to 0.6 with an interval of 0.1. The parameter configuration with which the RBF-MIP neural network achieves the lowest squared loss on the validation set is selected and then employed to train a new RBF-MIP neural network on the original training set. The iterations are repeated in the way that each bag in the data set has been used as the test bag once. At the end of the loop, the final squared loss and predictive error are both calculated as the average results of all test bags.

Ten times of leave-one-out tests are conducted for each artificial data set. The performance of RBF-MIP is compared with those of BP-MIP, Diverse Density and Citation-kNN, where the performance of BP-MIP and those of Diverse Density and Citation-kNN are reported in the literatures [37] and [2] respectively. The average predictive error and squared loss are shown in Tables 5 and 6 respectively, where the value following "±" is the standard deviation which is only available for RBF-MIP.

Tables 5 and 6 show that, with respect to both the predictive error and squared loss, the performance of RBF-MIP is worse than that of Citation-kNN [31], but

Table 5. Comparison of the predictive error (% error±std. deviation) on the artificial data sets.

Data set	RBF-MIP	BP-MIP	Diverse density	Citation-kNN
LJ-160.166.1	5.1 ± 0.7	16.3	23.9	4.3
LJ-160.166.1-S	1.1 ± 0.7	18.5	0.0	0.0
LJ-80.166.1	6.6 ± 0.8	18.5	N/A	8.6
LJ-80.166.1-S	18.5 ± 1.2	18.5	53.3	0.0

Table 6. Comparison of the squared loss (loss±std. deviation) on the artificial data sets.

Data set	RBF-MIP	BP-MIP	Diverse density	Citation-kNN
LJ-160.166.1	0.0108 ± 0.0001	0.0398	0.0852	0.0014
LJ-160.166.1-S	0.0075 ± 0.0002	0.0731	0.0052	0.0022
LJ-80.166.1	0.0167 ± 0.0005	0.0487	N/A	0.0109
LJ-80.166.1-S	0.0448 ± 0.0042	0.0752	0.1116	0.0025

it is apparently better than that of BP-MIP [37]. Furthermore, compared with the performance of Diverse Density [24], that of RBF-MIP is apparently better on LJ-160.166.1 and LJ-80.166.1-S, and it is comparable on LJ-160.166.1-S. The above results reveal that, although RBF-MIP is worse than Citation-kNN in multi-instance regression, it is better than BP-MIP and Diverse Density in both multi-instance classification and multi-instance regression.

5.3. NATURAL SCENE IMAGE DATABASE RETRIEVAL

In CBIR, the query, i.e. the example image posed by the user is actually ambiguous and difficult to be perceived. For instance, suppose a user poses the image shown in Figure 7 and asks the system to retrieval "similar" images from the database. This kind of query is rather ambiguous since the query can be regarded as "river", "mountains", "clouds", "trees", etc, while it is hard to ask the user precisely specify which one he or she really wants. However, if the query can be processed as an image bag that preserves original semantic meanings of the image, then the ambiguity can be tackled by multi-instance learning techniques. Briefly speaking, the query images posed by the user are firstly transformed into corresponding positive and negative bags by certain image bag generator. Then the system can learn what the user requires (i.e. the target concept) from those training bags with multi-instance learning algorithms. Finally, the images in the database are sorted according to the learned target concept and returned to the user by the system.

Several multi-instance learning based CBIR systems have been developed [25, 32, 34], where many image bag generators have been proposed and several multi-instance learning algorithms such as Diverse Density and EM-DD have been used to learn and retrieval images from the database. In this paper, the performance

Figure 7. A sample query image.

of RBF-MIP is further evaluated by retrieving images from a natural scene image database.

