Marginalized Multi-Instance Kernels

James T. Kwok **Pak-Ming Cheung**
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
{jamesk,pakming}@cse.ust.hk

Abstract

Support vector machines (SVM) have been highly successful in many machine learning problems. Recently, it is also used for multi-instance (MI) learning by employing a kernel that is defined directly on the bags. As only the bags (but not the instances) have known labels, this MI kernel implicitly assumes all instances in the bag to be equally important. However, a fundamental property of MI learning is that not all instances in a positive bag necessarily belong to the positive class, and thus different instances in the same bag should have different contributions to the kernel. In this paper, we address this instance label ambiguity by using the method of marginalized kernels. It first assumes that all the instance labels are available and defines a label-dependent kernel on the instances. By integrating out the unknown instance labels, a marginalized kernel defined on the bags can then be obtained. A desirable property is that this kernel weights the instance pairs by the consistencies of their probabilistic instance labels. Experiments on both classification and regression data sets show that this marginalized MI kernel, when used in a standard SVM, performs consistently better than the original MI kernel. It also outperforms a number of traditional MI learning methods.

1 Introduction

In supervised learning, each training pattern has a known class label. However, many applications only have weak label information and thus cannot be formulated as supervised learning problems. A classic example as discussed by [Dietterich *et al.*, 1997] is drug activity prediction, where the task is to predict whether a drug molecule can bind to the targets (enzymes or cell-surface receptors). A molecule is considered useful as a drug if one of its conformations can bind to the targets. However, biochemical data can only tell the binding capability of a molecule, but not a particular conformation. In other words, we only have class labels associated with sets of patterns (*bags*), instead of the individual patterns (*instances*). Dietterich *et al.* called this multi-instance (MI) learning. Another well-known MI application is content-based image retrieval [Chen and Wang, 2004], where each image is a bag and each local image patch an instance.

Following Dietterich *et al.*'s seminal work, a number of new MI learning methods, such as the Diverse Density (DD) algorithm [Maron and Lozano-Perez, 1998], have emerged. Here, we will focus on methods based on the support vector machines (SVM), which have been highly successful in many machine learning problems. There are two main approaches to extend the standard SVM for MI data. One is to modify the SVM formulation, as in [Andrews *et al.*, 2003; Cheung and Kwok, 2006]. However, unlike the standard SVM, they lead to non-convex optimization problems which suffer from local minima. Another approach is to design kernels directly on the bags [Gärtner *et al.*, 2002]. These so-called MI kernels can then be used in a standard SVM. As the instance labels are unavailable, the MI kernel implicitly assumes all instances in the bag to be equally important. However, a central assumption in MI learning is that not all the instances in a positive bag are necessarily positive. Thus, many of these instances should have small contributions and the assumption made by the MI kernel is very crude.

Recall that the major difference between MI learning and traditional single-instance learning lies in the ambiguity of the instances labels. If the instance labels were available, then the MI problem could be solved much more easily. A related idea is explored in the EM-DD algorithm [Zhang and Goldman, 2002]. In each bag, the instance that is most consistent with the current hypothesis is selected as the representative of the whole bag. This converts the multiple-instance data to single-instance data and leads to improved speed and accuracy over the DD algorithm. However, it implicitly assumes that there is only one positive instance in each positive bag. In practice, both DD and EM-DD are often inferior to kernel-based MI algorithms [Andrews *et al.*, 2003; Cheung and Kwok, 2006].

In this paper, we address the ambiguity of instance labels in the design of MI kernels by using the method of marginalized kernels [Tsuda *et al.*, 2002]. Similar to the Expectation-Maximization (EM) algorithm, it transforms an incomplete data problem (with only the observed data) to a complete data problem (with both the observed and hidden data), which is then often easier to solve. This method has been successfully used to define marginalized kernels for strings, [Tsuda *et al.*, 2002], trees and graphs [Mahé *et al.*, 2004].

The rest of this paper is organized as follows. Section 2 first gives a brief introduction to the marginalized kernel. Section 3 then describes the proposed marginalized kernel for MI classification, which is followed by an extension to MI regression in Section 4. Experimental results on a number of classification and regression data sets are presented in Section 5, and the last section gives some concluding remarks.

