Multiple-Instance Learning Via Random Walk

Dong Wang, Jianmin Li, and Bo Zhang

State Key Laboratory of Intelligent Technology and System,
Department of Computer Science and Technology,
Tsinghua University, Beijing, 100084, P.R. China
wdong01@mails.tsinghua.edu.cn, {lijianmin, dcszb}@mail.tsinghua.edu.cn

Abstract. This paper presents a decoupled two stage solution to the multiple-instance learning (MIL) problem. With a constructed affinity matrix to reflect the instance relations, a modified Random Walk on a Graph process is applied to infer the positive instances in each positive bag. This process has both a closed form solution and an efficient iterative one. Combined with the Support Vector Machine (SVM) classifier, this algorithm decouples the inferring and training stages and converts MIL into a supervised learning problem. Compared with previous algorithms on several benchmark data sets, the proposed algorithm is quite competitive in both computational efficiency and classification accuracy.

1 Introduction

Multiple-instance learning (MIL) is a generalization of supervised learning. Unlike supervised learning, this model assumes that instances are contained in bags and the instance labels are hidden. The bag label is related to the hidden labels of the instances as follows: the bag is labeled as positive if any single instance in it is positive, otherwise it is labeled negative. Given a training set of labeled bags, an MIL algorithm attempts to find a hypothesis that correctly explains the labels for the bags on the training set and generalizes well to predict the labels for unseen bags. Many real world applications can be formulated in the MIL setting, e.g. drug design, protein identification, hard drive failure prediction, stock prediction, text categorization, content-based image retrieval (CBIR) and more recently content-based video retrieval (CBVR).

After Dietterich *et. al* first formulate MIL in [1], substantial research has been carried out in this area in the last few years, e.g. [2,3,4,5,6]. They can be roughly divided into two categories. A few algorithms operate directly on the bag level while others try to infer the hidden instance labels and then resort to supervised learning techniques. Intuitively, an instance is likely to be positive if it relates to many positive bags and does not relate to any negative bags. However the strict intersection of the positive bags may be empty when features are real valued. Therefore, a notable concept of "Diverse Density" (DD) is defined to measure in the feature space the co-occurrence of instances from different positive bags [2]. Unfortunately, it is computationally intensive to exhaustively search the feature space for the point which maximizes DD [2,3] and overfitting may occur since

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): ECML 2006, LNAI 4212, pp. 473–484, 2006.

no regularization term is present. Searching for multiple local maximum points called "instance prototypes" [5] is also possible but it suffers from the same computation problem.

The motivation of the present study is to efficiently infer the underlying positive instances for all positive bags in parallel and let a classifier with proper regularization term do the left work. There are two advantages by doing so: 1) computational efficiency comes from operating only on the instances instead of searching in the whole space; 2) the two stages of inferring and training are decoupled and thus the training stage is open to different classification schemes. Holding this in mind, we take a Random Walk on a Graph approach to infer the underlying positive instances. Sending these inferred positive instances and the ones in negative bags into the state-of-the-art Support Vector Machine (SVM) classifier, we further takes advantage of the generalizing capability of SVM. Tested on several standard benchmark data sets, this approach is quite competitive in both computational efficiency and classification accuracy with previous ones.

The paper is organized as follows: Section 2 briefly presents related work. Section 3 introduces the Random Walk on a Graph and adapts it to the MIL setting. Section 4 gives the implementation details and Section 5 presents the experimental results. Section 6 discusses the relation of the present study and the most related work and Section 7 concludes this paper.

2 Related Work

2.1 Multiple Instance Learning

The MIL problem is first presented in [1], and an algorithm is also proposed with hypothesis classes consisting of Axis Parallel Rectangles (APR). Two methods focused on the DD concept are developed [2,3]. The former (DD algorithm) takes a two step scaling and gradient search approach, and the latter (EMDD) treats the hidden information of the underlying positive instances as a missing value and uses Expectation-Maximum (EM) to estimate it. Both include computationally dense operations. A few research tries to tailor the supervised learning algorithms for the MIL setting, such as [7,8,9]. Recently, an interesting comparison is made between supervised and multiple instance learning methods [6].

