Knowledge-Based Support Vector Machine Classifiers

Glenn Fung, Olvi L. Mangasarian and Jude Shavlik
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
{gfung,olvi,shavlik}@cs.wisc.edu

ABSTRACT

Prior knowledge in the form of multiple polyhedral sets, each belonging to one of two categories, is introduced into a reformulation of a linear support vector machine classifier. The resulting formulation leads to a linear program that can be solved efficiently. Real world examples, from DNA sequencing and breast cancer prognosis, demonstrate the effectiveness of the proposed method. Numerical results show improvement in test set accuracy after the incorporation of prior knowledge into ordinary data-based linear support vector machine classifiers. One experiment also shows that a linear classifier, based solely on prior knowledge, far outperforms the direct application of the prior knowledge rules to classify new examples.

Keywords ***knowledge sets, support vector machine classifier, knowledge-based classifier, linear programming***

1. INTRODUCTION

Support vector machines (SVMs) have played a major role in classification problems [24, 3, 12, 1, 13, 5, 6]. However unlike other classification tools such as knowledge-based neural networks [21, 22, 17, 7], little work [20] has gone into incorporating prior knowledge into support vector machines. In this work we present a novel approach to incorporating prior knowledge in the form of polyhedral *knowledge sets* in the input space of the given data. These knowledge sets, which can be as simple as cubes, are supposed to belong to one of two categories into which all the data is divided. Thus, a single knowledge set can be interpreted as a generalization of a training example, which typically consists of a single *point* in input space. In contrast, each of our knowledge sets consists of a *region* in the same space. By using a powerful tool from mathematical programming, theorems of the alternative [10, Chapter 2], we are able to embed such prior data into a linear program that can be efficiently solved by any of the publicly available solvers.

We briefly summarize the contents of the paper now. In Section 2 we describe the linear support vector machine classifier and give a linear programming for it. We then describe how prior knowledge, in the form of polyhedral knowledge sets belonging to one of two classes can be characterized. In Section 3 we incorporate these polyhedral sets into our linear programming formulation which results in our knowledge-based support vector machine (KSVM) formulation (19). This formulation is capable of generating a linear classifier based on real data and/or prior knowledge. Section 4 contains numerical test results comparing various linear and nonlinear classifiers with and without the incorporation of prior knowledge. 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 a vector x in R^n, the sign function $sign(x)$ is defined as $sign(x)_i = 1$ if $x_i > 0$ else $sign(x)_i = -1$ if $x_i \leq 0$, for $i = 1, \dots, n$. 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 and A_i will denote the i-th row 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 *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. The symbol $\wedge$ will denote the logical "and". For simplicity, the dimensionality of some vectors will not be explicitly given. The abbreviation "s.t." stands for "such that".

2. THE LINEAR SUPPORT VECTOR MACHINE AND PRIOR KNOWLEDGE

We consider the problem, depicted in Figure 1, of classifying m points in the n-dimensional input space R^n, represented by the $m \times n$ matrix A, according to membership of each point A_i in the class $A+$ or $A-$ as specified by a given $m \times m$ diagonal matrix D with plus ones or minus ones along its diagonal. For this problem, the linear programming support vector machine [12, 2] with a linear kernel, which is a variant of the standard support vector machine [24, 3], is given by the following linear program with parameter $\nu > 0$:

$$\begin{array}{rrcl} \min\limits_{(w,\gamma,y)\in R^{n+1+m}} & \nu e'y + \|w\|_1 & & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\ & y & \geq & 0, \end{array} \tag{1}$$

where $\|\cdot\|_1$ denotes the 1-norm as defined in the Introduction. That this problem is indeed a linear program, can be easily seen from the equivalent formulation:

$$\begin{array}{rrcl} \min\limits_{(w,\gamma,y,t)\in R^{n+1+m}} & \nu e'y + e't & & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\ & t \geq w & \geq & -t \\ & y & \geq & 0. \end{array} \tag{2}$$

For economy of notation we shall use the first formulation (1) with the understanding that computational implementation is via (2). As depicted in Figure 1, w is the normal to the bounding planes:

$$\begin{array}{rcl} x'w & = & \gamma + 1 \\ x'w & = & \gamma - 1, \end{array} \tag{3}$$

that bound most of the points belonging to the sets $A+$ and $A-$ respectively. The constant γ determines their location relative to the origin. When the two classes are strictly linearly separable, that is when the error variable $y = 0$ in (1) (which is not the case shown in Figure 1), the plane $x'w = \gamma + 1$ bounds all of the class $A+$ points, while the plane $x'w = \gamma - 1$ bounds all of the class $A-$ points as follows:

$$\begin{array}{rcll} A_iw & \geq & \gamma + 1, & \text{for} \quad D_{ii} = 1, \\ A_iw & \leq & \gamma - 1, & \text{for} \quad D_{ii} = -1. \end{array} \tag{4}$$