2 Marginalized Kernels

Like the EM algorithm, the data are assumed to be generated by a latent variable model, with observed variable x and hidden variable θ. The task is to define a (marginalized) kernel between two observed variables x_1 and x_2. With the help of the hidden variables, one can first define a joint kernel $k_z(z_1, z_2)$ over the pairs $z_1 = (x_1, \theta_1)$ and $z_2 = (x_2, \theta_2)$. As the hidden information is indeed unobserved, the posterior distribution of θ_1 ad θ_2 are obtained by some probabilistic model $p(\theta|x)$, which in turn is estimated from the data. The marginalized kernel is then obtained by taking expectation of the joint kernel w.r.t. the hidden variables, as:

$$k(x_1, x_2) = \sum_{\theta_1, \theta_2 \in \Theta} P(\theta_1|x_1)P(\theta_2|x_2)k_z(z_1, z_2)d\theta_1\, d\theta_2, \quad (1)$$

where Θ is the domain of the hidden variables. When the hidden variable is continuous, the summation is replaced by an integration. Note that computing (1) may be intractable when Θ is large, and so this is an important issue in designing marginalized kernels.

3 Marginalized Kernels for MI Classification

In MI classification, we are given a set of training bags $\{(B_1, y_1), \ldots, (B_m, y_m)\}$, where $B_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}$ is the ith bag containing instances $\mathbf{x}_{ij}$'s, and $y_i \in \{0, 1\}$. For each bag B_i, the observed information is then the associated $\mathbf{x}_{ij}$'s while the hidden information is the unknown instance labels $\mathbf{c}_i = [c_{i1}, \ldots, c_{in_i}]'$, where $c_{ij} \in \{0, 1\}$.

3.1 The Joint Kernel

The joint kernel is defined on two combined variables $\mathbf{z}_1 = (B_1, \mathbf{c}_1)$ and $\mathbf{z}_2 = (B_2, \mathbf{c}_2)$. It can thus utilize information on both the input ($\mathbf{x}_{ij}$'s) and instance labels ($\mathbf{c}_i$). Note that while traditional kernels (such as the polynomial and Gaussian kernels) are defined on the input part only, recent results show that the use of label information can lead to better kernels (e.g, [Cristianini *et al.*, 2002]).

In this paper, we define the joint kernel as:

$$k_z(\mathbf{z}_1, \mathbf{z}_2) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k_c(c_{1i}, c_{2j})k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j}), \quad (2)$$

where $k_x(\cdot, \cdot)$ and $k_c(\cdot, \cdot)$ are kernels defined on the input and label parts of the instances, respectively. A simple and reasonable definition of k_c is[1]

$$k_c(c_{1i}, c_{2j}) = I(c_{1i} = c_{2j}), \quad (3)$$

where $I(\cdot)$ returns 1 when the predicate is true, and 0 otherwise. In this case, (2) is also equal to the alignment between kernel k_x and the instance labels [Cristianini *et al.*, 2002]. A high alignment thus implies a high kernel value (or similarity) between B_1 and B_2.

[1]Other definitions are also possible, e.g., one can have $k_c(c_{1i}, c_{2j}) = 1$ if $c_{1i} = c_{2j} = 1$; and 0 otherwise.

3.2 Marginalizing the Joint Kernel

To obtain the marginalized kernel, we take expectation of the joint kernel in (2) w.r.t. the hidden variables $\mathbf{c}_1$ and $\mathbf{c}_2$, as:

$$k(B_1, B_2) = \sum_{\mathbf{c}_1, \mathbf{c}_2} P(\mathbf{c}_1|B_1)P(\mathbf{c}_2|B_2)k_z(\mathbf{z}_1, \mathbf{z}_2). \quad (4)$$

Computation of the conditional probability $P(\mathbf{c}_i|B_i)$ will be postponed to Section 3.3. Note that even when $P(\mathbf{c}_i|B_i)$ is known, a direct computation of (4) is still computationally infeasible (and takes $\Omega(2^{n_1+n_2})$ time) as $\mathbf{c}_i \in \{0, 1\}^{n_i}$.

With the joint kernel defined in (2), $k(B_1, B_2)$ in (4) can be simplified as:

$$\begin{aligned}
&\sum_{\mathbf{c}_1, \mathbf{c}_2} P(\mathbf{c}_1|B_1)P(\mathbf{c}_2|B_2)\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k_c(c_{1i}, c_{2j})k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j}) \\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j}) \sum_{c_{1i}, c_{2j}} \{k_c(c_{1i}, c_{2j}) \\
&\quad \cdot \sum_{\mathbf{c}_1 \setminus c_{1i}} P(c_{1i}, \mathbf{c}_1 \setminus c_{1i}|B_1) \sum_{\mathbf{c}_2 \setminus c_{2j}} P(c_{2j}, \mathbf{c}_2 \setminus c_{2j}|B_2)\},
\end{aligned}$$

where $\mathbf{c}_i \setminus c_{ij} = [c_{i1}, \ldots, c_{i,j-1}, c_{i,j+1}, \ldots, c_{in_i}]'$. Using $\sum_{\mathbf{c}_i \setminus c_{ij}} P(c_{ij}, \mathbf{c}_i \setminus c_{ij}|B_i) = P(c_{ij}|B_i)$ and the conditional independence assumption that $P(c_{ij}|B_i) = P(c_{ij}|\mathbf{x}_{ij})$, $k(B_1, B_2)$ can be reduced to

$$\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{c_{1i}, c_{2j}} k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j})k_c(c_{1i}, c_{2j})P(c_{1i}|\mathbf{x}_{1i})P(c_{2j}|\mathbf{x}_{2j}). \quad (5)$$

