Multiple Instance Classification via Successive Linear Programming

O. L. Mangasarian * Edward W. Wild [1]

Computer Sciences Department

University of Wisconsin

Madison, WI 53706

Abstract

The multiple instance classification problem [6,2,12] is formulated using a linear or nonlinear kernel as the minimization of a linear function in a finite dimensional (noninteger) real space subject to linear and bilinear constraints. A linearization algorithm is proposed that solves a succession of fast linear programs that converges in a few iterations to a local solution. Computational results on a number of datasets indicate that the proposed algorithm is competitive with the considerably more complex integer programming and other formulations. A distinguishing aspect of our linear classifier not shared by other multiple instance classifiers is the sparse number of features it utilizes. In some tasks the reduction amounts to less than one percent of the original features.

Key words: Multiple instance learning, support vector machines, successive linearization algorithm

---

1 Introduction

The multiple instance classification problem was introduced in [6,2,12] and consists of classifying *positive and negative bags* of points in the n-dimensional real space R^n on the following basis. Each bag contains a number of points and a classifier correctly classifies all the bags if for each positive bag *at least one* point in the bag is classified as positive, and for each negative bag *all* the points in the bag are classified as negative. Various formulations of the multiple instance classification problem including integer programming, expectation maximization, and kernel formulations have been proposed [1,9,22,23]. Ray and Craven [19] provide an empirical comparison of several multiple instance classification algorithms and their non-multiple-instance counterparts.

We propose here a novel mathematical programming formulation of the multiple instance classification problem that leads to an efficient successive linearization algorithm that typically converges in a few steps to a local solution of the problem which for a linear classifier utilizes as little as one percent of problem features. Our formulation uses a linear or nonlinear kernel support vector machine classifier [20,4,21,14] and is based on the following simple ideas. For a linear classifier, a positive bag is classified correctly *if and only if* some convex combination of points in the bag lies on the positive side of a separating plane. For a nonlinear kernel classifier, a similar statement applies to the higher dimensional space induced by the kernel. This leads to a constrained optimization problem where the objective function and constraints are linear except for a set of bilinear constraints corresponding to the positive bags. A

* www.cs.wisc.edu/∼olvi. Also, Department of Mathematics, University of California at San Diego, La Jolla, CA 92093. Email: olvi@cs.wisc.edu
[1] www.cs.wisc.edu/∼wildt. Email: wildt@cs.wisc.edu

local solution to this formulation is obtained by solving a sequence of linear programs that terminate in a few iterations.

The outline of the paper is as follows. In Section 2 we give our formulation of the multiple instance classification problem and state some of its properties. In Section 3 we state our algorithm and give its convergence. In Section 4 we present our numerical tests on five datasets. Section 5 concludes the paper.

We now describe our notation. All vectors will be column vectors unless transposed to a row vector by a prime $'$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. For $x \in R^n$ and $1 \leq p < \infty$, the p-norm and the ∞-norm are defined as follows:

$$\|x\|_p = \left(\sum_{j=1}^{n} |x_j|^p\right)^{\frac{1}{p}}, \ \|x\|_\infty = \max_{1\leq j\leq n} |x_j|.$$

The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix A' will denote the transpose of A, A_i will denote the i-th row of A, and $A_{\cdot j}$ will denote the j-th column of A. A vector of ones in a real space of arbitrary dimension will be denoted by e. Thus for $e \in R^m$ and $y \in R^m$ the notation $e'y$ will signify the summation $\sum_{i=1}^{m} y_i$. A vector of zeros in a real space of arbitrary dimension will be denoted by 0. The identity matrix of arbitrary dimension will be denoted by I. A *separating plane*, with respect to two given point sets $\mathcal{A}$ and $\mathcal{B}$ in R^n, is a plane that attempts to separate R^n into two halfspaces such that each open halfspace contains points mostly of $\mathcal{A}$ or $\mathcal{B}$. A bilinear function is the product of two linear functions. A *bounding plane* to the set $\mathcal{A}$ is a plane that places $\mathcal{A}$ in one of the two closed halfspaces that the plane generates. For $A \in R^{m\times n}$ and $B \in R^{n\times k}$, a *kernel* $K(A, B)$ maps $R^{m\times n} \times R^{n\times k}$ into $R^{m\times k}$. In particular, if x and y are column vectors in R^n, then $K(x', y)$ is a real number, $K(x', A')$ is a row vector in R^m and $K(A, A')$ is an $m \times m$ matrix. The base of the natural logarithm will be denoted by ε. A frequently used kernel in nonlinear classification that will be employed here is the Gaussian kernel [21,14] whose $ij-$th element, $i = 1, \dots, m,\ j = 1, \dots, k$, is given by: $(K(A, B))_{ij} = \varepsilon^{-\mu \|A_i{}' - B_{\cdot j}\|^2}$, where $A \in R^{m \times n}$, $B \in R^{n \times k}$ and μ is a positive constant. For simplicity, the dimensionality of some vectors will not be explicitly given. The abbreviation "s.t." stands for "such that".