Consequently, the plane:

$$x'w = \gamma, \tag{5}$$

midway between the bounding planes (3), is a separating plane that separates points belonging to $A+$ from those belonging to $A-$ completely if $y = 0$, else only approximately as depicted in Figure 1. The 1-norm term $\|w\|_1$ in (1), which is half the reciprocal of the distance $\frac{2}{\|w\|_1}$ measured using the ∞-norm distance [11] between the two bounding planes of (3) (see Figure 1), maximizes this distance, often called the "margin". Maximizing the margin enhances the generalization capability of a support vector machine [24, 3]. If the classes are linearly inseparable, which is the case shown in Figure 1, then the two planes bound the two classes with a "soft margin" (i.e. bound approximately with some error) determined by the nonnegative error variable y, that is:

$$\begin{array}{rcccll} A_iw & + & y_i & \geq & \gamma + 1, & \text{for } D_{ii} = 1, \\ A_iw & - & y_i & \leq & \gamma - 1, & \text{for } D_{ii} = -1. \end{array} \tag{6}$$

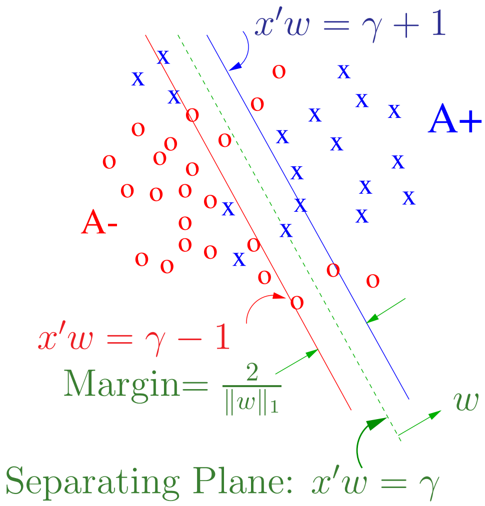

Figure 1: The linear programming support vector machine classifier in the w-space of R^n: The approximately bounding planes of equation (3) with a soft (i.e. with some error) margin $\frac{2}{\|w\|_1}$, and the plane of equation (5) approximately separating points in $A+$ from points in $A-$.

The 1-norm of the error variable y is minimized parametrically with weight ν in (1), resulting in an approximate separating plane (5) as depicted in Figure 1. This plane acts as a linear classifier as follows:

$$sign(x'w - \gamma\,) \begin{cases} = 1, & \text{then } x \in A+, \\ = -1, & \text{then } x \in A-, \end{cases} \tag{7}$$

where $sign(\cdot)$ is the sign function defined in the Introduction.

Suppose now that we have prior information of the following type. All points x lying in the polyhedral set determined by the linear inequalities:

$$Bx \leq b, \tag{8}$$

belong to class $A+$. Such inequalities generalize simple box constraints such as $a \leq x \leq d$. Looking at Figure 1 or at the inequalities (4) we conclude that the following implication must hold:

$$Bx \leq b \implies x'w \geq \gamma + 1. \tag{9}$$

That is, the *knowledge set* $\{x \mid Bx \leq b\}$ lies on the $A+$ side of the bounding plane $x'w = \gamma + 1$. This implication is equivalent to the following statement for a given (w, γ):

$$Bx \leq b, \;\; x'w < \gamma + 1, \;\; \text{has no solution } x. \tag{10}$$

This statement in turn is *implied* by the following statement:

$$B'u + w = 0, \;\; b'u + \gamma + 1 \leq 0, \;\; u \geq 0, \;\; \text{has a solution } (u, w). \tag{11}$$

To see this simple backward implication: (10)$\Longleftarrow$(11), we suppose the contrary that there exists an x satisfying (10) and obtain the contradiction $b'u > b'u$ as follows:

$$b'u \geq u'Bx = -w'x > -\gamma - 1 \geq b'u, \tag{12}$$

where the first inequality follows by premultiplying $Bx \leq b$ by $u \geq 0$. In fact, under the natural assumption that the prior knowledge set $\{x \mid Bx \leq b\}$ is nonempty, the forward implication: (10)$\Longrightarrow$(11) is also true, as a direct consequence of the nonhomogeneous Farkas theorem of the alternative [10, Theorem 2.4.8]. We state this equivalence as the following key proposition to our knowledge-based approach.