As will be shown in Section 3.4, this can now be computed in polynomial time. In particular, on using (3) as the instance label kernel, $k(B_1, B_2)$ is simply

$$\begin{aligned}
&\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} P(c_{1i} = 1|\mathbf{x}_{1i})P(c_{2j} = 1|\mathbf{x}_{2j})k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j}) \\
&+ \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} P(c_{1i} = 0|\mathbf{x}_{1i})P(c_{2j} = 0|\mathbf{x}_{2j})k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j}),
\end{aligned}$$

which is very intuitive. Finally, to avoid undesirable scaling problems, we normalize the kernel as in [Gärtner *et al.*, 2002]: $k(B_i, B_j) \leftarrow \frac{k(B_i, B_j)}{\sqrt{k(B_i, B_i)}\sqrt{k(B_j, B_j)}}$.

Note that (5) reduces to the MI kernel $k(B_i, B_j) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j})$ in [Gärtner *et al.*, 2002] if $k_c(\cdot, \cdot)$ is constant. Thus, while [Gärtner *et al.*, 2002] assumes that all the instance pairs between B_i and B_j are equally important, k_c in (5) weights them differently according to the consistency of their probabilistic labels. Moreover, compared to EM-DD which chooses only one representative instance from each bag during inference, here we perform marginalization over all possible instance labels and is thus more consistent with the Bayesian framework.

3.3 Defining the Conditional Probabilities

In this section, we consider how to obtain the conditional probability $P(\mathbf{c}_i|B_i)$. As mentioned in [Tsuda *et al.*, 2002], an advantage of the marginalized kernel approach is that the definitions of the joint kernel and probabilistic model are completely separated. Thus, one has a lot of freedom in picking a good probabilistic model.

Using the Probabilistic Model for Diverse Density
Motivated by the success of the DD algorithm in MI learning [Maron and Lozano-Perez, 1998], we first consider using its probabilistic model for estimating $P(c_{ij}|\mathbf{x}_{ij})$. The DD algorithm finds a hypothesis h over the whole instance space[2] $\mathcal{X}$ by maximizing the following DD function:

$$DD(h)=\prod_{B_i^+}\left(1-\prod_{\mathbf{x}_{ij}}\left(1-e^{-\|h-\mathbf{x}_{ij}\|^2}\right)\right)\prod_{B_i^-}\prod_{\mathbf{x}_{ij}}\left(1-e^{-\|h-\mathbf{x}_{ij}\|^2}\right).$$

Here, B_i^+ and B_i^- index all the positive and negative bags in the training set, respectively. Intuitively, h has a high DD value if all positive bags have instances close to h, while negative instances from all the negative bags are far away from h. Under this model, we can then define

$$P(h|\mathcal{D}) = DD(h)/Z, \tag{6}$$

where Z is a normalizing factor such that $\sum_h P(h|\mathcal{D}) = 1$,

$$P(c_{ij} = 1|\mathbf{x}_{ij}) = e^{-\|\mathbf{x}_{ij}-h\|^2} \tag{7}$$

and $P(c_{ij} = 0|\mathbf{x}_{ij}) = 1 - P(c_{ij} = 1|\mathbf{x}_{ij})$.

However, the DD function is highly nonlinear with many local minima. Hence, instead of using only one h obtained from the optimization process, it is often advantageous to use multiple hypotheses [Chen and Wang, 2004]. We then have[3]

$$\begin{aligned} P(c_{ij}|\mathbf{x}_{ij}) &= \sum_h P(h|\mathbf{x}_{ij})P(c_{ij}|\mathbf{x}_{ij}, h) \\ &= \sum_h P(h|\mathcal{D})P(c_{ij}|\mathbf{x}_{ij}, h), \end{aligned} \tag{8}$$

where the summation is over the set of hypotheses obtained. Note that (8) automatically weights each hypothesis by its likelihood. On the contrary, the DD-SVM in [Chen and Wang, 2004] has no such weighting and has to rely on additional heuristics to filter away the less important hypotheses.