Another line of research handles the bag directly. A kernel-based approach is suggested which derives MI-kernels on bags from statistics of instances [10]. [4] propose both the maximum pattern and the maximum bag margin formulation (mi/MI-SVM) of MIL and solve the derived mixed-integer programming problem in heuristic manner due to formidable computation otherwise. DD-SVM is developed to search for multiple local maximum points and define similar bag kernels in an SVM framework [5]. Approximate Box Counting (ABC) [11] is proposed for extended "*r-of-k*" MIL setting. Ensemble learning methods of Boosting [12] and Ensemble-EMDD [13] are also explored.

2.2 Random Walk on a Graph

Random Walk on a Graph has been initially proposed to compute the absolute relevance of pages (vertices) in a hyperlinked environment, like the web, in the form of the PageRank algorithm [14]. A slightly different algorithm is developed for manifold learning [15]. This manifold learning approach has attracted more research attention in the learning community recently. For example, sparsely or densely connected graphs are built to deal with semi-supervised learning problems [16,17,18,19,20] and to cluster data [21,22]. However, though connected by propagating the labels on manifold data structure, MIL is different from semi-supervised learning in that the latter deals with both labeled and unlabeled data while the former concerns itself with the hierarchical label ambiguity and tries to identify the underlying positive instances in positive bags.

3 Random Walk on a Graph for MIL

In this section, after introducing some assumptions and notations, we first adapt the Random Walk on a Graph algorithm to infer the underlying positive instances, then combine an SVM classifier to solve the converted supervised learning problem. We also show that the adapted Random Walk on a Graph algorithm is somewhat similar in spirit to the original DD concept.

3.1 Assumptions and Notations

We make two assumptions here. The first one is that the positive instances form certain clusters in the feature space and the negative ones are distributed in the remaining space. The second one is that the bags and the instances are drawn independently from the actual data. These are two general assumptions which are usually satisfied in real world scenarios.

Let B_i denote the i^{th} bag, B_i^+ a positive bag and B_i^- a negative one. Let $\mathcal{B} = \{B_i\}$, $\mathcal{B}^+ = \{B_i^+\}$ and $\mathcal{B}^- = \{B_i^-\}$. Let J denote the set of all instances and $n = |J|$. An instance $I_j, j \in J$ is denoted I_j^+ when it is positive and is denoted I_j^- when negative. I_j can also be denoted as B_{ij} to emphasis that $I_j \in B_i$ and as B_{ij}^+ if it is positive. Note that j is a global index for instances and does not relate to specific bag index. Let NN_j denote the set of k nearest neighbor (k-NN) instances from *other* bags for each instance I_j. Denote by $p(I_j^+)$ the probability of I_j being positive and $p(I_j^-)$ similarly.

Each instance I_j is represented by a d-dimensional feature vector. Let $A_{n\times n}$ be the affinity matrix defined for the instances. Normalize A by $S = D^{-1}A$, where D is the diagonal matrix with $D_{ii} = \sum_{k=1}^{n} A_{ik}$.

3.2 Random Walk on a Graph

A Random Walk on a given graph $G = \{V, E\}$, where V is the vertex set of size n and E the edge set, describes how a random walker jumps among vertices following the edges with certain probabilities. This can be characterized

by a discrete time markov chain which allows us to compute the probability x_p of being located in each vertex p at time t. Suppose that the transition probability matrix is P and the probability distribution over all the vertices is $x(t) = [x_1(t), \ldots, x_n(t)]^T$, a unique stationary distribution x^* is readily derived since P is a stochastic matrix having its maximum eigenvalue equal to one and this guarantees the convergence (see e.g. [23], chapter 4).