Proposition 2.1. **Knowledge Set Classification.** *Let the set $\{x \mid Bx \leq b\}$ be nonempty. Then for a given (w, γ), the implication (9) is equivalent to the statement (11). In other words, the set $\{x \mid Bx \leq b\}$ lies in the halfspace $\{x \mid w'x \geq \gamma + 1\}$ if and only if there exists u such that $B'u + w = 0$, $b'u + \gamma + 1 \leq 0$ and $u \geq 0$.*

Proof We establish the equivalence of (9) and (11) by showing the equivalence (10) and (11). By the nonhomogeneous Farkas theorem [10, Theorem 2.4.8] we have that (10) is equivalent to either:

$$B'u + w = 0,\ b'u + \gamma + 1 \leq 0,\ u \geq 0,\ \text{having solution } (u, w), \tag{13}$$

or

$$B'u = 0,\ b'u < 0,\ u \geq 0,\ \text{having solution } u. \tag{14}$$

However, the second alternative (14) contradicts the nonemptiness of the knowledge-set $\{x \mid Bx \leq b\}$, because for $x \in \{x \mid Bx \leq b\}$ and u solving (14) we obtain the contradiction:

$$0 \geq u'(Bx - b) = x'B'u - b'u = -b'u > 0. \tag{15}$$

Hence (14) is ruled out and we have that (10) is equivalent to (13) which is (11). □

This proposition will play a key role in incorporating knowledge sets, such as $\{x \mid Bx \leq b\}$, into one of two categories in a support vector classifier formulation as demonstrated in the next section.

3. KNOWLEDGE-BASED SVM CLASSIFICATION

We describe now how to incorporate prior knowledge in the form of polyhedral sets into our linear programming SVM classifier formulation (1).

We assume that we are given the following *knowledge sets*:

$$\begin{array}{l} k \text{ sets belonging to } A+ : \{x \mid B^i x \leq b^i\},\ i = 1, \ldots, k \\ \ell \text{ sets belonging to } A- : \{x \mid C^i x \leq c^i\},\ i = 1, \ldots, \ell \end{array} \tag{16}$$

By Proposition 2.1 this knowledge is equivalent to the following requirements with respect to the bounding planes (3):

$$\begin{array}{l} \text{There exist } u^i,\ i = 1, \ldots, k,\ v^j,\ j = 1, \ldots, \ell, \text{ such that:} \\ B^{i\prime} u^i + w = 0,\ b^{i\prime} u^i + \gamma + 1 \leq 0,\ u^i \geq 0,\ i = 1, \ldots, k \\ C^{j\prime} v^j - w = 0,\ c^{j\prime} v^j - \gamma + 1 \leq 0,\ v^j \geq 0,\ j = 1, \ldots, \ell \end{array} \tag{17}$$

All we need to do now in order to incorporate the knowledge sets (16) into the linear programming formulation (1) of an SVM classifier, is to add the conditions (17) as constraints to (1) as follows:

$$\begin{array}{rrcl} \displaystyle\min_{w,\gamma,y,u^i,v^j} & \nu e'y + \|w\|_1 & & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\ & y & \geq & 0 \\ & B^{i\prime} u^i + w & = & 0 \\ & b^{i\prime} u^i + \gamma + 1 & \leq & 0 \\ & u^i & \geq & 0,\ i = 1, \ldots, k \\ & C^{j\prime} v^j - w & = & 0 \\ & c^{j\prime} v^j - \gamma + 1 & \leq & 0 \\ & v^j & \geq & 0,\ j = 1, \ldots, \ell \end{array} \tag{18}$$

This linear programming formulation will ensure that each of the knowledge sets $\{x \mid B^i x \leq b^i\}$, $i = 1, \ldots, k$ and $\{x \mid C^i x \leq c^i\}$, $i = 1, \ldots, \ell$ lie on the appropriate side of the bounding planes (3). However, there is no guarantee that such bounding planes exist that will precisely separate these two classes of knowledge sets, just as there is no *a priori* guarantee that the original points belonging to the sets $A+$ and $A-$ are linearly separable. We therefore add error variables $r^i, \rho^i, i = 1, \ldots, k, s^j, \sigma^j, j = 1, \ldots, \ell$, just like the error variable y of the SVM formulation (1), and attempt to drive these error variables to zero by modifying our last formulation above as follows:

$$\begin{array}{rrcl} \displaystyle\min_{w,\gamma,y,u^i,r^i,\rho^i,v^j,s^j,\sigma^j} & \nu e'y + & & \\ & \mu\left(\displaystyle\sum_{i=1}^{k}(r^i + \rho^i)\right. & + & \left.\displaystyle\sum_{j=1}^{\ell}(s^j + \sigma^j)\right) + \|w\|_1 \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq & e \\ & y & \geq & 0 \\ & -r^i \leq B^{i\prime} u^i + w & \leq & r^i \\ & b^{i\prime} u^i + \gamma + 1 & \leq & \rho^i \\ & u^i, r^i, \rho^i & \geq & 0,\ i = 1, \ldots, k \\ & -s^j \leq C^{j\prime} v^j - w & \leq & s^j \\ & c^{j\prime} v^j - \gamma + 1 & \leq & \sigma^j \\ & v^j, s^j, \sigma^j & \geq & 0,\ j = 1, \ldots, \ell \end{array} \tag{19}$$