Substituting all these equations into (5), we finally have

$$\begin{aligned} k(B_1, B_2)=\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{h_1,h_2}\sum_{c_{1i},c_{2j}} & k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j})k_c(c_{1i}, c_{2j}) \\ & P(h_1|\mathcal{D})P(h_2|\mathcal{D})P(c_{1i}|\mathbf{x}_{1i}, h_1)P(c_{2j}|\mathbf{x}_{2j}, h_2). \end{aligned} \tag{9}$$

The complete algorithm is shown in Algorithm 1.

Very recently, [Rahmani and Goldman, 2006] proposed a graph-based MI semi-supervised learning method, where the edge weight between bags B_1 and B_2 is roughly equal to[4]

$$\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\frac{DD(\mathbf{x}_{1i}) + DD(\mathbf{x}_{2j})}{2}k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j}). \tag{10}$$

This is quite similar to (9). However, in general, (10) is not positive semi-definite and so not a valid kernel function.

[2]Searching over a large $\mathcal{X}$ can be prohibitive. To be computationally efficient, one often performs a gradient-based optimization of the DD function by initializing from every instance in every positive bag [Chen and Wang, 2004; Maron and Lozano-Perez, 1998].

[3]Note that all the probabilities here are implicitly conditioned on the training data $\mathcal{D}$. For clarity, in (8) we write $P(h|\mathcal{D})$ (the *posterior* probability of h given the data), instead of $P(h)$.

[4]In [Rahmani and Goldman, 2006], the DD values in (10) are first normalized, nonlinearly transformed and then filtered. Moreover, this weight is defined on positive bags only.

Algorithm 1 Marginalized MI kernel for classification.

Input: Training set $\mathcal{D}$; A pair of bags B_1 and B_2
Output: $k(B_1, B_2)$

1: Run the DD algorithm with different initializations and store the hypotheses obtained in the array $\mathcal{H}$.
2: **for all** $h \in \mathcal{H}$ **do**
3: Compute $P(h|\mathcal{D})$ using (6) or (11) and store the value in array Q.
4: **end for**
5: **for all** $\mathbf{x}_{ij} \in B_1 \cup B_2$ **do**
6: Compute $P(c_{ij}|\mathbf{x}_{ij})$ using (7) and (8).
7: **end for**
8: Compute $k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j})$ for all instance pairs in $B_1 \times B_2$.
9: Compute $k(B_1, B_2)$ as in (9), using the pre-computed values in the array Q, k_x and $P(c_{ij}|\mathbf{x}_{ij})$.

Using the Training Accuracy to Improve $P(h|\mathcal{D})$
As mentioned earlier, the DD algorithm has been very successful. Note that one only has to use the hypotheses, but not their DD values, on classification. Recall that the obtained hypotheses are local minima of the DD function. While many of them may have comparable classification accuracies, as will be demonstrated in Section 5, a few hypotheses can have DD values that are much higher than the others. This is not surprising as DD value may drop significantly even if the hypothesis is close to only one negative instance. Consequently, the summation in (8) is often dominated by a few hypotheses.

To alleviate this problem while still retaining the merits of DD, we will instead define $P(h|\mathcal{D})$ of each DD hypothesis to be proportional to its training accuracy. Let $\text{pred}_h(B_i)$ be the label predicted by h on B_i, which is equal to 1 if $\max_{\mathbf{x}_i \in B_i} e^{-\|\mathbf{x}_{ij}-h\|^2} \geq 0.5$, and 0 otherwise. Then,

$$P(h|\mathcal{D}) = \sum_{i=1}^{m} I(\text{pred}_h(B_i) = y_i)/Z, \tag{11}$$

where m is the number of training bags and Z is again for normalization. Its superiority over the definition in (6) will be experimentally verified in Section 5.

Because of the modular design of the marginalized kernel, we can easily plug in other good estimators of $P(c_{ij}|\mathbf{x}_{ij})$. For example, an EM-based approach may also be used.

3.4 Time Complexity

Let N_h be the number of DD hypotheses. In Algorithm 1, computing all the $P(c_{ij}|\mathbf{x}_{ij})$'s at Step 6 takes $O((n_1 + n_2)N_h d)$ time (assuming that $P(h|\mathcal{D})$'s can be obtained in constant time or have been pre-computed), where d is the data dimensionality. Assuming that each evaluation of the kernel k_x takes $O(d)$ time, then computing all the $k_x(\cdot, \cdot)$'s in Step 8 takes $O(n_1 n_2 d)$ time. The summation involved in $k(B_1, B_2)$ of Step 9 takes $O(N_h^2 n_1 n_2)$ time (as c_{ij} only takes 0 or 1). Thus, the total time complexity for computing $k(B_1, B_2)$ is $O((N_h^2 + d)n_1 n_2 + (n_1 + n_2)N_h d)$, which is polynomial.

