Rule Extraction from Linear Support Vector Machines

Glenn Fung, Sathyakama Sandilya, R. Bharat Rao
Computer-Aided Diagnosis & Therapy, Siemens Medical Solutions, Inc.
51 Valley Stream Parkway, Malvern, PA, USA
glenn.fung@siemens.com

ABSTRACT

We describe an algorithm for converting linear support vector machines and any other arbitrary hyperplane-based linear classifiers into a set of non-overlapping rules that, unlike the original classifier, can be easily interpreted by humans. Each iteration of the rule extraction algorithm is formulated as a constrained optimization problem that is computationally inexpensive to solve. We discuss various properties of the algorithm and provide proof of convergence for two different optimization criteria We demonstrate the performance and the speed of the algorithm on linear classifiers learned from real-world datasets, including a medical dataset on detection of lung cancer from medical images. The ability to convert SVM's and other "black-box" classifiers into a set of human-understandable rules, is critical not only for physician acceptance, but also to reducing the regulatory barrier for medical-decision support systems based on such classifiers.

Categories and Subject Descriptors

I.5.m [**Pattern Recognition**]: Miscellaneous

General Terms

Algorithms

Keywords

Rule extraction, Linear classifiers, Mathematical programming, medical decision-support.

1. INTRODUCTION

Support Vector Machines (SVMs) [22, 11] and other linear classifiers are popular methods for building hyperplane-based classifiers from data sets, and have been shown to have excellent generalization performance in a variety of applications. These classifiers, however, are hard to interpret by humans. For instance, when an unlabeled example is classified by the linear classifier as positive or negative, the only explanation that can be provided is that some linear weighted sum of the variables of the example are lower (higher) than some threshold; such an explanation is completely non-intuitive to human experts. Humans are more comfortable dealing with rules that can be expressed as a hypercube with axis-parallel surfaces in the variable space.

Previous work [20, 17] and more recent work [10] included rule extraction for neural networks but very few work has been done to extract rules from SVMs or any other kind of hyperplane-based classifier. Recently Nunez et al [15] proposed a method to extract rules from an SVM classifier which involves applying a clustering algorithm first to identify groups that later define the rules to be obtained.

We propose a methodology for converting any linear classifier into a set of such non-overlapping rules. This rule set is (asymptotically) equivalent to the original linear classifier, covers most of the training examples in the hyperplane halfspace. Unlike [15] our method does not require computationally expensive data preprocessing steps (as clustering) and the rule extraction is done in a very fast manner, typically it takes less than a second to extract rules from SVM's trained on thousands of samples. Our algorithm does not required anything more complicated that solving simple linear programming problems in $2n$ variables where n is the number of input features (after feature selection).

In the next section we briefly discuss the medical relevance of this research. The ability to provide explanations of decisions reached by "black-box" classifiers is not only important for physician acceptance, but it is also a vital step in potentially reducing the regulatory requirements for introducing a medical decision-support system based on such a classifier into clinical practice. Section 3 then describes the commonly used linear support vector machine classifier and gives a linear program for it. Section 4 provides our rule extraction algorithm; Each iteration of the rule extraction algorithm is formulated as one of two possible optimization problems based on different "optimal" rule criteria. The first formulation, which seeks to maximize the volume covered by each rule, is a constrained nonlinear optimization problem whose solution can be found by obtaining the closed form solution of a relaxed associated unconstrained problem. The second formulation, which maximizes the number of samples covered by each rule, requires us to solve a linear programming problem. In Section 5 we discuss finite termination and convergence conditions for our algorithm. Section 6 summarizes our results on 4 publicly available datasets, and an additional medical dataset from our previous work [3] in

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.

building a CAD system to detect lung cancer from computed tomography volumes. We conclude in Section 7 with some thoughts on further extensions and applications.

We now describe the notation used in this paper. 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. All vectors will be column vectors. For $x \in R^n$, $\|x\|_p$ denotes the p-norm, $p = 1, 2, \infty$. A vector of ones in a real space of arbitrary dimension will be denoted by e. Thus, for $e \in R^m$ and $y \in R^m$, $e'y$ is the sum of the components of y. A vector of zeros in a real space of arbitrary dimension will be denoted by 0. A *separating hyperplane*, with respect to two given point sets $\mathcal{A}$ and $\mathcal{B}$, 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" and the symbol $\vee$ will denote the logical "or". The abbreviation "s.t." stands for "such that". 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 $i = 1, \ldots, n$.