2 Problem Formulation

2.1 Linear Kernel Classifier

Let the positive bags be represented by the k matrices $B^i \in R^{m^i \times n}$, $i = 1, \dots, k$, where row ℓ, $B^i_\ell \in R^n$, of the matrix B^i represents point ℓ of bag i with $\ell = 1, \dots, m^i$. Similarly, we shall represent the negative bags by the $m - k$ matrices $C^i \in R^{m^i \times n}$, $i = k + 1, \dots, m$, where row ℓ, $C^i_\ell \in R^n$, of the matrix C^i represents point ℓ of bag i with $\ell = 1, \dots, m^i$. We shall first use a linear classifier given by the separating plane:

$$x'w = \gamma, \tag{1}$$

where $w \in R^n$ is the normal to a plane that attempts to separate the positive and negative bags, while γ determines the location of the plane relative to the origin in R^n. The separation will be achieved by attempting to place all the points in the negative bags in the halfspace $\{x | x'w \leq \gamma - 1\}$ and at the same time placing *some* convex combination of points of each positive bag in the halfspace $\{x | x'w \geq \gamma + 1\}$ while maximizing the margin (distance) $\frac{2}{\|w\|_1}$

between the bounding planes $x'w = \gamma \pm 1$ using the ∞-norm to measure the margin, as described in [13] . Our mathematical programming formulation will be based on the simple observation that a positive bag will be classified as being in the halfspace $\{x|x'w \geq \gamma + 1\}$ by the separating plane (1) if and only if some convex combination of points in that bag lies in the positive half-space. This leads to the following mathematical program with some positive parameter ν that weights data fitting versus generalization:

$$
\begin{aligned}
\min_{w,\gamma,y,v^1,\ldots,v^k} \quad & \nu e'y + \|w\|_1 \\
\text{s.t. } & v^{i'}B^i w - \gamma + y_i \geq 1,\ i = 1,\ldots,k, \\
& -C^i_\ell w + \gamma + y^i_\ell \geq 1,\ i = k+1,\ldots,m,\ \ell = 1,\ldots,m^i, \\
& e'v^i = 1,\ v^i \geq 0,\ i = 1,\ldots,k, \\
& y \geq 0.
\end{aligned}
\tag{2}
$$

Here, the vector y with components $y_i,\ i = 1,\ldots,k$ and $y^i_\ell,\ i = k+1,\ldots,m,\ \ell = 1,\ldots,m^i$ represents nonnegative slack variables that are driven towards zero by the objective function term $\nu e'y$. The objective function term $\|w\|_1$ represents twice the reciprocal of the soft margin between the bounding planes $x'w = \gamma \pm 1$. Other than the first set of k constraints which are bilinear, the optimization problem has a linear objective function and constraints. We note that the term $\|w\|_1$ is easily converted to a linear term $e's$ with the added constraint $s \geq w \geq -s$. We shall propose and establish convergence to a local solution of our formulation (2) of the multiple-instance problem via a successive linearization algorithm in the next section after we have formulated the nonlinear kernel problem.