Initialize $x(0) = r$ at $t = 0$ so that r is a probabilistic distribution for the random walker to start with. A restart coefficient α is introduced to indicate the probability αr that the walker stops following edges and restarts from the vertices again. Suppose that a random walker starts at vertex u, it follows the arc (u, v) to vertex v with probability $(1-\alpha)p_{vu}$, where $p_{vu} = P(v, u)$ is the transition probability from vertex u to v, or restarts from v with probability αr_v. The probability $x_v(t)$ are updated at each time step by the following equation

$$x_v(t+1) = (1-\alpha)\sum_{u \in V} p_{vu} x_u(t) + \alpha r_v. \tag{1}$$

The compact matrix form is that

$$x(t+1) = (1-\alpha)Px(t) + \alpha r. \tag{2}$$

Inserting $y = x - \alpha(I - (1-\alpha)P)^{-1}r$ into (2), we have that $y(t+1) = (1-\alpha)Py(t)$. The convergence to the stationary distribution x^* comes directly since $y^* = 0$. We can easily show that

$$x^* = \alpha(I - (1-\alpha)P)^{-1}r. \tag{3}$$

This completes the closed form solution.

Although x^* can be expressed in a closed form, the iterative solution provides a more efficient algorithm for large scale problems. It just uses (2) to iterative update x until convergence arrived. The exponential convergence of x^* is easily derived from the Dobrushin convergence theorem (see also [23]).

3.3 Adapting Random Walk to the MIL Setting

The original DD concept does capture the nature of the MIL problem. It tries to find the point c $\arg\max_c p(\mathcal{B}|c) = p(\mathcal{B}^+|c)p(\mathcal{B}^-|c)$. Assuming both bag and instance independence and under a noisy-or model[2], it turns out to be $p(\mathcal{B}|c) = \prod_i \big(1 - \prod_j (1 - p(c|B_{ij}^+))\big)\prod_l \prod_k \big(1 - p(c|B_{lk}^-)\big)$ which measures how *different* positive bags have instances near c and how far negative instances are from c [2]. One well placed negative instance will bring DD to near zero since DD is very sensitive to instances in negative bags. However, the computation of searching the whole feature space for c is formidable.

Adopting an additive model instead of the noisy-or model, we attempt to estimate a likelihood ratio (LR) of $p(\mathcal{B}^+|c)/p(\mathcal{B}^-|c)$ instead of the likelihood for each instance in parallel, and treat the inferred instances in a supervised learning framework. This additive model assumes that each instance casts its probabilistic

vote to every instance independently and each instance receives votes additively. So it is more robust to instances in negative bags than the noisy-or model due to its additive nature. We substitute c with I_j so that $p(\mathcal{B}^+|c) = p(\mathcal{B}^+|I_j^+)$ and $p(\mathcal{B}^-|c) = p(\mathcal{B}^-|I_j^-)$ since we are only interested in the instances, not the whole feature space. To be specific, we model that

$$p(\mathcal{B}^+|I_j^+) \propto p(I_j^+|\mathcal{B}^+) = \sum_i p(I_j^+|B_i^+) = \sum_i \sum_k p(I_j^+|B_{ik}^+)\hat{p}(B_{ik}^+|B_i^+), \quad (4)$$

where $\hat{p}(B_{ik}^+|B_i^+)$ is the local probability for I_k being positive given its bag label as positive. The interpretation of (4) is that the collective instance votes for instance I_j sum up to the probability of I_j being positive given $\mathcal{B}^+$. We model that

$$p(\mathcal{B}^-|I_j^-) \propto p(I_j^-|\mathcal{B}^-) = \sum_l p(I_j^-|B_l^-) = \sum_l \sum_k p(I_j^-|B_{lk}^-)\hat{p}(B_{lk}^-|B_l^-), \quad (5)$$

similarly. This results in our LR measure of $p(\mathcal{B}^+|I_j^+)/p(\mathcal{B}^-|I_j^-)$ since $p(\mathcal{B}^+|I_j^+)$ and $p(\mathcal{B}^-|I_j^-)$ are modeled in the same way. In contrast, the likelihood measure is adopted since the noisy-or model treat these two parts asymmetrically.