This is our final knowledge-based linear programming formulation which incorporates the knowledge sets (16) into the linear classifier with weight μ, while the (empirical) error term $e'y$ is given weight ν. As usual, the value of these two parameters, ν, μ, are chosen by means of a tuning set extracted from the training set. If we set $\mu = 0$ then the linear program (19) degenerates to (1), the linear program associated with an ordinary linear SVM. However, if set $\nu = 0$, then the linear program (19) generates a linear SVM that is strictly based on knowledge sets, but not on any specific training data. This might be a useful paradigm for situations where training datasets are not easily available, but expert knowledge, such as doctors' experience in diagnosing certain diseases, is readily available. This will be demonstrated in the breast cancer dataset of Section 4.

Note that the 1-norm term $\|w\|_1$ can be replaced by one half the 2-norm squared, $\frac{1}{2}\|w\|_2^2$, which is the usual margin maximization term for ordinary support vector machine classifiers [24, 3]. However, this changes the linear program (19) to a quadratic program which typically takes longer time to solve.

For completeness we state our knowledge-based algorithm as follows.

Algorithm 3.1. **Knowledge-Based Linear SVM (KSVM).** *Given m data points in R^n represented by the $m \times n$ matrix A and a diagonal matrix D of ± 1 labels denoting the class of each row of A, and given the knowledge sets (16), we generate the linear classifier (7) as follows:*

(i) Compute $\begin{bmatrix} w \\ \gamma \end{bmatrix}$ by solving the linear program (19) for some positive values of ν and μ. Typically ν and μ are chosen by means of a tuning (validating) set.

(ii) Classify a new x by using (7) and the solution $\begin{bmatrix} w \\ \gamma \end{bmatrix}$ from Step (i) above.

For standard SVMs, support vectors consist of all data points which are the complement of the data points that can be dropped from the problem without changing the separating plane (5) [24, 12]. Thus for our knowledge-based linear programming formulation (19), support vectors correspond to data points (rows of the matrix A) for which the Lagrange multipliers are nonzero, because solving (19) with these data points only will give the same answer as solving (19) with the entire matrix A.

The concept of support vectors has to be modified as follows for our knowledge sets. Since each knowledge set in (16) is represented by a matrix B^i or C^j, each row of these matrices can be thought of as characterizing a plane boundary of the knowledge set. In our formulation (19) above, such rows are wiped out if the corresponding component of the variables u^i or v^j are zero at an optimal solution. We call the complement of these components of the the knowledge sets (16) *support constraints*. Deleting constraints (rows of B^i or C^j), for which the corresponding components of u^i or v^j are zero, will not alter the solution of the knowledge-based linear program (19). This in fact is corroborated by numerical tests that were carried out.

We demonstrate the geometry of incorporating knowledge sets into a classification problem by considering a synthetic example in R^2 with $m = 200$ points, 100 of which are in $A+$ and the other 100 in $A-$. Figure 2 depicts ordinary linear separation using the linear SVM formulation (1). We now incorporate three knowledge sets into the the problem: $\{x \mid B^1x \le b^1\}$ belonging to $A+$ and $\{x \mid C^1x \le c^1\}$ and $\{x \mid C^2x \le c^2\}$ belonging to $A-$, and solve our linear program (19). We depict the new linear separation in Figure 3 and note the substantial change generated in the linear separation by the incorporation of these three knowledge sets.

4. NUMERICAL TESTING

All our computations were performed on the University of Wisconsin Data Mining Institute "locop1" machine, which utilizes a 400 Mhz Pentium II and allows a maximum of 2 Gigabytes of memory for each process. This computer runs on Windows NT server 4.0, with MATLAB 6 installed.

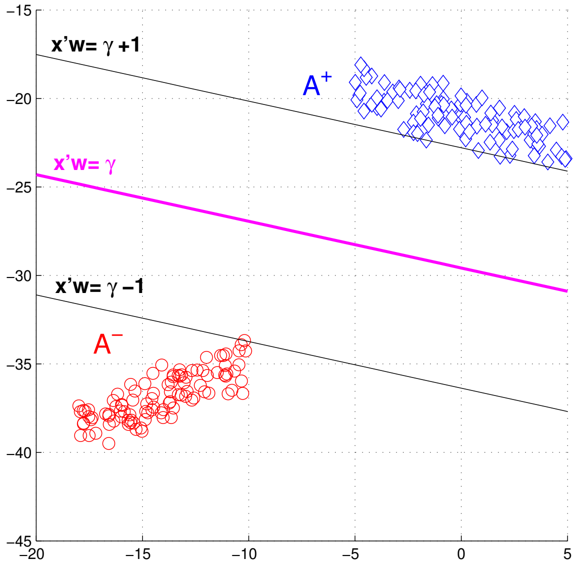

Figure 2: A linear SVM separation for 200 points in R^2 using the linear programming formulation (1).