An image database consisting of 2,000 images is used in the experiments, which includes 400 images from each of the five natural scene image types: *desert, mountains, sea, sunset*, and *trees*. Each image is transformed into an image bag by the popular SBN [25] (i.e. single blob with neighbors) image bag generator.[2] In detail, each image is smoothed by a Gaussian filter and subsampled to an 8×8 matrix of color blobs where each blob is a 2×2 set of pixels within the 8×8 matrix. An SBN is defined as the combination of a single blob with its four neighboring blobs (up, down, left, right). The sub-image is described as a 15-dimensional vector, where the first three attributes represent the mean R, G, B values of the central blob and the remaining 12 attributes correspond to the differences in mean color values between the central blob and other four neighboring blobs respectively. Therefore, each image bag is represented by a collection of nine 15-dimensional feature vectors obtained by using each of the nine blobs not along the border as the central blob.

A *potential training* set of 400 images is created by randomly choosing 80 images from each of the five image types. The remaining images constitute a *test set* consisting of 1,600 images, 320 from each of the five image types. Separating the potential training set from the test set is to ensure that results of using various multi-instance learning algorithms could be compared fairly. In this paper, each image type corresponds to a concept class to be learned. For each image type, an *initial training* set is created by randomly picking several positive examples of the target concept and several negative examples, all from the potential training set. SBN is used to generate image bags and the concept is learned by some specific multi-instance learning algorithm. After the concept has been learned, the 1600 images in the test set are sorted based on the learned concept. Two different training schemes are used: *3p3n* that picks three positive examples and three negative examples to form the initial training set; and *5p5n* that picks five positive examples and five negative examples to form the initial training set. Figure 8 shows a sample run of RBF-MIP (combined with SBN) using training scheme *5p5n* where the target concept is *mountains*.

In this paper, the retrieval performance of RBF-MIP is compared with those of BP-MIP [37], Diverse Density [24] and EM-DD [33].[3] For the problem of CBIR, the training set is too small (six examples for *3p3n* and 10 examples for *5p5n*) to effectively estimate the parameter configuration of RBF-MIP as in Sections 5.1

[2]Note that the purpose of the experiments is to compare the performance of different multi-instance learning algorithms instead of testing the effectiveness of different image bag generators, thus only one image bag generator is used for all comparing algorithms.

[3]For RBF-MIP and BP-MIP, the test images are sorted based on the maximum output of any of the image's instances on the trained neural networks; While for Diverse Density and EM-DD, the test images are sorted based on the minimum distance of any of the image's instances from the learned concept point.