2.2 Nonlinear Kernel Classifier

We now describe how to generate a nonlinear classifier via a nonlinear kernel formulation. We replace the separating plane (1) by the nonlinear separating surface:

$$K(x', H')u = \gamma, \tag{3}$$

where $u \in R^m$ is a dual variable and the $m \times n$ matrix H is defined as:

$$H' = [B^{1'} \dots B^{k'}\ C^{k+1'} \dots C^{m'}] \tag{4}$$

and $K(x', H')$ is an arbitrary kernel map from $R^n \times R^{n \times \bar{m}}$ into $R^{\bar{m}}$, where $\bar{m} = \sum_{i=1}^{m} m^i$. We note that the linear classifier (1) is recovered from (3) if we use the linear kernel $K(x', H') = x'H'$ and define $w = H'u$. With this nonlinear kernel formulation, the mathematical program for generating the nonlinear classifier becomes the following, upon kernelizing (2):

$$\begin{array}{ll} \min\limits_{u,\gamma,y,v^1,\dots,v^k} & \nu e'y + \|u\|_1 \\ \text{s.t. } & v^{i'}K(B^i, H')u - \gamma + y_i \geq 1,\ i = 1, \dots, k, \\ & -K(C^i_\ell, H')u + \gamma + y^i_\ell \geq 1,\ i = k+1, \dots, m,\ \ell = 1, \dots, m^i, \\ & e'v^i = 1,\ v^i \geq 0,\ i = 1, \dots, k, \\ & y \geq 0. \end{array} \tag{5}$$

We note again that the only nonlinearity is in the first k bilinear constraints. A necessary and sufficient condition for nonlinear kernel separation is that

$y < e$.

3 Multiple Instance Classification Algorithm

Since only the first constraints in our multiple instance formulation are nonlinear, and in fact are bilinear, an obvious method of solution suggests itself as follows. Alternately hold one set of variables that constitute the bilinear terms constant while varying the other set. This leads to the successive solution of linear programs that underly our algorithm which we specify now.

Algorithm 3.1 MICA: Multiple Instance Classification Algorithm with a Nonlinear Kernel K

(0) *Choose as initial guess for $v^{i0} = \frac{e}{m^i}$, $i = 1, \ldots, k$. Set counter $r = 0$.*
Note: This initial choice of $v^{10}, \ldots, v^{k0}$ results in using the mean for each positive bag in (i) below.

(i) *For fixed $v^{1r}, \ldots, v^{kr}$, where v^{ir} is iterate r for v^i, solve the following linear program for (u^r, γ^r, y^r) for some positive value of the parameter ν:*

$$\begin{array}{ll} \min\limits_{u,\gamma,y} & \nu e'y + \|u\|_1 \\ \text{s.t. } v^{ir'}K(B^i, H')u - \gamma + y_i \geq 1,\ i = 1, \ldots, k, & \\ \quad -K(C^i_\ell, H')u + \gamma + y^i_\ell \geq 1,\ i = k+1, \ldots, m,\ \ell = 1, \ldots, m^i, & \\ \quad y \geq 0. & \end{array} \tag{6}$$

(ii) *For u^r fixed at the value obtained in (i), solve the following linear program* *for* $(\gamma, y, v^{1(r+1)}, \ldots, v^{k(r+1)})$*:*

$$\begin{array}{rl} \min\limits_{\gamma, y, v^1, \ldots, v^k} & e'y \\ & s.t.\ v^{i'}K(B^i, H')u^r - \gamma + y_i \geq 1,\ i = 1, \ldots, k, \\ & -K(C^i_\ell, H')u^r + \gamma + y^i_\ell \geq 1,\ i = k+1, \ldots, m,\ \ell = 1, \ldots, m^i, \\ & e'v^i = 1,\ v^i \geq 0,\ i = 1, \ldots, k, \\ & y \geq 0. \end{array} \tag{7}$$

(iii) Stop if $\|(v^{1(r+1)} - v^{1r}, \ldots, v^{k(r+1)} - v^{kr})\|_2$ *is less than some desired tolerance. Else replace* $(v^{1r}, \ldots, v^{kr})$ *by* $(v^{1(r+1)}, \ldots, v^{k(r+1)})$*,* r *by* $r + 1$ *and go to (i).*

Since the objective function of our original multiple instance formulation (5) is bounded below by zero and is nonincreasing in the iterations (i) and (ii) of the MICA Algorithm 3.1, it must converge. We can state the following convergence result.