2. MEDICAL RELEVANCE

From the earliest days of computing, physicians and scientists have explored the use of artificial intelligence systems in medicine [18]. A long-standing area of research has been building *computer-aided diagnosis* (CAD) systems for the automated interpretation and analysis of medical images [16]. Despite the demonstrated success of many such systems in research labs and clinical settings, these systems were not widely used, or even available, in clinical practice. The primary barrier to entry in the United States is the reluctance of the US Government to allow the use of "black box" systems that could influence patient treatment.

Although the Food and Drug Administration (FDA) has recently granted approval for CAD systems based on "black-box" classifiers [19], the barrier to entry remains very high. These systems may only be used as "second-readers", to offer advice after the initial physician diagnosis. More significantly, these CAD systems must receive pre-market approval (PMA). A PMA is equivalent to a complete clinical trial (similar to the ones used for new drugs), where the CAD system must demonstrate statistically significant improvement in diagnostic performance when used by physicians on a large number of completely new cases. This is a obviously a key area of research in CAD, but not the focus of this paper. The FDA has indicated that the barrier to entry for CAD systems that are able to explain their conclusions, could be significantly lowered. Note, this will not lower the barrier in terms of generalization performance on unseen cases, but the FDA is potentially willing to consider using performance on retrospective or previously seen cases and significantly reduce the number of cases needed for a prospective clinical trial. This is critical, because a full-blown clinical trial can add several years delay to the release of a CAD system into general clinical practice.

Much research in the field of artificial intelligence, and now knowledge discovery and data mining has focused on the endowing systems with the ability to explain their reasoning, both to make the consultation more acceptable to the user, and to help the human expert more easily identify errors in the conclusion reached by the system [4]. On the other hand, when building classifiers from (medical) data sets, the best performance is often achieved by "black-box" systems, such as, Support Vector Machines (SVMs). The research described in this paper will allow us to use the superior generalization performance of SVM's and other linear hyperplane-based classifiers in CAD system, and using the explanation features of the rule extraction algorithm to reduce the regulatory requirements for market introduction of such systems into daily clinical practice.

3. HYPERPLANE CLASSIFIERS: 1-NORM SUPPORT VECTOR MACHINES

We consider the problem 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, depicted in Figure 1, the linear programming support vector machine [11, 5] with a linear kernel (this is a variant of the standard SVM [22, 6]) 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).

If the classes are linearly inseparable, which is often the case in real-world datasets, then 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:

$$\begin{array}{ll} A_i w + y_i \geq \gamma + 1, & \text{for} \quad D_{ii} = 1, \\ A_i w - y_i \leq \gamma - 1, & \text{for} \quad D_{ii} = -1. \end{array} \tag{3}$$

The 1-norm of the error variable y is minimized parametrically with weight ν in (1), resulting in an approximate separating plane. This plane classifies data as follows:

$$sign(x'w - \gamma\ ) \begin{cases} = 1, & \text{then } x \in A_+, \\ = -1, & \text{then } x \in A_-, \end{cases} \tag{4}$$

where $sign(\cdot)$ is the sign function defined in the Introduction. Empirical evidence [5] indicates that the 1-norm formulation has the advantage of generating very sparse solutions. This results in the normal w to the separating plane $x'w = \gamma$ having many zero components, which implies that many input space features do not play a role in determining the linear classifier. This makes this approach suitable for feature selection in classification problems. Since our rule extraction algorithm depends directly on the features used

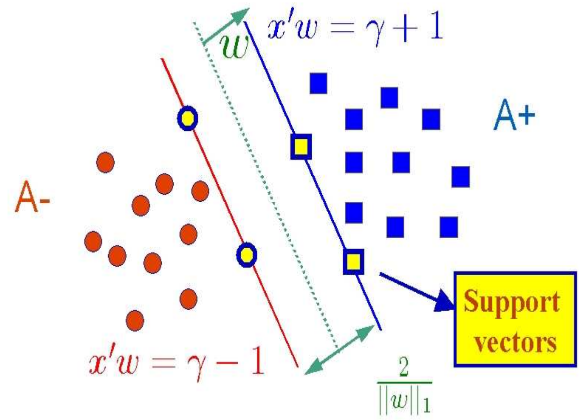

Figure 1: The LP-SVM classifier in the w-space of R^n. The plane of equation (3) approximately separating points in $A+$ from points in $A-$.