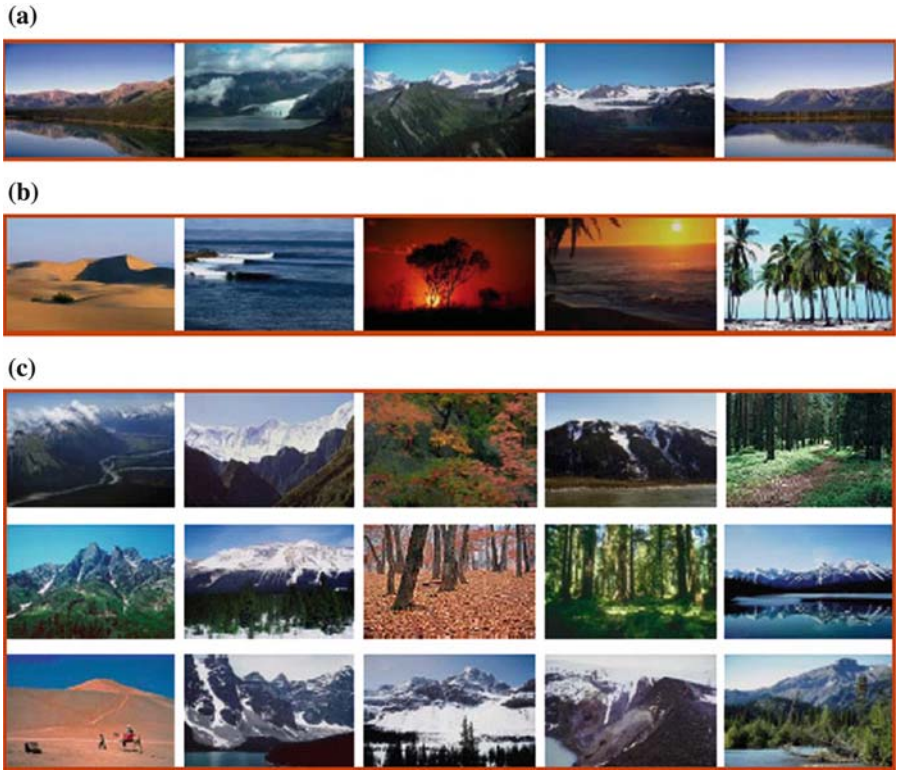

Figure 8. A sample run of RBF-MIP (combined with SBN) for retrieving *mountains* using training scheme *5p5n*. (a) User-selected positive examples; (b) User-selected negative examples; (c) Final retrieval from test set (top 15 images).

and 5.2. Thus, in the following experiments, the number of remaining clusters M in the first layer of RBF-MIP is simply set to be the same number of training examples, i.e. by taking each training example as a cluster of its own. The scaling factor μ is fixed to be 0.3. For BP-MIP, the number of hidden neurons is set to be 15, which equals the dimensionality of each instance in the bags. For Diverse Density and EM-DD, the default parameters are adopted.

On way to evaluate image retrieval performance is to measure the *precision* and *recall*. Precision is the ratio of the number of correctly retrieved images to the number of all images retrieved so far. Recall is the ratio of the number of correctly retrieved images to the total number of correct images in the test set. Given a specific training scheme, for each image type, 10 runs of experiments are performed for each of the four multi-instance learning algorithms. Figure 9 gives the precision-recall curves for training scheme *3p3n*, where precision is plotted against recall as the number of retrieved images increases. The curve of each algorithm is the averaged results of five different image types each with 10 runs of experiments. The higher the precision-recall curve, the better the performance. Similarly, Figure 10 gives the precision-recall curves for training scheme *5p5n*.

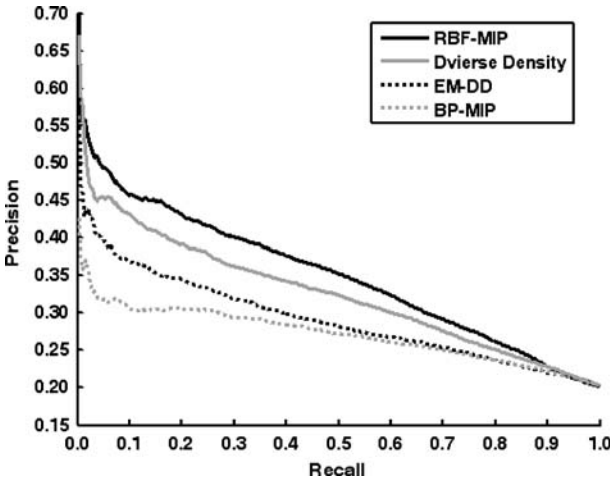

Figure 9. Precision-recall curves for training scheme *3p3n*, where the curve of each algorithm is the averaged results of five different image types each with 10 runs of experiments.

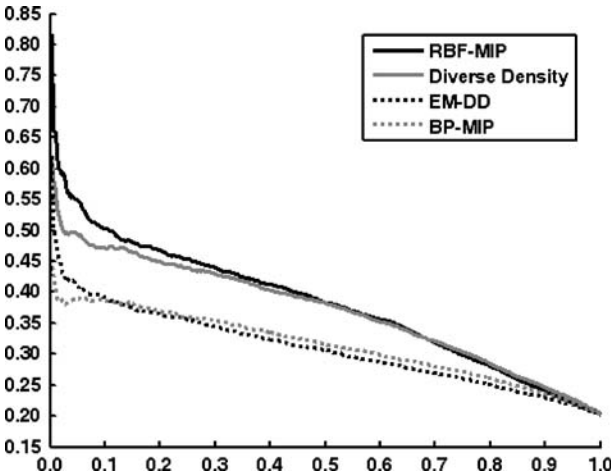

Figure 10. Precision-recall curves for training scheme *5p5n*, where the curve of each algorithm is the averaged results of five different image types each with 10 runs of experiments.