We can grasp the solution (3) from this probabilistic voting view. Let $\beta = 1 - \alpha$, and omit the constant coefficient α in (3) for the time being, we have

$$\begin{aligned} x^* &= (I - \beta P)^{-1} r \\ &= (I + \beta P + \beta^2 P^2 + \ldots) r \\ &= r + \beta P r + \beta P(\beta P r) + \ldots . \end{aligned} \quad (6)$$

Compare (4) and (6) by plugging in $r_j = \hat{p}(I_j^+|\mathcal{B}^+)$, $x_j^* = p(I_j^+|\mathcal{B}^+)$ for each instance I_j and remember that $B_{ik} = I_k$, we have that

$$p(I_j^+|B_{ik}^+) = \delta_{jk} + \beta p_{jk} + \sum_l \beta^2 p_{jl} p_{lk} + \ldots \quad (7)$$

where δ_{jk} is the indication function. $p(I_j^+|B_{ik}^+)$ is the probability of commuting from I_k to I_j in infinite time with a damping factor β punishing long commuting time. This choice of $p(I_j^+|B_{ik}^+)$ is in accordance with our intuition that closeness makes similar. The second assumption stated in Section 3.1 guarantees that the closeness in the feature space is meaningful for inferring instances. Estimating $p(I_k^-|\mathcal{B}^-)$ follows an identical procedure with similar input r_k as $\hat{p}(B_{lk}^-|B_l^-)$.

The LR calculation is thus completed within the Random Walk framework. However, two issues of how to construct P and choose r remain to be solved. Firstly in the graph construction, the normalized affinity matrix $S = D^{-1}A$ is taken as the transition probability matrix P. The affinity matrix A can typically be defined by a Gaussian weighted k-NN distances in (8). The intra-bag k-NN relations are intentionally exclude by only allowing k-NN from *other* bags in A. So the random walker is forced to move among different bags in every step to

avoid *self-reinforcement*. Otherwise two nearby negative instances in the same positive bag will vote heavily on each other and cause false high scores in x^*.

Secondly, the input vector r^+ for estimating $p(I_j^+|\mathcal{B}^+)$ is set so that $r_j = 1$ if $I_j \in B_i^+$ and $r_j = 0$ otherwise. This means that $\hat{p}(B_{ik}^+|B_i^+) = 1$ if $I_k \in B_i^+$ and $\hat{p}(B_{ik}^+|B_i^+) = 0$ if $I_k \in B_l^-$. So according to (4), the instances in negative bags will not contribute to $p(I_j^+|\mathcal{B}^+)$. Similarly, the input vector r^- for estimating $p(I_j^-|\mathcal{B}^-)$ is set so that $r_j = 1$ if $I_j \in B_l^-$ and $r_j = 0$ otherwise. However, noticing that $P\mathbf{1} = \mathbf{1}$, where $\mathbf{1} = [1, \ldots, 1]^T$ and $r^+ + r^- = \mathbf{1}$, it follows that $p(I_j^+|\mathcal{B}^+) + p(I_j^-|\mathcal{B}^-) = 1$. So the LR for I_j is further simplified to $\frac{x_j^*}{1-x_j^*}$ and instances in each positive bag can be ranked as being positive according to this LR measure.

After adapting the Random Walk on a Graph process to the MIL setting, a further instance selection and classifier training stage is added. SVM is chosen for its generalization ability. We omit an introduction to SVM and refer the readers to [24]. However the Random Walk, by itself, is neutral to the choice of classifiers.

4 Algorithm Description and Details

The completed running algorithm, Random Walk with SVM (RW-SVM) is shown in Figure 1. The algorithm implementation and parameter settings are given as follows.

4.1 Implementation Details

Given the bags and the instance features, the first step to construct the affinity matrix A is to calculate the k nearest neighbors (k-NN) from *other* bags for each instance and set the edge with the distance defined in (8). There are several other methods to generate A. For example, A can be dense if all inter-bag distances are incorporated. However the computational cost will increase dramatically. Another

Algorithm: Random Walk with SVM

Input: n instances with the corresponding feature vectors, bag indexes and bag labels.
Output: SVM classifier C.