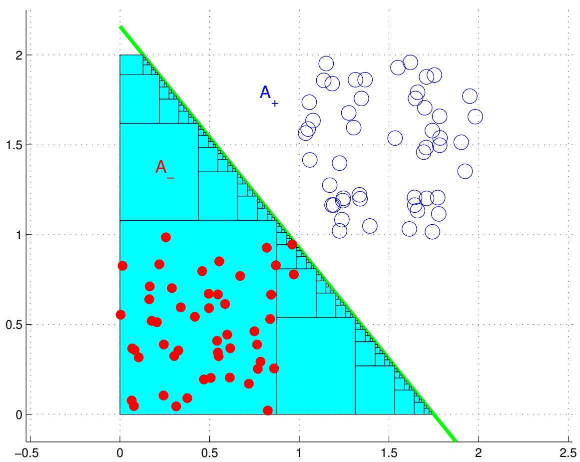

Figure 2: Two-dimensional example where the non-overlapping rules covering the halfspace ($\{x$ s.t. $w'x < \gamma\}$) are represented as cyan rectangles

.

by the hyperplane classifier, sparser normal vectors w will lead to rules depending on a fewer number of features.

4. RULE EXTRACTION FROM HYPERPLANE CLASSIFIERS

In the previous section, we described a linear programming SVM formulation to generate hyperplane classifiers. We now present an algorithm to extract rules of the form:

$$\wedge_{i=1}^{n} l_i \leq x_i < u_i$$

to approximate these classifiers. Note that every rule form defined above defines an hypercube in the n dimensional space with edges parallel to the axis. Rule of this form are very intuitive and can be easily interpreted by humans.

Our rule extraction approach can be applied to any linear classifier regardless of the algorithm or criteria used to construct the classifier, including Linear Fisher Discriminant (LFD) [13], Least squares SVM's (LS-SVMs) [21] or Proximal SVMs (PSVM) [7]. Denote by $P_{-}(w, \gamma, I)$ the problem of constructing rules for the classifier for the region:

$$I = \{x \text{ s.t. } w'x < \gamma,\, l_i \leq x_i \leq u_i,\ 1 \leq i \leq n\}$$

based on the classification hyperplane $w'x = \gamma$ obtained by solving problem (1). Note that the problem of rule extraction $P_{+}(w, \gamma, I')$ where

$$I' = \{x \text{ s.t. } w'x > \gamma,\, l_i \leq x_i \leq u_i,\ 1 \leq i \leq n\}$$

is the same as $P_{-}(-w, -\gamma, I)$. We now establish that this is equivalent to solving the problem with positive hyperplane coefficients, $\gamma = 1$ and the feature domain being the unit hypercube. Consider a diagonal matrix T constructed in the following way:

$$T_{ii} = \frac{sign(w_i)}{u_i - l_i},\ \ i \in \{1, \ldots, n\} \tag{5}$$

and a vector b with components $b = \{u_i \text{ if } w_i < 0,\ l_i \text{ if } w_i > 0\}$ We now define a transformation of coordinates such that $y = T(x - b)$. Note that

$$w_i > 0 \Rightarrow 0 \leq y_i = T_{ii}(x_i - l_i) = \frac{x_i - l_i}{u_i - l_i} \leq 1$$

$$w_i < 0 \Rightarrow 0 \leq y_i = T_{ii}(x_i - u_i) = \frac{-(x_i - u_i)}{u_i - l_i} = \frac{(u_i - x_i)}{u_i - l_i} \leq 1 \tag{6}$$

hence, I is transformed to $[0, 1]^n$. Furthermore $x = T^{-1}y + b$, and hence, the hyperplane of interest becomes

$$w'T^{-1}y = \gamma - w'b$$

which is equivalent to:

$$\tilde{w}y = \left(\frac{w'T^{-1}}{\gamma - w'b}\right) y = 1 \tag{7}$$

Thus the problem becomes $P_{-}(\tilde{w}, 1, I_0)$ in the new domain, where $I_0 = [0, 1]^n$. [1] Note that the components of $\tilde{w}$ are positive as $w'b < \gamma$ and $w_i T_{ii} > 0$.

For the rest of this paper we will concentrate in finding rules with the following properties:

[1]In mapping the original problem to the unit hypercube the measure of volume is merely a scaled version of the original problem, and thus the optimum remains the same.