For training scheme *3p3n* (as shown in Figure 9), the performance of RBF-MIP is better than those of Diverse Density, EM-DD and BP-MIP. For training scheme *5p5n* (as shown in Figure 10), RBF-MIP is comparable to Diverse Density and both of them outperform EM-DD and BP-MIP. For image database retrieval, in most cases, the users may only be interested in the top portion of the images returned by the system which correspond to the beginning of the precision-recall curve. Therefore, Tables 7 and 8 report the precision and recall of each comparing algorithm on the top 200 retrieved images with training scheme *3p3n* respectively, where the number following "±" is the corresponding

standard deviation. Similarly, Tables 9 and 10 report the precision and recall of each comparing algorithm on the top 200 retrieved images with training scheme *5p5n* respectively.

Tables 7 and 8 show that, in terms of both precision and recall, RBF-MIP outperforms EM-DD on all image types, outperforms BP-MIP on all image types except *sea*, and outperforms Diverse Density on *desert, sunset* and *trees*, but is inferior to Diverse Density on *mountains* and *sea*. On the average (as shown in the last line of Tables 7 and 8), RBF-MIP performs slightly better than Diverse Density and both of them significantly outperform EM-DD and BP-MIP. Tables 9 and 10 show that, in terms of both precision and recall, RBF-MIP outperforms EM-DD on all image types, outperforms BP-MIP on *desert, mountains* and *trees*, but

Table 7. Precision of each comparing algorithm on the top 200 retrieved images with training scheme *3p3n*.

Image type	RBF-MIP	Diverse density	EM-DD	BP-MIP
Desert	0.362 ± 0.130	0.333 ± 0.145	0.269 ± 0.099	0.265 ± 0.094
Mountains	0.381 ± 0.056	0.393 ± 0.090	0.338 ± 0.078	0.308 ± 0.118
Sea	0.249 ± 0.066	0.285 ± 0.046	0.236 ± 0.051	0.270 ± 0.088
Sunset	0.511 ± 0.083	0.445 ± 0.104	0.426 ± 0.098	0.433 ± 0.145
Trees	0.469 ± 0.113	0.379 ± 0.124	0.366 ± 0.066	0.116 ± 0.087
Average	0.394 ± 0.090	0.367 ± 0.102	0.327 ± 0.079	0.278 ± 0.107

Table 8. Recall of each comparing algorithm on the top 200 retrieved images with training scheme *3p3n*.

Image type	RBF-MIP	Diverse density	EM-DD	BP-MIP
Desert	0.226 ± 0.081	0.208 ± 0.091	0.168 ± 0.062	0.165 ± 0.059
Mountains	0.238 ± 0.035	0.245 ± 0.056	0.211 ± 0.049	0.192 ± 0.074
Sea	0.156 ± 0.041	0.178 ± 0.029	0.147 ± 0.032	0.168 ± 0.055
Sunset	0.319 ± 0.052	0.278 ± 0.065	0.266 ± 0.061	0.271 ± 0.090
Trees	0.293 ± 0.071	0.237 ± 0.077	0.229 ± 0.041	0.073 ± 0.055
Average	0.246 ± 0.056	0.229 ± 0.064	0.204 ± 0.049	0.174 ± 0.067

Table 9. Precision of each comparing algorithm on the top 200 retrieved images with training scheme *5p5n*.