Proposition 3.2 : Convergence to a Local Minimum Value *The nonnegative nonincreasing values of the sequence of objective function values* $\{\nu e'y^r + \|u^r\|_1\}_{r=1}^{r=\infty}$ *converges to* $(\nu e'\bar{y} + \|\bar{u}\|_1)$ *where* $(\bar{u}, \bar{\gamma}, \bar{y}, \bar{v}^1, \ldots, \bar{v}^k)$ *is any accumulation point of the sequence of iterates* $\{(u^r, \gamma^r, y^r, v^{1r}, \ldots, v^{kr})\}$ *generated by the MICA Algorithm 3.1. The point* $(\bar{u}, \bar{\gamma}, \bar{y}, \bar{v}^1, \ldots, \bar{v}^k)$ *has the following* *local minimum property:*

$$
\begin{aligned}
\nu e'\bar{y} + \|\bar{u}\|_1 = \min_{u,\gamma,y} \quad & \nu e'y + \|u\|_1 \\
\text{s.t. } & (\bar{v}^i)'K(B^i, H')u - \gamma + y_i \geq 1,\ i = 1,\ldots,k, \\
& -K(C^i_\ell, H')u + \gamma + y^i_\ell \geq 1,\ i = k+1,\ldots,m,\ \ell = 1,\ldots,m^i, \\
& y \geq 0.
\end{aligned} \tag{8}
$$

Proof That the sequence $\{\nu e'y^r + \|u^r\|_1\}$ converges follows from the fact that it is nonincreasing and bounded below by zero. That (8) is satisfied follows from the fact each point of the sequence $\{(u^r, \gamma^r, y^r, v^{1r}, \ldots, v^{kr})\}$ satisfies (8) with $(\bar{u}, \bar{y}, \bar{v}^1, \ldots, \bar{v}^k)$ replaced by $(u^r, y^r, v^{1r}, \ldots, v^{kr})$ on account of step (i) of the MICA Algorithm 3.1. Hence, any accumulation point $(\bar{u}, \bar{\gamma}, \bar{y}, \bar{v}^1, \ldots, \bar{v}^k)$ of $\{(u^r, \gamma^r, y^r, v^{1r}, \ldots, v^{kr})\}$ satisfies (8). $\diamondsuit$

In practice the MICA Algorithm 3.1 terminates very quickly, typically before ten iterations.

A similar algorithm and proposition can be given for a linear classifier which is a special case of the above nonlinear kernel classifier.

Before we turn to our numerical results, it is important to point out some significant differences between our MICA Algorithm 3.1 and the mi-SVM and MI-SVM mixed integer programming formulations introduced by Andrews et al. in [1]. A key difference is that MICA employs the 1-norm, rather than the 2-norm used by mi-SVM and MI-SVM. The 1-norm SVM formulation is known to lead to sparse solutions [3,24], which corresponds to using few input features when a linear classifier is used. Our experimental results will demonstrate this behavior. The 1-norm also allows MICA to be solved by a succession of linear programs, rather than more complex quadratic programs. Also, MICA uses an arbitrary convex combination of points in the positive bags to represent each such bag. This representation is done by means of a continuous nonnegative variable v^i for each positive bag. This is fundamentally different from both mi-SVM which uses integer variables to assign labels to each point in each positive bag, and from MI-SVM which chooses a single "witness" point to represent each positive bag. Furthermore, while MI-SVM chooses the witness point to be the point furthest from the decision boundary, MICA does not necessarily choose a convex combination furthest from the decision boundary. Finally, MICA and mi-SVM use one slack variable per negative instance, while MI-SVM uses one slack variable per negative bag. Our formulation leads to a simple algorithm without heuristics which we have shown converges to a local solution.