- The hypercube defined by the extracted rule

$$\wedge_{i=1}^{n} l_i \leq x_i < u_i$$

is a subset of the bounded region $I = \{x \text{ s.t. } w'x < \gamma\}$

- The resulting hypercube cube defined by the extracted rule contains one vertex that lies in the separating hyperplane $w'x - \gamma = 0$. This assumption allows to obtain set of disjoint rules that are easy to generate and simplifies the problem considerable.

Figure 2 illustrates an example in two dimensions where the halfspace $w'x < \gamma$ is almost totally covered by rules represented by hypercubes with a vertex in the hyperplane $w'x - \gamma = 0$.

Given a region I we can define the "optimal" rule according to different criteria, Next we present two of them.

4.1 Volume Maximization Criteria

An optimal rule can be defined as the rule that covers the hypercube with axis-parallel faces with the largest possible volume. Since the log function is strictly increasing, $\arg\max f(x) = \arg\max \log(f(x))$, we can find the rule that maximizes the log of the volume of the region that it encloses (instead of the volume). Assuming that the linear transformation T was already applied and that one corner of the region lies on the hyperplane, this rule can be found by solving the following problem:

$$\max_{x \in R^n} \log(\prod_{i=1}^{n} x_i) \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i x_i = \gamma, 0 \leq x \leq 1 \tag{8}$$

The Lagrangian function for this nonlinear constrained optimization problem is:

$$L(x, \lambda, \theta) = \log(\prod_{i=1}^{n} x_i) - \lambda(w'x - \gamma) - \sum_{i=1}^{n} \theta_i (x-1) + \sum_{i=1}^{n} \delta_i x \tag{9}$$

The KKT optimality conditions for problem 8 are given by:

$$\begin{array}{rl} \frac{1}{x_i} - \lambda w_i - \theta_i + \delta_i = 0 & \forall i \in \{1, \ldots, n\} \\ w'x = \gamma & \\ 0 \leq x_i \leq 1, \lambda \geq 0, \theta_i \geq 0, \delta_i \geq 0 & \forall i \in \{1, \ldots, n\} \\ \theta_i (x_i - 1) = 0 & \forall i \in \{1, \ldots, n\} \\ \delta_i x_i = 0 & \forall i \in \{1, \ldots, n\} \end{array} \tag{10}$$

In order to find a solution for problem (8) we will first consider solutions for the relaxed equality constrained problem:

$$\max_{x \in R^n} \log(\prod_{i=1}^{n} x_i) \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i x_i = \gamma, \tag{11}$$

The KKT optimality conditions for problem (11) (which are very similar to the KKT conditions of problem (8)) are given by:

$$\frac{1}{x_i} - \lambda w_i = 0, \; i \in \{1, \ldots, n\}, \quad wx - \gamma = 0 \tag{12}$$

From the KKT optimality conditions (12) we obtained the following closed form solutions for the relaxed optimization problem:

$$\tilde{x_i} = \frac{1}{\lambda w_i} = \frac{\gamma}{n w_i} \; i \in \{1, \ldots, n\}, \quad \tilde{\lambda} = \frac{n}{\gamma} \tag{13}$$

A solution x^* of the original optimization problem (8) can be obtained from the solution (13). Let's define x^* as follows:

$$x_i^* = \left\{ \begin{array}{ll} \frac{1}{\lambda^* w_i} & \text{if} \quad \tilde{x_i} \leq 1, \; i \in \{1, \ldots, n\} \\ 1 & \text{otherwise} \end{array} \right\} \tag{14}$$

Where,

$$\lambda^* = \frac{n_I}{\gamma - \sum_{\{i \in A\}} w_i} \tag{15}$$

where $A = \{i \,|\tilde{x_i} > 1\}$ and n_I is $n - |A|$. with λ^* defined as above we have that:

$$\begin{aligned} wx^* - \gamma &= \sum_{i=1}^{n} w_i x_i^* = \sum_{i \in I} w_i x_i^* + \sum_{i \in A} w_i x_i^* - \gamma \\ &= \textstyle\sum_{i \in I} \frac{w_i}{\lambda^* w_i} + \sum_{i \in A} w_i - \gamma \\ &= \frac{n_I}{\lambda^*} + \sum_{i \in A} w_i - \gamma \\ &= n_I \frac{\gamma - \sum_{i \in A} w_i}{n_I} + \sum_{i \in A} w_i - \gamma \\ &= \gamma - \gamma = 0 \end{aligned} \tag{16}$$

if $0 \leq x_i^* \leq 1, \forall i \in \{1, \ldots, n\}$, then x^* is the optimal solution for problem 8, otherwise define $\tilde{x} = x^*$ and recalculate x^* until $0 \leq x_i^* \leq 1, \forall i \in \{1, \ldots, n\}$. This iterative procedure can be seen as a gradient projection method for which convergence is well established [2, 1].