4 Marginalized Kernels for MI Regression

In regression, we assume that each bag output y_i is normalized to the range $[0, 1]$. Subsequently, each (hidden) instance

label c_{ij} is also in $[0, 1]$. In this real-valued setting, we follow the extended DD model in [Amar *et al.*, 2001], and have:

$$P(c_{ij}|\mathbf{x}_{ij}) = \left(1 - |c_{ij} - e^{-\|h-\mathbf{x}_{ij}\|^2}|\right)/Z(\mathbf{x}_{ij}), \quad (12)$$

where $Z(\mathbf{x}_{ij})$ ensures that $\int_0^1 P(c_{ij}|\mathbf{x}_{ij})dc_{ij} = 1$. It can be easily shown that $Z(\mathbf{x}_{ij}) = 0.75 - (e^{-\|h-\mathbf{x}_{ij}\|^2} - 0.5)^2$.

Proceeding as in Section 3, we employ the same joint kernel in (2) (but with summations of c_{1i} and c_{2j} replaced by integrations over $[0, 1]$) in the marginalized kernel $k(B_1, B_2)$ of (5). Again, this can then avoid integrating over an $(n_1 + n_2)$-dimensional space.

The probability that two instances share the same real-valued label is zero, and so (3) is a poor measure of instance label similarity. Intuitively, the larger the difference between c_{1i} and c_{2j}, the smaller is $k_c(c_{1i}, c_{2j})$. Noting that $|c_{1i} - c_{2j}| \in [0, 1]$, we have two natural choices for k_c (Figure 1)[5]:

$$\text{(linear)} \quad k_c(c_{1i}, c_{2j}) = (1 - |c_{1i} - c_{2j}|), \quad (13)$$

$$\text{(Gaussian)} \quad k_c(c_{1i}, c_{2j}) = \exp\left(-\beta(c_{1i} - c_{2j})^2\right). \quad (14)$$

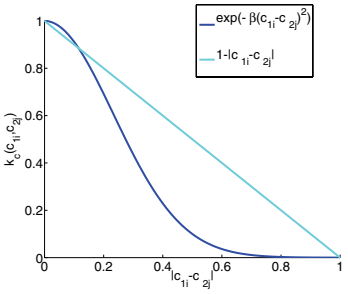


Figure 1: Two possible definitions of the label kernel $k_c(c_{1i}, c_{2j})$ in the regression setting.

Putting all these together, and after long but straightforward calculations, it can be shown that the marginalized MI kernel between bags B_1 and B_2 is given by:

$$k(B_1, B_2) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \frac{k(\mathbf{x}_{1i}, \mathbf{x}_{2j})G(e^{-\|h-\mathbf{x}_{1i}\|^2}, e^{-\|h-\mathbf{x}_{2j}\|^2}))}{Z(\mathbf{x}_{1i})Z(\mathbf{x}_{2j})},$$

where $G(u, v)$ is equal to

$$\int_0^1\int_0^1 (1 - |c_{1i} - c_{2j}|)\,(1 - |c_{1i} - u|)(1 - |c_{2j} - v|)dc_{1i}\,dc_{2j}, \quad (15)$$

when (13) is used; and is equal to

$$\int_0^1\int_0^1 \exp\left(-\beta(c_{1i}-c_{2j})^2\right)(1-|c_{1i}-u|)(1-|c_{2j}-v|)dc_{1i}\,dc_{2j}, \quad (16)$$

when (14) is used. Closed form expressions for these two integrals can be found in the Appendix. Moreover, as in classification, both definitions of $k(B_1, B_2)$ can be computed in polynomial time.

[5]In the experiments, we set β such that $k_c(c_{1i}, c_{2j}) = 0.0001$ when $|c_{1i} - c_{2j}| = 1$.

When multiple hypotheses are used, $k(B_1, B_2)$ can be similarly generalized to:

$$\sum_{i,j,h_1,h_2} \frac{P(h_1|D)P(h_2|D)k(\mathbf{x}_{1i}, \mathbf{x}_{2j})G(e^{-\|h_1-\mathbf{x}_{1i}\|^2}, e^{-\|h_2-\mathbf{x}_{2j}\|^2})}{Z(\mathbf{x}_{1i})Z(\mathbf{x}_{2j})}.$$

As for the probability $P(h|\mathcal{D})$, again one can follow the extended DD model and obtain:

$$P(h|\mathcal{D}) = \prod_{B_i}\left(1 - |y_i - \max_{x_{ij}\in B_i} e^{-\|h-\mathbf{x}_{ij}\|^2}|\right)/Z.$$