4 Numerical Testing

To demonstrate the capabilities of our formulation, we report results on twelve datasets, two from the UCI machine learning repository [17], and ten from [1]. Detailed information about these datasets is summarized in Table 1. We use the datasets from [1] to evaluate our linear classification algorithm. Three of these datasets are from an image annotation task in which the goal is to determine whether or not a given animal is present in an image. The other seven datasets are from the OHSUMED data and the task is to learn binary concepts associated with the Medical Subject Headings of MEDLINE documents. The two datasets from the UCI repository [17] are the Musk datasets, which

Table 1
Description of the datasets used in the experiments. Elephant, Fox, Tiger, and the TST datasets are used in [1], while Musk-1 and Musk-2 are available from [17]. + Bags denotes the number of positive bags in each dataset, while + Instances denotes the total number of instances in all the positive bags. Similarly, - Bags and - Instances denote corresponding quantities for the negative bags.

Data Set	+ Bags	+ Instances	- Bags	- Instances	Features
Elephant	100	762	100	629	143
Fox	100	647	100	673	143
Tiger	100	544	100	676	143
TST1	200	1580	200	1644	6668
TST2	200	1715	200	1629	6842
TST3	200	1626	200	1620	6568
TST4	200	1754	200	1637	6626
TST7	200	1746	200	1621	7037
TST9	200	1684	200	1616	6982
TST10	200	1818	200	1635	7073
Musk-1	47	207	45	269	166
Musk-2	39	1017	63	5581	166

are commonly used in multiple instance classification. We report results on these datasets for our nonlinear classification algorithm.

We compare our linear classification algorithm to the linear versions of mi-SVM and MI-SVM [1]. Both mi-SVM and MI-SVM use mixed-integer programming to learn a linear classifier, and as such are natural candidates for comparison with MICA. Since Andrews et al. also report results on Zhang and Goldman's expectation-maximization approach EM-DD [23] on these datasets [1], we include those results here as well. Finally, Ray and Craven [19] demonstrate that in some domains, algorithms that make no use of multiple instance information may outperform their multiple instance counterparts. Thus, we include a linear programming formulation of an SVM (SVM1) [3,14] in our comparisons as the natural non-multiple-instance counterpart to MICA.

Table 2 reports results comparing MICA to MI-SVM, MI-SVM, EM-DD, and SVM1. Correctness results for mi-SVM, MI-SVM and EM-DD were taken from [1]. Correctness for each algorithm was measured by averaging ten ten-fold cross validation runs. The regularization parameters for MICA and SVM1 were selected from the set $\{2^i | i = -7, \ldots, 7\}$ by ten-fold cross validation on each training fold for the image annotation datasets, and by using a random ten percent of each training fold as a tuning set. The final classifier for each fold was trained using all the data in the training fold. MICA was stopped if the difference between the v variables was less than 10^{-4} or if $r > 80$. SVM1 was trained by assuming all instances in each positive bag had a positive label, but for tuning and testing the classification rule was the same as for MICA. We note that MICA had the best correctness on four datasets, while mi-SVM had the best correctness on three datasets. SVM1 had the best correctness on two datasets, and MI-SVM had the best correctness on one dataset. EM-DD did not have the best correctness on any dataset.

In order to evaluate the difference between the algorithms more precisely, we used the Friedman test [8] on the results reported in Table 2. The Friedman test is a nonparametric test that compares the average ranks of the algorithms, where the algorithm with the highest correctness on a dataset is given a rank of 1 on that dataset, and the algorithm with the worst correctness is given a rank of 5. For example, on the Elephant dataset, mi-SVM has rank 1, MICA has rank 3, and EM-DD has rank 5. The average rank for MICA was 2.1, for mi-SVM 2.3, for SVM1 2.9, for MI-SVM 3.1, and for EM-DD 4.6. The

Table 2

Linear kernel MICA, mi-SVM [1], MI-SVM [1], EM-DD [23], and SVM1 testing correctness and number of features used averaged over ten ten-fold cross validation experiments. The datasets are those used by Andrews et al. in [1]. The number of features used is available on all datasets for MICA and SVM1, and on the Elephant, Fox, and Tiger datasets for mi-SVM and MI-SVM. Best correctness on each dataset is in bold. Note the substantial reduction in features by MICA and SVM1.