4.2 Point Coverage Maximization Criteria

Another optimal rule can be defined as the rule that covers the hypercube with axis-parallel faces with that contains the largest possible number of training points in the halfspace. Given a transformed problem $P_-(\tilde{w}, 1, I_0)$, we want to find x^* such that $w'x^* - \gamma = 0$ and $|C|$ (cardinality of C)is maximal, where:

$$C = (A_- \cap \{x | \, w'x < 1\}) \cap \{x | \, 0 \leq x \leq x^*\}$$

The following Linear programming formulation is an approximation to this problem :

$$\begin{array}{llll} \min\limits_{x,y} & e'y & & \\ \text{s.t.} & w'x & = 1 & \\ & A_{.i} - ey_i & \leq x_i & , \quad \forall i \in \{1, \ldots, n\} \\ & & & 0 \leq x \leq 1 \\ & & & y \geq 0. \end{array} \tag{17}$$

Note that the variable $y \geq 0$ acts as a slack or error variable that is minimized in order in order for the rule to cover the largest possible amount of points.

We can now use either one of the optimal rule definitions described in subsections 4.1 and 4.2 to propose an iterative procedure that extract as many rules as we require to describe adequately the region of interest. We first demonstrate that in a n-dimensional feature space, extracting one such a rule results in n new similar problems to solve. Let the first rule extracted for the transformed problem $P_-(\tilde{w}, 1, I_0)$ be $\wedge_{i=1}^{n}(0 \leq x_i < x_i^*)$. The remaining volume on this side of the hyperplane that is not covered, is the union of n nonintersecting regions similar to the original region, namely

$$I_i = \begin{cases} x \in R^n, \text{s.t.} \quad 0 \le x_j < x_j^* \quad \forall 1 \le j < i \\ \qquad\qquad\quad x_i^* \le x_i < 1 \\ \qquad\qquad\quad 0 \le x_j < 1 \quad \forall j > i \end{cases} \tag{18}$$

that is, the rule inequalities for the first $i-1$ components of x are satisfied, the inequality that relates to the i^{th} component is not satisfied, and the rest are free. Consider i, j with $j > i$. For each $x \in I_j$, we have $0 \le x_i < x_i^*$ and for each $x \in I_i$, we have $x_i^* \le x_i < 1$. Hence, I_i are nonintersecting, and the rules that we arrive at for each I_i will be "independent". Now we extract the optimal rule for each of these regions that contains a training data point using a depth first search. Note that the problem for I_i is $P_-(\tilde{w}, 1, I_i)$, and we can now use the same transformation as described in equations (5)-(7) to transform each of the n subproblems $P_-(\tilde{w}, 1, I_i)$ to problems equivalent to the original problem $P_-(\tilde{w}, 1, I_0)$.

Next, we state our algorithm to obtain a set of rules R that cover all the training points belonging to A_- such that $w'x < \gamma$. Let R be the set containing all the extracted rules, and U be the set containing the indices of the points uncovered by the rules in R. R and U are initialized to $\emptyset$ and A_- respectively, d_{max} (which bounds the maximum depth of the depth first search, typically less than 20) is assigned, and w, γ are obtained by solving the LP-SVM (1) before **ExtractRules** is invoked for the first time.

ALGORITHM 4.1. ***ExtractRules***(w, γ, I, d)**: Algorithm for rule extraction from linear classifiers.**