Alternatively, as in (11), we can also use the training accuracy of each hypothesis. Following [Dooly *et al.*, 2002], we use $\max_{\mathbf{x}_{ij}\in B_i} e^{-\|h-\mathbf{x}_{ij}\|^2}$ as h's predicted output of bag B_i. The error of h is then $e_h = \sum_{i=1}^m \left(y_i - \max_{\mathbf{x}_{ij}\in B_i} e^{-\|h-\mathbf{x}_{ij}\|^2}\right)^2$, and we can define

$$P(h|\mathcal{D}) = \left(1 - \frac{e_h - \min_{h'} e_{h'}}{\max_{h'} e_{h'} - \min_{h'} e_{h'}}\right)/Z. \quad (17)$$

5 Experiments

In this section, we compare the performance of the proposed marginalized kernels with the MI kernel in [Gärtner *et al.*, 2002] on a number of MI data sets. As for the kernel defined on the input parts, we use the Gaussian kernel $k_x(\mathbf{x}_{1i}, \mathbf{x}_{2j}) = \exp\left(-\gamma\left\|\mathbf{x}_{1i} - \mathbf{x}_{2j}\right\|^2\right)$, where γ is the width parameter.

5.1 MI Classification

For simplicity, the marginalized kernels defined in (6) and (11) are denoted by MG-DDV and MG-ACC, respectively.

Drug Activity Prediction

The first experiment is performed on the popular Musk1 and Musk2 data sets[6]. The task is to predict if a drug is musky. As mentioned in Section 1, each drug is considered a bag, and each of its low-energy conformations an instance. We use 10-fold cross-validation. In each fold, the DD algorithm is applied only on the *training* data and so the DD hypotheses cannot capture any information in the test set. For comparison, the DD-SVM [Chen and Wang, 2004] is also run.

As can be seen from Table 1, the MG-ACC kernel always performs better than the MI kernel. It also outperforms the DD-SVM, which is also using multiple DD hypotheses that are assumed to be equally important. This thus demonstrates the importance of weighting instance pairs based on the consistency of their underlying labels (Section 3.2).

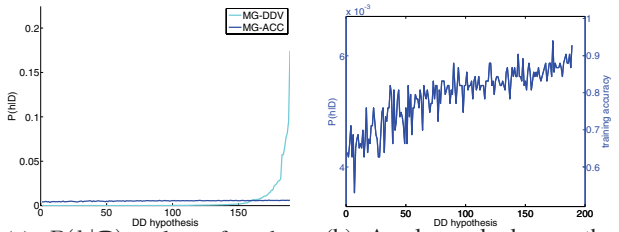
Table 1: Testing accuracies (%) on the Musk data sets.

	Musk1	Musk2
DD-SVM	89.0	87.9
MI kernel	89.3	87.5
MG-DDV kernel	87.8	88.6
MG-ACC kernel	**90.1**	**90.4**

Moreover, the MG-ACC kernel is better than the MG-DDV kernel. Figure 2(a) compares their corresponding $P(h|\mathcal{D})$

[6]ftp://ftp.ics.uci.edu/pub/machine-learning-databases/musk/

values ((6) for MG-DDV and (11) for MG-ACC) for all the DD hypotheses obtained from a typical fold of the Musk1 data. As can be seen, for the MG-DDV kernel, a small subset of $P(h|\mathcal{D})$ values are very high, and so the summation in (8) is dominated by these few hypotheses. On the other hand, for the MG-ACC kernel, the $P(h|\mathcal{D})$ values vary much gradually (Figure 2(b)), and hence more hypotheses can be utilized in computing $P(c_{ij}|\mathbf{x}_{ij})$. Moreover, recall that $P(h|\mathcal{D})$ in (6) is proportional to h's DD value, while the one in (11) is proportional to h's training accuracy. Hence, the almost linear trend in Figure 2(b) shows that the higher the DD value, the higher is the training accuracy of the hypothesis. It thus provides another evidence for the success of the DD algorithm.



(a) $P(h|\mathcal{D})$ values for the MG-DDV and MG-ACC kernels.

(b) A closer look on the $P(h|\mathcal{D})$ values for the MG-ACC kernel.

Figure 2: $P(h|\mathcal{D})$ values obtained from a typical fold of the Musk1 data. The x-axis shows the different hypotheses obtained by the DD algorithm, sorted in increasing $P(h|\mathcal{D})$ values as obtained for the MG-ACC kernel.

Image Categorization

The second experiment is on an image categorization task using the data set[7] in [Chen and Wang, 2004]. There are 10 classes (beach, flowers, horses, etc.), with each class containing 100 images. Each image is regarded as a bag, and each segment an instance. We follow exactly the same setup as [Chen and Wang, 2004]. The data is randomly divided into a training and test set, each containing 50 images of each category. Since this is a multi-class classification problem, we employ the standard one-vs-rest approach to convert it to a number of binary classification problems. The experiment is repeated 5 times, and the average testing accuracy reported.