Image type	RBF-MIP	Diverse density	EM-DD	BP-MIP
Desert	0.391 ± 0.092	0.320 ± 0.157	0.290 ± 0.079	0.245 ± 0.126
Mountains	0.427 ± 0.057	0.389 ± 0.125	0.315 ± 0.063	0.408 ± 0.085
Sea	0.316 ± 0.041	0.309 ± 0.047	0.282 ± 0.044	0.349 ± 0.075
Sunset	0.506 ± 0.137	0.642 ± 0.109	0.444 ± 0.118	0.534 ± 0.188
Trees	0.521 ± 0.109	0.379 ± 0.151	0.402 ± 0.124	0.116 ± 0.066
Average	0.432 ± 0.087	0.407 ± 0.118	0.346 ± 0.086	0.330 ± 0.108

Table 10. Recall of each comparing algorithm on the top 200 retrieved images with training scheme *5p5n*.

Image type	RBF-MIP	Diverse density	EM-DD	BP-MIP
Desert	0.244 ± 0.058	0.200 ± 0.098	0.181 ± 0.049	0.153 ± 0.079
Mountains	0.267 ± 0.036	0.243 ± 0.078	0.197 ± 0.039	0.255 ± 0.053
Sea	0.197 ± 0.025	0.193 ± 0.030	0.176 ± 0.028	0.218 ± 0.047
Sunset	0.316 ± 0.086	0.401 ± 0.068	0.277 ± 0.074	0.334 ± 0.117
Trees	0.325 ± 0.068	0.237 ± 0.095	0.251 ± 0.077	0.072 ± 0.041
Average	0.270 ± 0.055	0.255 ± 0.074	0.216 ± 0.054	0.206 ± 0.067

is inferior to BP-MIP on *sea* and *sunset*, and outperforms Diverse Density on all image types except *sunset*. On the average (as shown in the last line of Tables 9 and 10), RBF-MIP performs slightly better than Diverse Density and both of them significantly outperform EM-DD and BP-MIP. The above results indicate that, besides multi-instance classification and multi-instance regression, RBF-MIP could also work well with application to CBIR.

6. Conclusion and Future Work

In this paper, a multi-instance neural network algorithm named RBF-MIP is proposed. It is derived from the traditional RBF method through employing a particular two-stage training procedure, where its first layer is composed of clusters of bags formed by merging training bags agglomeratively and its second layer weights are optimized by minimizing a sum-of-squares function and worked out through SVD. Experiments on the *Musk* data sets, several artificial data sets and natural scene image database retrieval show that RBF-MIP is among the top-ranked learning algorithms on multi-instance problems.

In the trained RBF-MIP neural networks, there exist some clusters containing only one bag when many clusters are remained in the first layer. Investigating some appropriate methods to eliminate these "trivial" clusters is an interesting issue for future work.

Furthermore, recent research has shown that neural network ensemble could significantly improve the generalization ability of neural network based learning systems, which has become a hot topic in both machine learning and neural network communities [36]. Besides, Zhou and Zhang [38] have proposed to build ensembles of several multi-instance learners to solve multi-instance problems, and shown that the investigated multi-instance learners can be enhanced by utilizing ensemble learning paradigms. So, it is interesting to see if better results could be obtained with ensembles of RBF-MIP neural networks.

Acknowledgements

The comments and suggestions from the anonymous reviewers greatly improved this paper. This work was supported by the National Science Foundation of

China under the Grant No. 60473046, the Foundation for the author of National Excellent Doctoral Dissertation of China under the Grant No. 200343, and the National 973 Fundamental Research Program of China under the Grant No. 2002CB312002.