1. *If* $d = d_{max}$*, stop.*
2. *Transform problem* $P_-(w, \gamma, I)$ *into* $P_-(\tilde{w}, 1, I_0)$ *using the linear transformation described in Section 5, equations (5)-(10).*
3. *Obtain* y^* *by solving problem* $P_-(\tilde{w}, 1, I_0)$ *using either equations (14)-(15) or equation (17).*
4. *Calculate* $x^* = T^{-1}y^* + b$*, get new rules* $\tilde{R}(x^*)$*, update* $R \leftarrow R \cup \tilde{R}(x^*)$*.*
5. *Let* $C = \{x \in U \text{ st. } \tilde{R}(x^*) \text{ is true}\} = U \cap \tilde{R}(x^*)$*, this is, a set containing the indices of the points in* U *that are covered by the new obtained rule.*
6. *Update* $U \leftarrow U - C$*. If* $U = \emptyset$*, stop. Else* $d \leftarrow d + 1$*.*
7. *for* $k = 1$ *to* n *do*
   - *Calculate* $\hat{I}_k = T^{-1}I_k + b$*. If* $U \cap \hat{I}_k \ne \emptyset$ *apply recursively* ***ExtractRules***$(w, \gamma, \hat{I}_k, d)$*, where* $\hat{I}_k$ *is one of the n remaining regions of interest uncovered by rule* $\tilde{R}(x^*)$ *as defined in (18).*

5. ALGORITHM CONVERGENCE PROPERTIES

We now derive the rate at which the volume covered by the rules extracted for $P(w, 1, I_0)$ converges to the total volume of the region of interest.

Lemma: The volume of the region $\{x \text{ s.t. } w'x < \gamma, x_i \ge 0\}$ is

$$V_n(w, \gamma) = \frac{\prod_{i=1}^{n} \frac{\gamma}{w_i}}{n!}$$

Proof: We show this by induction. For $n = 2$, this is the area of a right-angled triangle with sides γ/w_1 and γ/w_2, which is $\gamma^2/2w_1w_2$. Now, assume that this is true for $n = k$.

$$\begin{aligned} V_{k+1}(w,\gamma) &= \int_0^{\gamma/w_1} \dots \int_0^{(\gamma - w_1x_1 - \dots - w_kx_k)/w_{k+1}} dx_1 dx_2 \dots dx_{k+1} \\ &= \int_0^{\gamma/w_1} dx_1 \int_0^{(\gamma - w_1x_1)/w_2} dx_2 \dots \int_0^{(\gamma - w_1x_1 \dots - w_kx_k)/w_{k+1}} dx_{k+1} \\ &= \int_0^{\gamma/w_1} dx_1 V_k(w_{-1}, \gamma - w_1x_1) \\ &= \int_0^{\gamma/w_1} dx_1 \frac{1}{k!} \prod_{i=2}^{k+1} \frac{\gamma - w_1x_1}{w_i} \\ &= \frac{1}{k!} \prod_{i=2}^{k+1} \frac{1}{w_i} \int_0^{\frac{\gamma}{w_1}} dx_1 (\gamma - w_1x_1)^k \\ &= \frac{1}{k!} (\prod_{i=2}^{k+1} \frac{1}{w_i}) \frac{\gamma^{k+1}}{(k+1)w_1} = \frac{1}{(k+1)!} \prod_{i=1}^{k+1} \frac{\gamma}{w_i} \end{aligned}$$

where w_{-i} contains all components of w except the i-th.

Lemma: For any $S \subseteq \{1, 2, \dots, n\}$, the volume of a region defined by $w'x < 1$ and $0 \le x_i < 1, 1 \le i \le n$ is bounded by

$$\frac{1}{|S|!} \prod_{i \in S} \frac{1}{w_i}$$

Proof: We can assume without loss of generality that S is $\{1, 2, \dots, k\}$ (if it is not, the coordinates may be permuted so that it is). The volume of interest, say V is given by

$$\begin{aligned} V &= \int_0^{\min(1,1/w_1)} dx_1 \int_0^{\min(1,(1-w_1x_1)/w_2)} dx_2 \dots \\ &\quad \dots \int_0^{\min(1,(1-w_1x_1-\dots-w_{n-1}x_{n-1})/w_n)} dx_n \\ &\le \int_0^{\min(1,1/w_1)} \dots \int_0^{\min(1,(1-w_1x_1-\dots-w_{k-1}x_{k-1})/w_k)} \dots \\ &\quad \dots \int_0^1 \dots \int_0^1 dx_1 dx_2 \dots dx_n \\ &\le \int_0^{1/w_1} \dots \int_0^{(1-w_1x_1-\dots-w_{k-1}x_{k-1})/w_k} dx_1 dx_2 \dots dx_k \\ &= \frac{1}{k!} \prod_{i=1}^{k} \frac{1}{w_i} \end{aligned}$$

where the first two inequalities are because the upper limit in the integral is replaced by an upper bound, and the last equality comes from the previous lemma with $\gamma = 1$.