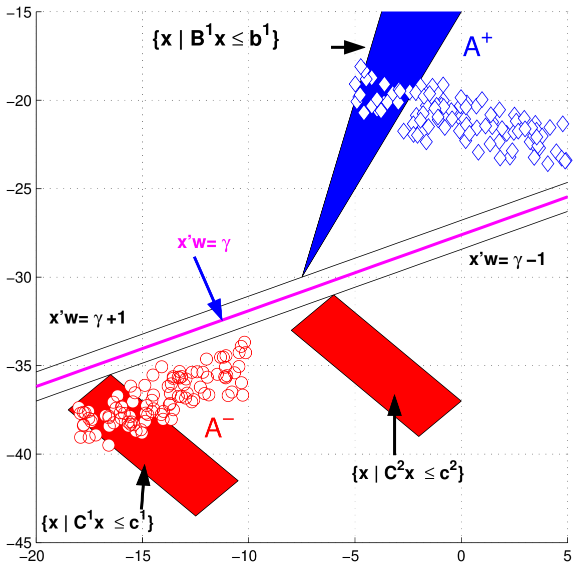

Figure 3: A linear SVM separation for the same 200 points in R^2 as those in Figure 2 but using the linear programming formulation (19) which incorporates three knowledge sets: $\{x \mid B^1x \le b^1\}$ into the halfspace of $A+$, and $\{x \mid C^1x \le c^1\}$, $\{x \mid C^2x \le c^2\}$ into the halfspace of $A-$, as depicted in the figure above. Note the substantial difference between the linear classifiers $x'w = \gamma$ of Figures 2 and 3.

Even though "locop1" is a multiprocessor machine, only one processor was used for all the experiments since MATLAB is a single threaded application and does not distribute any load across processors [14].

Our numerical testing was performed on two publicly available datasets [15]. A description of the datasets and the experiments performed is presented next.

4.1 Promoter Recognition Dataset

The first dataset [23] is from the domain of DNA sequence analysis and is called the promoter recognition dataset. A promoter is a short DNA sequence that precedes a gene sequence. Hence, it is important distinguish between *promoter* and *nonpromoter* sequences, since this will make it possible to identify the starting locations of genes in long, uncharacterized sequences of DNA.

4.1.1 Description of the data

For classification purposes of this experiment it is sufficient to consider the DNA as a linear sequence of characters belonging to a four elements set $\{A, G, C, T\}$. These four elements are commonly referred to nucleotides.

Each promoter consists of a sequence of 57 consecutive DNA nucleotides. The locations are numbered with respect to a fixed meaningful, reference point. Starting at position -50 (p_{-50}) and ending at position +7 (p_7). Negative numbers indicate sites preceding the reference point while positive numbers indicate sites following the reference point. Since each one of the 57 feature takes nominal values from the set $\{A, G, C, T\}$, a real valued representation is needed to apply our KSVM formulation. Each nominal value is mapped into a four dimensional binary vector depending on the nucleotide that is being represented. This simple and widely used "1 of N" mapping scheme for converting nominal attributes into real-valued attributes is illustrated in Figure 4.

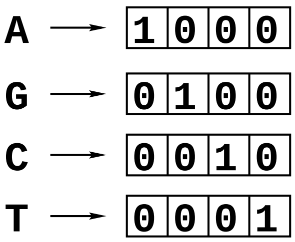

Figure 4: Real-valued representation of the nucleotide set $\{A, G, C, T\}$.

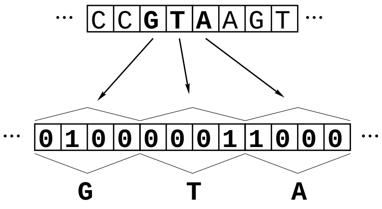

Figure 5: Real-valued representation of a promoter, from a 57 nominal-valued vector to a $57 \times 4 = 228$ real-valued vector.

Once this simple conversion is applied to the dataset, the feature space is transformed from a 57-dimensional space with nominal values into a $57 \times 4 = 228$ real-valued dimensional space. So, each data point is now a point in R^{228}. See Figure 5.

The training example consist of 53 sample promoters and 53 nonpromoter sequences for a total of 106 data points.

4.1.2 The domain theory

The domain theory for the promoters dataset is also publicly available [15] using a prolog notation. However in order to make these rules accessible to a wider audience we will use standard logic notation to describe them below.