References

1. Alphonse, E. and Matwin, S.: Filtering multi-instance problems to reduce dimensionality in relational learning, *Journal of Intelligent Information Systems*, **22**(1) (2004), 23–40.
2. Amar, R. A., Dooly, D. R., Goldman, S. A. and Zhang, Q.: Multiple-instance learning of real-valued data, In: *Proceedings of the 18th International Conference on Machine Learning*, pp. 3–10, Williamstown, MA, 2001. [http://www.cs.wustl.edu/~sg/multi-inst-data]
3. Andrews, S., Tsochantaridis, I. and Hofmann, T.: Support vector machines for multiple-instance learning, In: Becker, S., Thrun, S. and Obermayer, K. (eds), *Advances in Neural Information Processing Systems 15*, pp. 561–568, MIT Press, Cambridge, MA, 2003.
4. Auer, P.: On learning from multi-instance examples: empirical evaluation of a theoretical approach, In: *Proceedings of the 14th International Conference on Machine Learning*, pp. 21–29, Nashville, TN, 1997.
5. Auer, P., Long, P. M. and Srinivasan, A.: Approximating hyper-rectangles: learning and pseudo-random sets, *Journal of Computer and System Sciences*, **57**(3) (1998), 376–388.
6. Bishop, C. M.: *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.
7. Blake, C., Keogh, E. and Merz, C. J.: *UCI repository of machine learning databases*. Department of Information and Computer Science, University of California, Irvine, CA, 1998. [http://www.ics.uci.edu/~mlearn/MLRepository.html]
8. Blum, A. and Kalai, A.: A note on learning from multiple-instance examples, *Machine Learning*, **30**(1) (1998), 23–29.
9. Chevaleyre, Y. and Zucker, J.-D.: Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem, In: Stroulia, E. and Matwin, S. (eds), *Lecture Notes in Artificial Intelligence 2056*, pp. 204–214, Springer, Berlin, 2001.
10. Dempster, A. P., Laird, N. M. and Rubin, D. B.: Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistics Society, Series B*, **39**(1) (1977), 1–38.
11. De Raedt, L.: Attribute-value learning versus inductive logic programming: the missing links, In: Page, D. (ed.), *Lecture Notes in Artificial Intelligence 1446*, pp. 1–8, Springer, Berlin, 1998.
12. Dietterich, T. G., Lathrop, R. H. and Lozano-Pérez, T.: Solving the multiple-instance problem with axis-parallel rectangles, *Artificial Intelligence*, **89**(1–2) (1997), 31–71.
13. Dooly, D. R., Goldman, S. A. and Kwek, S. S.: Real-valued multiple-instance learning with queries, In: Abe, N., Khardon, R. and Zeugmann, T. (eds), *Lecture Notes in Artificial Intelligence 2225*, pp. 167–180, Springer, Berlin, 2001.
14. Edgar, G. A.: *Measure, Topology, and Fractal Geometry*, 3rd print, Springer-Verlag, Berlin, 1995.
15. Gärtner, T., Flach, P. A., Kowalczyk, A. and Smola, A. J.: Multi-instance kernels, In: *Proceedings of the 19th International Conference on Machine Learning*, pp. 179–186, Sydney, Australia, 2002.
16. Goldman, S. A., Kwek, S. S. and Scott, S. D.: Agnostic learning of geometric patterns, *Journal of Computer and System Sciences*, **62**(1) (2001), 123–151.