Table 2 shows the results, along with those of the DD-SVM, Hist-SVM [Chapelle *et al.*, 1999] and MI-SVM as reported in [Chen and Wang, 2004]. As can be seen, the MG-ACC kernel is again superior to the MI kernel and others. To ensure that the improvement over the MI kernel is statistically significant, we repeat the experiment 50 times (instead of only 5). The difference is confirmed to be significant at the 0.05 level of significance by using the paired t-test.

5.2 MI Regression: Synthetic Musk Molecules

We perform MI regression on the data set used in [Dooly *et al.*, 2002], where the goal is to predict the binding energies of the musk molecules. We use three data sets (LJ-16.30.2, LJ-80.166.1 and LJ-160.166.1) downloaded from the author's website[8], and three more[9] (LJ-16.50.2, LJ-80.206.1 and LJ-160.566.1) used in a recent study [Cheung and Kwok, 2006]. The latter were obtained by adding irrelevant features to the former data sets, while keeping its real-valued outputs intact.

Table 2: Testing accuracies (%) on the image data set.

	5 repetitions	50 repetitions
DD-SVM	81.50 ± 3.00	-
Hist-SVM	66.70 ± 2.20	-
MI-SVM	74.70 ± 0.60	-
MI kernel	84.12 ± 0.90	84.12 ± 1.22
MG-DDV kernel	76.52 ± 17.29	-
MG-ACC kernel	$\mathbf{84.64} \pm 1.28$	$\mathbf{84.54} \pm 1.21$

As demonstrated in Section 5.1, the MG-ACC kernel is superior than the MG-DDV kernel. Hence, we will only experiment with the MG-ACC kernel defined with (17) here. As in [Dooly *et al.*, 2002], we report both the percentage error (%err)[10] and mean squared error (MSE).

Table 3 shows the results, along with those of DD, EM-DD and citation-kNN [Wang and Zucker, 2000] as reported in [Cheung and Kwok, 2006]. As can be seen, the proposed MG-ACC kernel with the Gaussian instance label kernel k_c in (14) consistently outperforms the others. Its superiority over the MG-ACC kernel (with a linearly decaying instance label kernel) indicates that a much smaller weight should be assigned to instance pairs whose labels are quite different (Figure 1). This also explains the relatively inferior performance of the MI kernel, which weights all instance pairs equally.

6 Conclusion

In this paper, we show how to design marginalized kernels in multiple-instance learning. First, we pretend that all the hidden instance labels are known, and define a joint kernel using both the instance inputs and labels. By integrating out the hidden data, the marginalized kernel is obtained. This kernel differs from the existing MI kernel in that different instance pairs are weighted by the consistency of their probabilistic labels. Experimentally, this marginalized MI kernel always performs better than the MI kernel and also often outperforms other traditional MI learning methods.

Note that the proposed kernel can be straightforwardly used in the regularization framework recently proposed in [Cheung and Kwok, 2006]. Moreover, because of the modularity of the marginalized kernel, we will also explore even better definitions of the joint kernel and probabilistic model.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region.

[7] http://www.cs.uno.edu/∼yixin/ddsvm.html

[8] http://www.cs.wustl.edu/∼sg/multi-inst-data. There are four more data sets available. However, they are easier variations of the three that we used, and so are dropped in this study.

[9] http://www.cs.ust.hk/∼jamesk/papers/icml06_data.zip

[10] %err is the classification error obtained by thresholding both the target output and the predicted output at 0.5.

Table 3: Performance of MI regression on the synthetic affinity data sets.

	DD		EM-DD		citation-kNN		SVM (MI kernel)		SVM (MG-ACC kernel) linear (13)		SVM (MG-ACC kernel) Gaussian (14)	
data set	%err	MSE	%err	MSE	%err	MSE	%err	MSE	%err	MSE	%err	MSE
LJ-16.30.2	**6.7**	0.0240	**6.7**	**0.0153**	16.7	0.0260	8.3	0.0197	8.3	0.0197	8.3	0.0186
LJ-80.166.1	(not available)		37.0	0.1825	8.6	0.0109	7.6	0.0121	7.6	0.0119	**6.5**	**0.0102**
LJ-160.166.1	23.9	0.0852	21.7	0.0850	4.3	**0.0014**	**0.0**	0.0053	**0.0**	0.0052	**0.0**	0.0044
LJ-16.50.2	-	-	40.0	0.2357	53.3	0.0916	10.0	0.0202	10.0	0.0202	**6.7**	**0.0191**
LJ-80.206.1	-	-	37.0	0.2449	30.4	0.0463	6.5	0.0110	6.0	0.0098	**5.4**	**0.0089**
LJ-160.566.1	-	-	37.0	0.2414	34.8	0.0566	1.1	0.0056	0.5	0.0050	**0.0**	**0.0046**