1. Construct affinity matrix A by setting $a_{jk} = a_{kj} = d(j,k)$, where $k \in NN_j$, k-NN from *other* bags for I_j, and $d(j,k)$ is the distance function in (8) or (9).
2. Normalize A so that $S = D^{-1}A$ in which D is diagonal with $d_{ii} = \sum_{k=1}^{n} a_{ik}$.
3. Set input vector r as $r_j = 1$ if $I_j \in B_i^+$ and $r_j = 0$ otherwise.
4. Let $P = S$, $x(0) = r$, and iteratively compute $x(t)$ with (2) until it converges to x^*.
5. Select I_j for each B_i^+ that $\arg\max_j \frac{x_j^*}{1-x_j^*}, I_j \in B_i^+$ and select all instances $I_j \in B_l^-$.
6. Train the SVM classifier C using the selected instances.

Fig. 1. Algorithm description of Random Walk with SVM

possibility is that the relative importance of each feature dimension may be different regarding the distance calculation. However, the scaling parameter σ_l for every feature dimension l, which is iteratively optimized in [2,3], is set to a constant σ across feature dimensions in (8) since the equal importance of feature dimensions works well in [4].

Notice that in this way, the constructed graph is not necessarily connected and may consist of several separate clusters. Some instances will still have zero scores after the iteration process. However, those are of course negative ones since they are not connected with the instances in positive bags.

In step 2, S can also be symmetrically normalized as $S = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ at the cost of losing the probabilistic interpretation for the Random Walk process. So the current form is preserved.

After S and r are set, they are used to compute the stationary distribution x^* until convergence. Then the instance I_j that maximizes $x^*_j, I_j \in B_i^+$ is chosen as the positive instance for each B_i^+; all instances are chosen as negative instances for each B_i^-. This is a conservative scheme for choosing positive instances since each positive bag contains at least one positive instance. More complicated schemes for choosing the positive instances are also possible, e.g. set a threshold for positive instances across positive bags, or use mi/MI-SVM like iterative scheme.

These selected instances are fed into the SVM classifier. LIBSVM [24] is used as our SVM implementation.

During test phase, the instances are classified by C and the bag is decided to be positive when any of the instances in it is classified as positive.

4.2 Parameter Settings

There are three parameters for graph construction, namely, the distance function, σ and k. The L_2 distance with Gaussian kernel is chosen for defining edge weight in affinity matrix A as follows:

$$a_{pq} = d_G(p,q) = \frac{1}{Z}\prod_{l=1}^{d}\exp\Big(\frac{-(p_l - q_l)^2}{\sigma_l^2}\Big), \tag{8}$$

where p, q are two instances in the feature space and Z the normalization constant. Note that $\sigma_l = \sigma$ for all dimensions. For some features depending on the data set (see 5 for details), the cosine distance with Laplace kernel is adopted as follows:

$$a_{pq} = d_C(p,q) = \frac{1}{Z}\exp\Big(\frac{-(1 - \cos<p,q>)}{\sigma_c}\Big). \tag{9}$$

There is one parameter for the iteration process, the restart probability α. These four parameters are so-called hyper-parameters. There are also different parameters for different kinds of SVMs on each data set.

Generally speaking, it is difficult to tune the hyper-parameters. Due to the limited data available and the need of fair comparison with previous approaches, the parameter tuning method adopted here is somewhat complicated. The final classification results are obtained by 10-fold cross-validation runs on each data set.

Inside each cross-validated run, a nested 3-fold cross-validation is carried out to determine these parameters. The hyper-parameters are chosen with fixed SVM parameters and each hyper-parameter is determined independently with other hyper-parameters fixed. After these three parameters are set, the SVM parameters are further determined on the same 3-fold cross validation data by a 5×5 grid search procedure. The distance function are chosen according to the data set used. The SVM kernel type are chosen as the same with the previous approaches for fairness.