Data Set #Features	MICA % Correct # Features	mi-SVM % Correct # Features	MI-SVM % Correct # Features	EM-DD % Correct	SVM1 % Correct # Features
Elephant 143	80.5% 59.0	**82.2%** 143.0	81.4% 143.0	78.3%	78.5% 19.2
Fox 143	**58.7%** 78.0	58.2% 143.0	57.8% 143.0	56.1%	56.7% 73.6
Tiger 143	82.6% 50.8	78.4% 143.0	**84.0%** 143.0	72.1%	77.3% 46.0
TST1 6668	**94.5%** 50.5	93.6%	93.9%	85.8%	94.2% 41.1
TST2 6842	**85.0%** 97.2	78.2%	84.5%	84.0%	77.6% 68.4
TST3 6568	86.0% 123.7	87.0%	82.2%	69.0%	**87.3%** 39.4
TST4 6626	**87.7%** 59.9	82.8%	82.4%	80.5%	81.0% 69.0
TST7 7037	78.9% 145.3	**81.3%**	78.0%	75.4%	79.2% 37.3
TST9 6982	61.4% 302.1	**67.5%**	60.2%	65.5%	65.8% 119.6
TST10 7073	82.3% 132.2	79.6%	79.5%	78.5%	**83.6%** 37.6

Friedman test indicated that these results were significantly different at the five percent level, so we went on to perform a Bonferroni-Dunn post-hoc test [7] to compare MICA with the other algorithms. We found that the only algorithm with statistically significant difference from MICA was EM-DD. Demšar [5] gives an introduction to these tests, and demonstrates their applicability to the comparison of machine learning algorithms. These results indicate that MICA, which is the only algorithm that incorporates *both* multiple instance information and substantial feature reduction, has correctness comparable to the other linear classifiers and better than EM-DD on these datasets.

Table 2 also reports the number of features used by MICA and SVM1 on all datasets, and by the mixed-integer programming formulations on the image annotation datasets (Elephant, Fox, and Tiger). The feature selection results for mi-SVM and MI-SVM were computed by running a single ten-fold cross validation experiment on each dataset. The regularization parameter was chosen from the set $\{2^i | i = -7, \ldots, 7\}$ by a tuning set consisting of a random ten percent of each training fold, and the final classifier was trained on all the data in the training fold for each fold. Our implementations achieved correctness within a few percentage points of the reported accuracies in [1]. We do not believe that changing the tuning procedure or running more experiments would change the number of features used, since our results are in line with previous comparisons between 1-norm and 2-norm penalties for SVMs [3,24]. We note that although there were 230 features reported in the three image annotation datasets, 87 are zero in every instance for each dataset. Thus, neither MI-SVM nor mi-SVM reduce features on these datasets. While we did not test mi-SVM and MI-SVM on the OHSUMED data, the above results and cited work indicate that they will not reduce features as drastically as MICA

Table 3
Nonlinear kernel MICA, mi-SVM [1], MI-SVM [1], EM-DD [23], DD [15] MI-NN [18], IAPR [6], and MIK [9] ten-fold testing correctness on the Musk-1 and Musk-2 datasets. Best accuracy is in bold.

Data Set	MICA	mi-SVM	MI-SVM	EM-DD	DD	MI-NN	IAPR	MIK
Musk-1	84.4%	87.4%	77.9%	84.8%	88.0%	88.9%	**92.4%**	91.6%
Musk-2	**90.5%**	83.6%	84.3%	84.9%	84.0%	82.5%	89.2%	88.0%

or SVM1. We note that in some cases MICA used less than one percent of the features, and never used more than five percent on the OHSUMED data or more than 55% of the 143 non-zero features on the image annotation data.

To demonstrate the efficiency of our proposed formulation, we describe its behavior on the Elephant dataset. The average time to learn a classifier once the parameter ν was chosen was 25.2 seconds, and the average number of MICA iterations required was 5.8. Note that each iteration involves solving two linear programs. These results were obtained on a Pentium III 650 MHz desktop machine with 256MB RAM running Tao Linux, Version 1. The algorithm was implemented in MATLAB [16] and the linear programs were solved using the dual simplex method of the CPLEX linear programming solver [10].