A Appendix

Using Mathematica, it can be shown that (16),

$$\begin{aligned} G(u,v) = \quad & \tfrac{1}{12\beta^2}\,(2+g_0(u,v)+g_1(u,v)+g_1(v,u) \\ & +g_2(u,v)+g_2(v,u)+g_3(u,v)+g_4(u,v)) \\ & +\tfrac{\sqrt{\pi}}{24\beta^{3/2}}\,(g_5(u,v)+g_5(v,u)+g_6(u,v)+g_6(v,u) \\ & +g_7(u,v)+g_8(u,v))\,, \end{aligned}$$

where

$$\begin{aligned} g_0(u,v)&=4(\beta(u-v)^2+1)\exp\left(-\beta(u-v)^2\right), \\ g_1(u,v)&=(\beta(1-u)(1+2u-6v)-2)\exp\left(-\beta(1-u)^2\right), \\ g_2(u,v)&=(\beta u(6v-2u-3)-2)\exp\left(-\beta u^2\right), \\ g_3(u,v)&=(\beta(6u+6v-12uv+8)+2)\exp\left(-\beta\right), \\ g_4(u,v)&=6\beta(u+v-2uv-2), \\ g_5(u,v)&=\left((2\beta u^2+1)(6v-2u-3)-4u\right)\mathrm{erf}\left(\sqrt{\beta}u\right), \\ g_6(u,v)&=(2\beta(1+2u-6v)(1-u)^2+6(u-v)-3)\mathrm{erf}\left(\sqrt{\beta}(1-u)\right), \\ g_7(u,v)&=\left(8\beta(u-v)^2+12\right)(u-v)\mathrm{erf}\left(\sqrt{\beta}(u-v)\right), \\ g_8(u,v)&=\left(12\beta(u-y)^2+16\beta+6\right)\mathrm{erf}\left(\sqrt{\beta}\right), \end{aligned}$$

and $\mathrm{erf}(u) = \frac{2}{\sqrt{\pi}}\int_0^u \exp\left(-t^2\right)dt$ is the so-called error function. Similarly, (15) can be shown to be

$$\begin{aligned} G(u,v) \quad = \quad & \frac{-1}{30}\,|u-v|^5 + \frac{1}{6}(u-v)^4 + \frac{1}{3}uv(u^2+v^2) \\ & -\frac{1}{6}(u+v)^3 - \frac{1}{3}(u-v)^2 + \frac{1}{3}(u+v) + \frac{11}{60}. \end{aligned}$$

References

[Amar *et al.*, 2001] R.A. Amar, D.R. Dooly, S.A. Goldman, and Q. Zhang. Multiple-instance learning of real-valued data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 3–10, Williamstown, MA, USA, 2001.

[Andrews *et al.*, 2003] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[Chapelle *et al.*, 1999] O. Chapelle, P. Haffner, and V.N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, September 1999.

[Chen and Wang, 2004] Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

[Cheung and Kwok, 2006] P.M. Cheung and J.T. Kwok. A regularization framework for multiple-instance learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 193–200, Pittsburgh, USA, June 2006.

[Cristianini *et al.*, 2002] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[Dietterich *et al.*, 1997] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

[Dooly *et al.*, 2002] D.R. Dooly, Q. Zhang, S.A. Goldman, and R.A. Amar. Multiple-instance learning of real-valued data. *Journal of Machine Learning Research*, 3:651–678, 2002.

[Gärtner *et al.*, 2002] T. Gärtner, P.A. Flach, A. Kowalczyk, and A.J. Smola. Multi-instance kernels. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 179–186, Sydney, Australia, July 2002.

[Mahé *et al.*, 2004] P. Mahé, N. Ueda, T. Akutsu, J.L. Perret, and J.P. Vert. Extensions of marginalized graph kernels. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 552–559, Banff, Alberta, Canada, July 2004.

[Maron and Lozano-Perez, 1998] O. Maron and T. Lozano-Perez. A framework for multiple-instance learning. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10*. Morgan Kaufmann, San Mateo, CA, 1998.

[Rahmani and Goldman, 2006] R. Rahmani and S.A. Goldman. MISSL: Multiple-instance semi-supervised learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 705–712, Pittsburgh, PA, USA, 2006.

[Tsuda *et al.*, 2002] K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18:S268–S275, 2002.

[Wang and Zucker, 2000] J. Wang and J.-D. Zucker. Solving multiple-instance problem: A lazy learning approach. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1119–1125, Stanford, CA, USA, 2000.

[Zhang and Goldman, 2002] Q. Zhang and S.A. Goldman. EM-DD: An improved multiple-instance learning. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.