5 Experiments

5.1 Experimental Setup

Following [4], we perform experiments on the same data sets[1] to evaluate the proposed algorithm and compare it with other methods. The data sets are from a variety of application domains, including the most frequently used MUSK, a small portion of Corel and TREC9. Bag classification accuracy is taken as the performance measure for comparison. The results are averaged over 10-fold cross-validation runs. This random cross-validation is again repeated 10 times to get the significance of the results at $p > 0.95$. However we do not have the corresponding confidence interval for the comparative methods on these data sets since this kind of measurement is not provided in previous studies.

The testing data are *excluded* from the Random Walk process in each fold. The performances of previous methods are adopted from [4] unless otherwise stated. The actual running time are also reported on the MUSK data set to show the effectiveness and efficiency of the proposed approach. However for the Corel and TREC9 data sets, comparison are made only among EMDD, mi/MI-SVM and RW-SVM due to the lack of performance data of other approaches mentioned.

To simplify the parameter tuning process, we choose the default value of the numerical hyper-parameters as $\sigma = 10$ for (8), $k = 15$ and $\alpha = 0.1$. These values are those frequently chosen by the inner 3-fold cross-validation processes on the MUSK data set. These default parameters works pretty well for the data sets and are used unless otherwise stated.

5.2 Drug Design

The MUSK data sets, which are used for drug design, are the benchmark used in virtually all previous approaches. Both data sets, MUSK1 and MUSK2, consist of descriptions of molecules (bags) using multiple low-energy conformations (instances). Each conformation is represented by a 166-dimensional feature vector derived from surface properties. The L_2 distance function in (8) is used for constructing A. The RBF kernel $K(x; y) = exp(-\kappa\|x-y\|^2)$ is adopted for both data sets and feature vectors are normalized in a per-dimension min-max style [24] for

[1] For detailed data set descriptions , see [4]. The data sets are available from http://www.cs.brown.edu/people/stu/mil/datasets.html

Table 1. Classification accuracy of different methods on the Musk data sets

Data set	APR	DD	EMDD	MI-NN	mi-SVM	MI-SVM	DD-SVM	ABC	Boosting	RW-SVM
Musk1	**92.4**	88.0	84.8	88.9	87.4	77.9	85.8	91.2	92.0	87.6±1.1
Musk2	89.2	84.0	84.9	82.5	83.6	84.3	**91.3**	90.3	87.1	87.1±0.9

Table 2. Classification accuracy of different methods on the Corel image data sets

Data set	EMDD	mi-SVM			MI-SVM			RW-SVM		
Category		linear	poly	rbf	linear	poly	rbf	linear	poly	rbf
Elephant	78.3	82.2	78.1	80.0	81.4	79.0	73.1	82.1±0.8	**83.7±1.1**	83.3±1.7
Fox	56.1	58.2	55.2	57.9	57.8	59.4	58.8	57.0±1.8	58.5±2.0	**60.0±2.0**
Tiger	72.0	78.4	78.1	78.9	**84.0**	81.6	66.6	78.1±1.0	78.7±1.0	79.5±0.7

SVM input. As summarized in Table 1 with the confidence interval, RW-SVM achieves fairly good performance on both MUSK1 and MUSK2[2] among all these algorithms (See Section 2 for proper abbreviations for the algorithms). We try to add more instances as positive by introducing a global threshold of $th = \max x_j^*$ for all instances in negative bags. All instances in positive bags with $x_j^* > th$ are taken as positive ones. However, no significantly better result is produced.

5.3 Automatic Image Annotation

For the image annotation task from the Corel data, the original color images (bags) have been segmented to regions (instances), each characterized by color, texture and shape descriptors of total 230 dimensions. Three different categories ("elephant", "fox", "tiger") are used. In each case, the data set has 100 positive and 100 negative example images. The latter have been randomly drawn from a pool of photos of other animals. The L_2 distance in (8) is chosen for constructing A. The feature vectors are normalized in a per-dimension min-max style [24] for the three kinds of kernels used. As shown in Table 2, RW-SVM outperforms EMDD by a few percent, and performs comparably with mi/MI-SVM.