Although our numerical testing is focused on linear classification, Table 3 gives ten-fold cross validation accuracy results for MICA using a Gaussian kernel and previously published results of several other algorithms on the Musk-1 and Musk-2 datasets which are available from the UCI repository [17]. The results for EM-DD are taken from [1]. The MIK entry reports the best ten-fold cross validation result among the multi-instance kernel methods of Gartner et al. [9]. The IAPR entry reports the results obtained by Dietterich et al. using the Iterated Discrimination Axis-Parallel Rectangle algorithm [6]. The parameters ν and μ of MICA were both chosen from the set $\{2^i | i = -7, \ldots, 7\}$ using a subset of the training set as a tuning set for each fold. MICA was stopped if the difference between the v variables was less than 10^{-4} or if $r > 80$. To speed computation and reduce the risk of overfitting, a reduced kernel containing ten percent of the rows of H was used as in [11]. We note that MICA had the best correctness among eight methods on Musk-2.

5 Conclusion & Outlook

We have introduced a mathematical programming formulation of the multiple-instance problem that has a linear objective with linear and bilinear constraints. Our mathematical program can be efficiently solved by a succession of fast linear programs that converge in a few iterations to a local solution. Results on previously published datasets indicate that our approach is very effective at finding substantially sparse linear classifiers. Furthermore, our approach can be easily extended to finding nonlinear classifiers with potentially high accuracy through the use of nonlinear kernels. Improvements in the mathematical programming formulation and evaluation using a wide variety of datasets and algorithms, such as those in [19], are promising avenues of future research.

Acknowledgments

The research described in this Data Mining Institute Report 05-02, May 2005, revised May 2006, was supported by National Science Foundation Grants CCR-0138308 and IIS-0511905, the Microsoft Corporation and ExxonMobil. We are grateful to our colleague Soumya Ray for access to some of the datasets and to the anonymous reviewers for their constructive comments.

References

[1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 561–568. MIT Press, Cambridge, MA, October 2003.

[2] P. Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Proceedings 14th International Conference on Machine Learning*, pages 21–29. Morgan Kaufmann, 1997.

[3] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.

[5] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[6] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1998.

[7] O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

[8] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[9] Thomas Gartner, Peter A. Flach, Adam Kowalczyk, and Alex J. Smola. Multi-instance kernels. In Claude Sammut and Achim Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, July 2002.

[10] ILOG, Incline Village, Nevada. *ILOG CPLEX 9.0 User's Manual*, 2003. http://www.ilog.com/products/cplex/.

[11] Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the First SIAM International Conference on Data Mining, Chicago, April 5-7, 2001, CD-ROM.* ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-07.ps.

[12] P. M. Long and L. Tan. PAC learning axis aligned rectangles with respect to product distributions from multiple instance examples. *Machine Learning*, 30(1):7–22, 1998.

[13] O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24:15–23, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-07r.ps.

[14] O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

[15] O. Maron and A. L. Ratan. Multiple-instance learning for natural scene classification. In *15th International Conference on Machine Learning*, San Francisco, CA, 1998. Morgan Kaufmann.

[16] MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2001. http://www.mathworks.com.

[17] P. M. Murphy and D. W. Aha. UCI machine learning repository, 1992. www.ics.uci.edu/∼mlearn/MLRepository.html.

[18] J. Ramon and L. De Raedt. Multi-instance neural networks. In *Proceedings of ICML-2000, Workshop on Attribute-Value and Relational Learning*, 2000.

[19] S. Ray and M. Craven. Supervised versus multiple instance learnining: An emperical comparison. In *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

[20] B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[21] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, second edition, 2000.

[22] J. Wang and J.-D. Zucker. Solving the multiple instance problem: A lazy learning approach. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000), Stanford University*, pages 1119–1125, San Francisco, CA, 2000. Morgan Kaufman Publishers.

[23] Q. Zhang and S. A. Goldman. EM-DD: an improved miltiple-instance learning technique. In *Neural Information Processing Systems 2001*, pages 1073–1080, Cambridge, MA, 2002. MIT Press.

[24] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-Norm support vector machines. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16–NIPS2003.* MIT Press, 2004.