Each $PROMOTER$ must have a region where a certain protein (RNA polymerase) must make $CONTACT$ and the helical DNA sequence must have a valid conformation ($CONF$) so the two pieces of the contact region spatially align. This rule can be written as:

$$CONTACT \wedge CONF \Longrightarrow PROMOTER \qquad (20)$$

There are two regions named $M35$ and $M10$ in the original representation of the sequences in which the RNA polymerase makes $CONTACT$, this means:

$$M35 \wedge M10 \Longrightarrow CONTACT \qquad (21)$$

The following 4 rules R_1, R_2, R_3, R_4 imply possible contact in the region $M35$:

$$R_1: \begin{array}{l} (p_{-37} = C) \wedge (p_{-36} = T) \wedge (p_{-35} = T) \wedge \\ (p_{-34} = G) \wedge (p_{-33} = A) \wedge (p_{-32} = C) \end{array} \Longrightarrow M35$$

$$R_2: \begin{array}{l} (p_{-36} = T) \wedge (p_{-35} = T) \wedge (p_{-34} = G) \wedge \\ (p_{-32} = C) \wedge (p_{-31} = A) \end{array} \Longrightarrow M35$$

$$R_3: \begin{array}{l} (p_{-36} = T) \wedge (p_{-35} = T) \wedge (p_{-34} = G) \wedge \\ (p_{-33} = A) \wedge (p_{-32} = C) \wedge (p_{-31} = A) \end{array} \Longrightarrow M35$$

$$R_4: \begin{array}{l}(p_{-36}=T)\wedge(p_{-35}=T)\wedge(p_{-34}=G)\wedge\\(p_{-33}=A)\wedge(p_{-32}=C)\end{array} \implies M35$$

The following 4 rules R_5, R_6, R_7, R_8 imply possible contact in the region $M10$:

$$R_5: \begin{array}{l}(p_{-14}=T)\wedge(p_{-13}=A)\wedge(p_{-12}=T)\wedge\\(p_{-11}=A)\wedge(p_{-10}=A)\wedge(p_{-09}=T)\end{array} \implies M10$$

$$R_6: \begin{array}{l}(p_{-13}=T)\wedge(p_{-12}=A)\wedge(p_{-10}=A)\wedge\\(p_{-08}=T)\end{array} \implies M10$$

$$R_7: \begin{array}{l}(p_{-13}=T)\wedge(p_{-12}=A)\wedge(p_{-11}=T)\wedge\\(p_{-10}=A)\wedge(p_{-09}=A)\wedge(p_{-08}=T)\end{array} \implies M10$$

$$R_8: (p_{-12}=T)\wedge(p_{-11}=A)\wedge(p_{-07}=T) \implies M10$$

The following 4 rules $R_9, R_{10}, R_{11}, R_{12}$ imply characteristics that produce acceptable conformations:

$$R_9: \begin{array}{l}(p_{-47}=C)\wedge(p_{-46}=A)\wedge(p_{-45}=A)\wedge\\(p_{-43}=T)\wedge(p_{-42}=T)\wedge(p_{-40}=A)\wedge\\(p_{-39}=C)\wedge(p_{-22}=G)\wedge(p_{-18}=T)\wedge\\(p_{-16}=C)\wedge(p_{-08}=G)\wedge(p_{-07}=C)\wedge\\(p_{-06}=G)\wedge(p_{-05}=C)\wedge(p_{-04}=C)\wedge\\(p_{-02}=C)\wedge(p_{-01}=C)\end{array} \implies CONF$$

$$R_{10}: (p_{-45}=A)\wedge(p_{-44}=A)\wedge(p_{-41}=A) \implies CONF$$

$$R_{11}: \begin{array}{l}(p_{-49}=A)\wedge(p_{-44}=T)\wedge(p_{-27}=T)\wedge\\(p_{-22}=T)\wedge(p_{-18}=T)\wedge(p_{-16}=A)\wedge\\(p_{-15}=C)\wedge(p_{-01}=C)\end{array} \implies CONF$$

$$R_{12}: \begin{array}{l}(p_{-45}=A)\wedge(p_{-41}=A)\wedge(p_{-28}=T)\wedge\\(p_{-27}=T)\wedge(p_{-23}=T)\wedge(p_{-21}=A)\wedge\\(p_{-20}=A)\wedge(p_{-17}=T)\wedge(p_{-15}=T)\wedge\\(p_{-04}=T)\end{array} \implies CONF$$

It is important to note that this domain theory matches none of the given examples of the training set. Hence, this set of rules is useless as a classifier by itself. However, they do capture a significant amount of information about promoters and it has been shown that incorporating them into a classifier results in a better and more accurate classifier using knowledge-based neural networks [23]. We will demonstrate a similar result using our simpler linear support vector machine linear classifier.