5.4 Text Categorization

We also test our algorithm on data sets from text categorization. These data sets are extremely sparse and high-dimensional, which makes them more challenging. In short, the data set which comes from TREC9 is randomly subsampled and split into subsets. Each subset contains a few thousands paragraphs (instances) and at lease 100 documents (bags). The L_2 normalized cosine distance in (9) is chosen for constructing A and $\sigma_c = 0.5$ for (9) and $k = 40$ are set for these data sets. The two parameters are again chosen as the values which are frequently selected

[2] The performances of DD-SVM, ABC and Boosting are adopted from [5,11,12] respectively.

Table 3. Classification accuracy of different methods on the TREC9 document categorization sets

Data set	EMDD	mi-SVM			MI-SVM			RW-SVM		
Category		linear	poly	rbf	linear	poly	rbf	linear	poly	rbf
TST1	85.8	93.6	92.5	90.4	93.9	93.8	93.7	**96.1±0.3**	95.3±0.4	96.0±0.3
TST2	84.0	78.2	75.9	74.3	**84.5**	84.4	76.4	81.4±0.6	78.8±1.8	82.3±0.8
TST3	69.0	87.0	83.3	69.0	82.2	85.1	77.4	**88.9±0.4**	71.7±2.6	88.9±0.4
TST4	80.5	82.8	80.0	69.6	82.4	82.9	77.3	84.7±0.5	74.2±3.5	**84.9±0.6**
TST7	75.4	**81.3**	78.7	**81.3**	78.0	78.7	64.5	79.1±0.8	77.9±2.1	79.9±0.9
TST9	65.5	67.5	65.6	55.2	60.2	63.7	57.0	70.9±0.5	62.1±0.9	**71.4±1.1**
TST10	78.5	79.6	78.3	52.6	79.5	81.0	69.1	83.6±0.5	73.4±2.1	**83.7±0.5**

by the inner 3-fold cross-validation processes for hyper-parameters on TS1 data set. The feature vectors are normalized to unit length with L_2 norm for the three kinds of kernels. As shown in Table 3, RW-SVM shows improved performance over mi/MI-SVM (and EM-DD subsequently) in five out of seven subsets.

5.5 Running Time

As shown in Table 4, the total time spent by our algorithm was 56.5s (2 hours) for Musk 1 (Musk 2) on a P4-3.0 GHz PC. This time includes graph construction and 5×5 grid search of 3 fold cross validation for SVM parameters which is carried out in each of the 10 runs. The Random Walk process itself typically requires less than a few tens of iterations which equal to 0.1 second for several thousand instances. Although the time is not directly comparable across algorithms[3], it does give us some hint of the order of the algorithms' computational complexity.

Table 4. Running time of several algorithms on the MUSK data sets. Note that it only provides some hint of the order of computational complexity of each algorithm.

Data set	RW-SVM	ABC	EMDD
MUSK1	56.5 seconds	2 hours	135 hours
MUSK2	2 hours	40 hours	485 hours

6 Discussion

Our algorithm extends the DD concept in the following aspects: the iterative two stage operations are separated and simplified as two cascade stages of inferring and training; the influence of not-so-near instances is accumulated with discounting factor β; the additive probabilistic model, which is more preferable in some cases

[3] The running time for ABC and EMDD are adopted from [11] on a 750 MHz Sun Blade.

as discussed in [5], replaces the noisy-or model; a regularization term is added in both the Random Walk [25] and the SVM classification process afterwards.

Both [4] and [5] consider SVMs for MIL. The main difference between [4] and our algorithm is that their formulation still results in two interleaved stages of inferring and classifying while ours decouples the two stages. In the presentation of [5], multiple local maximum points for DD are searched in the whole feature space. In the current study, the inferred underlying positive instances instead of the local maximum points are sent to SVM for classification and a dramatic speedup is thus produced.