17. Goldman, S. A. and Scott, S. D.: Multiple-instance learning of real-valued geometric patterns, *Annals of Mathematics and Artificial Intelligence*, **39**(3) (2003), 259–290.
18. Jollife, I. T.: *Principle Component Analysis*, Springer-Verlag, New York, 1986.
19. Kearns, M. J.: Efficient noise-tolerant learning from statistical queries, In: *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pp. 392–401, San Diego, CA, 1993.
20. Kearns, M. J. and Schapire, R. E.: Efficient distribution-free learning of probabilistic concepts, *Journal of Computer and System Sciences*, **48**(3) (1994), 464–497.
21. Lindsay, R., Buchanan, B., Feigenbaum, E. and Lederberg, J.: *Applications of Artificial Intelligence to Organic Chemistry: The DENDRAL Project*, McGraw-Hill, New York, 1980.
22. Long, P. M. and Tan, L.: PAC learning axis-aligned rectangles with respect to product distribution from multiple-instance examples, *Machine Learning*, **30**(1) (1998), 7–21.
23. Maron, O.: Learning from Ambiguity, PhD dissertation, Department of Electronical Engineering and Computer Science, MIT, Cambridge, MA, Jun. 1998.
24. Maron, O. and Lozano-Pérez, T.: A framework for multiple-instance learning, In: Jordan, M. I., Kearns, M. J. and Solla, S. A. (eds), *Advances in Neural Information Processing Systems 10*, pp. 570–576, MIT Press, Cambridge, MA, 1998.
25. Maron, O. and Ratan, A. L.: Multiple-instance learning for natural scene classification, In: *Proceedings of the 15th International Conference on Machine Learning*, pp. 341–349, Madison, WI, 1998.
26. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P.: *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edn, Cambridge University Press, New York, 1992.
27. Ray, S. and Page, D.: Multiple instance regression, In: *Proceedings of the 18th International Conference on Machine Learning*, pp. 425–432, Williamstown, MA, 2001.
28. Ruffo, G.: Learning single and multiple decision trees for security applications, PhD dissertation, Department of Computer Science, University of Turin, Italy, 2000.
29. Rumelhart, D. E., Hinton, G. E. and Williams, R. J.: Learning internal representations by error propagation, In: Rumelhart, D. E. and McClelland, J. L. (eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, pp. 318–362, MIT Press, Cambridge, MA, 1986.
30. Sebag, M. and Rouveirol, C.: Tractable induction and classification in first order logic, In: *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pp. 888–893, Nagoya, Japan, 1997.
31. Wang, J. and Zucker, J.-D.: Solving the multiple-instance problem: a lazy learning approach, In: *Proceedings of the 17th International Conference on Machine Learning*, pp. 1119–1125, San Francisco, CA, 2000.
32. Yang, C. and Lozano- Pérez, T.: Image database retrieval with multiple-instance learning techniques, In: *Proceedings of the 16th International Conference on Data Engineering*, pp. 233–243, San Diego, CA, 2000.
33. Zhang, Q. and Goldman, S. A.: EM-DD: an improved multiple-instance learning technique, In: Dietterich, T. G., Becker, S. and Ghahramani, Z. (eds), *Advances in Neural Information Processing Systems 14*, pp. 1073–1080, MIT Press, Cambridge, MA, 2002.
34. Zhang, Q., Yu, W., Goldman, S. A. and Fritts, J. E.: Content-based image retrieval using multiple-instance learning, In: *Proceedings of the 19th International Conference on Machine Learning*, pp. 682–689, Sydney, Australia, 2002.
35. Zhang, M.-L. and Zhou, Z.-H.: Improve multi-instance neural network through feature selection, *Neural Processing Letters*, **19**(1) (2004), 1–10.
36. Zhou, Z.-H., Wu, J. and Tang, W.: Ensembling neural networks: many could be better than all, *Artificial Intelligence*, **137**(1–2) (2002), 239–263.

37. Zhou, Z.-H. and Zhang, M.-L.: Neural networks for multi-instance learning, Technical Report, AI Lab, Computer Science & Technology Department, Nanjing University, China, Aug. 2002.
38. Zhou, Z.-H. and Zhang, M.-L.: Ensembles of multi-instance learners, In: Lavrač, N., Gamberger, D., Blockeel, H. and Todorovski, L. (eds), *Lecture Notes in Artificial Intelligence 2837*, pp. 492–502, Springer-Verlag, Berlin, 2003.
39. Zucker, J.-D. and Ganascia, J.-G.: Changes of representation for efficient learning in structural domains, In: *Proceedings of the 13th International Conference on Machine Learning*, pp. 543–551, Bary, Italy, 1996.
40. Zucker, J.-D. and Ganascia, J.-G.: Learning structurally indeterminate clauses, In: Page, D. (ed.), *Lecture Notes in Artificial Intelligence 1446*, pp. 235–244, Springer, Berlin, 1998.