4.1.3 Conversion of the rules to sets of the form $Bx \leq b$

To convert the above rules $R_1, \ldots, R_{12}$ to a set of matrix inequalities we group them as follows:

$$\begin{array}{ccccccc} M35: & & M10: & & CONF: & & \\ \begin{bmatrix} R_1\\ or\\ R_2\\ or\\ R_3\\ or\\ R_4 \end{bmatrix} & \wedge & \begin{bmatrix} R_5\\ or\\ R_6\\ or\\ R_7\\ or\\ R_8 \end{bmatrix} & \wedge & \begin{bmatrix} R_9\\ or\\ R_{10}\\ or\\ R_{11}\\ or\\ R_{12} \end{bmatrix} & \implies & PROMOTER \\ 4 & \times & 4 & \times & 4 & = & 64 \text{ Rules.} \end{array}$$

This results in 64 rules which imply the occurrence of a promoter. Each of these 64 rules can be represented as a constraint set $\{x | B^i x \leq b^i\}$ for $i = 1, \ldots, 64$, which define part of the promoter region in R^{228}.

4.1.4 Numerical Comparisons

The performance in terms of generalization ability of our proposed algorithm, is compared to six other learning algorithms using the same promoter dataset. These algorithms are:

1. **KBANN: Knowledge Based Artificial Neural Networks**
   A hybrid learning system that maps problem specific "domain theories", represented in propositional logic into neural networks and then, refines this reformulated knowledge using back propagation [23].
2. **BP: Standard Back Propagation**
   Neural networks with a simple connected layer of hidden units [19].
3. **O'Neill's Method**
   An empirical method suggested by a biologist based in a collection of "filters" to be used for promoter recognition [16].
4. **NN: Nearest Neighbor algorithm**
   PEBLS Nearest algorithm [4] with $k = 3$.
5. **ID3: Decision Tree**
   Quinlan's decision tree builder [18].
6. **SVM$_1$: 1-norm SVM**
   Standard 1-norm support vector machine [2].

Following the methodology used in prior work [23], we tested our algorithm on this dataset using a "leave-one-out" cross validation methodology in which the entire training set of 106 elements is repeatedly divided into a training set of size 105 and a test set of size 1, in all 106 possible ways. The values of ν and μ associated with both KSVM and SVM$_1$ where obtained by a tuning procedure. The tuning procedure consists of removing a subset of the training data, solving (19) for various various values of ν and μ and choosing values of these parameters based on best accuracy on the removed "tuning" set. The number of times that a test element is misclassified for each method is counted as an error and reported in Table 1.

Table 1: Comparison of KSVM leave-one-out total error with various classification algorithms.

Method	Number of Errors (out of 106)
KBANN*	4
KSVM	5
BP*	8
SVM_1	9
O'Neill*	12
NN*	13
ID3*	19

* Results reported in [23].

Note that our proposed KSVM is second best among all tested methods with 5 misclassification while KBANN is best with 4 misclassification even though our classifier is a simple linear classifier, $sign(x'w - \gamma)$, while a neural network classifier is a more complex nonlinear classifier. Furthermore, we note that KSVM is relatively simpler to implement than KBANN and requires merely a commonly available linear programming solver. In addition, KSVM which is a linear support vector machine classifier, improves by 44.4% the error of an ordinary linear 1-norm SVM classifier that does not use knowledge sets.

4.2 Wisconsin Breast Cancer Prognosis Dataset

4.2.1 Description of the data

The second dataset used in our numerical tests was the Wisconsin breast cancer prognosis dataset. Each data point consists of 30 nuclear features plus the diameter of the excised tumor and the number of metastasized lymph nodes. There are 110 instances corresponding to 41 patients whose cancer had recurred (RECUR) in less than 60 months and 69 patients whose cancer had not recurred (NONRECUR) in less than 60 months. We tested our algorithm on this dataset using a ten-fold cross validation methodology in which the entire training set of 110 elements is divided ten times into a training set of size 99 and a test set of size 11. The testing correctness reported is an average taken over the testing correctness obtained in each fold. The parameters ν and μ associated with both KSVM and SVM_1 where obtained by a tuning procedure similar to that of Section 4.1.4.