7 Conclusion and Further Work

In this paper, a decoupled two stage solution is presented for the multiple-instance learning (MIL) problem. With an affinity matrix to reflect the data manifold, a modified Random Walk on a Graph process tries to infer the positive instances. This algorithm has both a guaranteed convergence and a fast iterative implementation. It is also open to different classification schemes. Comparative experiments on some benchmark data sets have shown that this algorithm is quite competitive with previous algorithms in both accuracy and speed when combined with SVMs.

Further work includes designing different schemes to choose the positive and negative instances for classifier training, comparing more thoroughly with other supervised and MI learning methods, applying advanced manifold learning algorithms, fusing different kinds of features and classifiers in this framework and applying this algorithm to large scale real-world tasks which are not handled before due to computational limits, such as the large scale CBVR data set of TRECVID 2006.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (60135010, 60321002), by the Chinese National Key Foundation Research and Development Plan (2004CB318108). Special thanks goes to Andrews Stuart for providing the data sets and insightful comments and Chih-Jen Lin for the LIBSVM implementation. Thanks the anonymous reviewers for invaluable advices.

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. Maron, O.: Learning from Ambiguity. PhD thesis, MIT (1998)
3. Zhang, Q., Goldman, S.: Em-dd: An improved multiple-instance learning technique. In: NIPS (14). (2002) 1073–1080
4. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: NIPS(15). (2003)

5. Chen, Y., Wang, J.Z.: Image categorization by learning and reasoning with regions. Journal of Machine Learning Research **5** (2004) 913–939
6. Ray, S., Craven, M.: Supervised versus multiple instance learning: An empirical comparison. In: Proc. 22th International Conf. on Machine Learning. (2005)
7. Ramon, J., Raedt, L.D.: Multi instance neural networks. In: Proceedings of ICML-2000, Workshop on Attribute-Value and Relational Learning. (2000)
8. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: a lazy learning approach. In: Proc. 17th Int. Conf. on Machine Learning. (2000) 1119–1125
9. Ruffo, G.: Learning single and multiple instance decision trees for computer security applications. PhD thesis, Universitá di Torino, Torino, Italy (2000)
10. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-instance kernels. In: Proc. 19th International Conf. on Machine Learning. (2002) 179–186
11. Tao, Q., Scott, S.D., Vinodchandran, N.V.: Svm-based generalized multiple-instance learning via approximate box counting. In: Proc. 21th International Conf. on Machine Learning. (2004)
12. Auer, P., Ortner, R.: A boosting approach to multiple instance learning. In: Proc. of the 15th European Conf. on Machine Learning. (2004)
13. Rahmani, R., Goldman, S.A., Zhang, H., Krettek, J., Fritts, J.E.: Localized content based image retrieval. In: Proc. ACM Int. Conf. on Multimedia (ACM MM). (2005)
14. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Proc. 7th Int. World Wide Web Conf. (1998)
15. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS(15). (2003) 237–244
16. Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: NIPS. (2002)
17. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML. (2003)
18. Belkin, M., Niyogi, P.: Using manifold stucture for partially labeled classification. In: NIPS. (2003)
19. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: Proc. 22th International Conf. on Machine Learning. (2005)
20. Zhu, X., Lafferty, J.: Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In: Proc. 22th International Conf. on Machine Learning. (2005)
21. Kulis, B., Basu, S., Dhillon, I.S., Mooney, R.J.: Semi-supervised graph clustering: A kernel approach. In: Proc. 22th International Conf. on Machine Learning. (2005)
22. Breitenbach, M., Grudic, G.Z.: Clustering through ranking on manifolds. In: Proc. 22th International Conf. on Machine Learning. (2005)
23. Seneta, E.: Non-Negative Matrices and Markov Chains. Springer-Verlag (1981)
24. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001)
25. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Ranking on data manifolds. In: NIPS(15). (2003) 169–176