4.2.2 The domain theory

The domain theory we used in this experiment are rules used by doctors [9] and they are related to two of the features of the dataset: tumor size (feature 31), that is the diameter of the excised tumor in centimeters and lymph node status which refers to the number of metastasized axillary lymph nodes (feature 32). The rules are:

$RECUR$ Rule:

$$(LYMPH \geq 5) \wedge (TUMOR \geq 4) \implies RECUR$$

$NONRECUR$ rule:

$$(LYMPH = 0) \wedge (TUMOR \leq 1.9) \implies NONRECUR$$

4.2.3 Numerical results

It is important to note that the rules described above can be applied directly to classify only 32 of the given 110 given points of the training dataset and correctly classify 22 of these 32 points. The remaining 78 points are not classifiable by the above rules. Hence, if the rules are applied as a classifier by themselves the classification accuracy would be 20%. As such, these rules are not very useful by themselves and doctors use them in conjunction with other rules [9]. However, using our approach the rules were converted to linear inequalities and used in our KSVM algorithm without any use of the data, i.e. $\nu = 0$ in the linear program (19). The resulting linear classifier achieved 66.4% accuracy. The correctness achieved by standard SVM using all the data is 66.2% [2]. This result is remarkable because our knowledge-based formulation can be applied to problems where training data may not be available whereas expert knowledge may be readily available in the form of knowledge sets. This fact makes this method considerably different from previous hybrid methods like KBANN where training examples are needed in order to refine prior knowledge.

5. CONCLUSION & FUTURE DIRECTIONS

We have proposed an efficient procedure for incorporating prior knowledge in the form of knowledge sets into a linear support vector machine classifier either in combination with a given dataset or based solely on the knowledge sets. This novel and promising approach of handling prior knowledge is worthy of further study, especially ways to handle and simplify the combinatorial nature of incorporating prior knowledge into linear inequalities.

A class of possible future applications might be to problems where training data may not be easily available whereas expert knowledge may be readily available in the form of knowledge sets. This would correspond to solving our knowledge based linear program (19) with $\nu = 0$. A typical example of this type was breast cancer prognosis [8, 9] where knowledge sets by themselves generated a linear classifier as good as any classifier based on data points. This is a new way of incorporating prior knowledge into powerful support vector machine classifiers. Another important avenue to pursue is that of knowledge sets characterized by convex but nonpolyhedral sets as well as nonlinear classifiers determined by nonlinear kernels [24, 12] which are capable of handling more complex classification problems.

Acknowledgments

The research described in this Data Mining Institute Report 01-09, November 2001, was supported by National Science Foundation Grants CCR-9729842, IRI-9502990 and CDA-9623632, by Air Force Office of Scientific Research Grant F49620-00-1-0085, by National Library of Medicine Grant 1 R01 LM07050-01, and by the Microsoft Corporation.

6. REFERENCES

[1] P. S. Bradley, Usama M. Fayyad, and O. L. Mangasarian. Data mining: Overview and

optimization opportunities. *INFORMS Journal on Computing*, 11:217–238, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-01.ps.

[2] 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.

[3] V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods*. John Wiley & Sons, New York, 1998.

[4] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–58, 1993.

[5] G. Fung and O. L. Mangasarian. Data selection for support vector machine classification. In R. Ramakrishnan and S. Stolfo, editors, *Proceedings KDD-2000: Knowledge Discovery and Data Mining, August 20-23, 2000, Boston, MA*, pages 64–70, New York, 2000. Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-02.ps.

[6] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings KDD-2001: Knowledge Discovery and Data Mining, August 26-29, 2001, San Francisco, CA*, pages 77–86, New York, 2001. Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps.

[7] F. Girosi and N. Chan. Prior knowledge and the creation of "virtual" examples for RBF networks. In *Neural networks for signal processing, Proceedings of the 1995 IEEE-SP Workshop*, pages 201–210, New York, 1995. IEEE Signal Processing Society.

[8] Y.-J. Lee, O. L. Mangasarian, and W. H. Wolberg. Breast cancer survival and chemotherapy: a support vector machine analysis. Technical Report 99-10, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, December 1999. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Volume 55, 2000, 1-10. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-10.ps.

[9] Y.-J. Lee, O. L. Mangasarian, and W. H. Wolberg. Survival-time classification of breast cancer patients. Technical Report 01-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, March 2001. *Computational Optimization and Applications*, to appear. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-03.ps.

[10] O. L. Mangasarian. *Nonlinear Programming*. SIAM, Philadelphia, PA, 1994.

[11] 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.

[12] 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.

[13] O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-06.ps.

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

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

[16] M. C. O'Neill. Escherchia coli promoters: I. concensus as it relates to spacing class, specificity, repeat substructure, and three dimensional organization. *Journal of Biological Chemistry*, 264:5522–5530, 1989.

[17] D. W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In *Advances in Neural Information Processing Systems*, pages 535–543, Cambridge, MA, 1996. MIT Press.

[18] J. R. Quinlan. *Induction of Decision Trees*, volume 1. 1986.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, pages 318–362, Cambridge, Massachusetts, 1986. MIT Press.

[20] B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640 – 646, Cambridge, MA, 1998. MIT Press.

[21] G. G. Towell and J. W. Shavlik. The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

[22] G. G. Towell and J. W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70:119–165, 1994.

[23] G. G. Towell, J. W. Shavlik, and M. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 861–866, 1990.

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