Lemma: At each "stage", the algorithm covers at least $\alpha = \frac{n!}{n^n}$ of the volume yet to be covered. Hence,the volume remaining after k stages is at most $(1 - \alpha)^k V_0$.

Proof: The volume covered by the rule is given by

$$\begin{aligned} V_{rule} &= \prod_{i=1}^{n} x_i^* = (\prod_{i\notin A} \frac{1}{\lambda^* w_i})(\prod_{i\in A} 1) \\ &= \prod_{i\notin A} \frac{1}{w_i} \frac{1-\sum_{i\in A} w_i}{n-|A|} \\ &\geq \prod_{i\notin A} \frac{1}{w_i} \frac{1-|A|/n}{n-|A|} \\ &= \prod_{i\notin A} \frac{1}{w_i} \frac{1}{n} \end{aligned}$$

where A as before is the set of active constraints, and the inequality above comes from the fact that for $i \in A$, $\frac{1}{nw_i} > 1$ (the original solution to the relaxed problem violates the constraints). Using the result of the previous lemma, and setting $S = \{1, \ldots, n\} \backslash A$, we have

$$\frac{V_{rule}}{V_{total}} \geq \frac{\prod_{i\notin A} \frac{1}{w_i} \frac{1}{n}}{\frac{1}{(n-|A|)!} \prod_{i\notin A} \frac{1}{w_i}} = \frac{(n-|A|)!}{n^{n-|A|}} \geq \frac{n!}{n^n}$$

the last inequality arises because the bound is monotonically increasing in $|A|$ with it being the smallest when $|A| = 0$.

Lemma: At each stage, the algorithm reduces the largest distance from an interior point yet to be covered to the separating hyperplane by a factor of $1 - 1/n$.

Proof: We establish the lemma for one stage of $P(w, 1, I_0)$ (a simple scaling argument would extend it to a general γ and I, and hence to further stages of the problem as well). The largest distance from the plane in I_0

$$d^0_{max} = \sup_{x\in I_0, w'x<1} (1 - w'x)/||w|| = 1/||w||$$

In region I_i, as $x_i \geq x_i^*$ and $w'x$ is monotonically increasing in each coordinate

$$d^i_{max} = \sup_{x\in I_i, w'x<1} (1 - w'x)/||w|| = (1 - w_i x_i *)/||w||$$

When $i \in A$, then I_i has no interior points. When $i \notin A$, $\tilde{x_i} = w_i x_i^* = 1/n$. Hence,

$$d^i_{max} = (1 - 1/n)/||w|| = (1 - 1/n)d^0_{max}$$

Theorem: After extracting t rules, the remaining volume is at most $(1-\alpha)^{\log_n t-1}$ of the original volume. Moreover, the rule extraction algorithm covers in finite time any dataset that has all points in the interior of I.

Proof: As described before, each rule extraction leads to n further "subproblems". Hence, the number of rules to be extracted in stage k is n^{k-1}, and the number of rules extracted upto and including stage k is $\frac{n^k-1}{n-1}$. Hence, if t rules have been extracted and k stages are complete,

$$t < \frac{n^{k+1}-1}{n-1} \Rightarrow t < n^{k+1} \Rightarrow k > \log_n t - 1$$

Hence, at least $\log_n t - 1$ stages are complete, and hence, by a previous lemma, at most $(1-\alpha)^{\log_n t-1}$ of the volume remains (which converges to 0 as $t \to \infty$). Moreover, by the previous lemma we have that at the end of stage k,

$$d_{max} = (1 - 1/n)^k \gamma/||w||$$

Hence, for a data point x, we have that x is covered when

$$(\gamma - w'x)/||w|| > (1 - 1/n)^k \gamma/||w||$$

i.e. when

$$k \geq \log_{(1-1/n)} (1 - w'x/\gamma)$$

Hence, the entire data set A_- is covered when

$$k \geq \log_{(1-1/n)} (1 - \max_{x\in A_-} (w'x)/\gamma)$$

i.e., when

$$t = n^{1+\log_{(1-1/n)} (1-\max_{x\in A_-} (w'x)/\gamma)}$$

We now use this to establish termination of the algorithm for a given data set in finite time. Let us assume the contrary, i.e. that there is a point $x^\#$ such that $w'x^\# < \gamma$ and it is not covered in the rule extraction process. By the previous lemma, we have that $y = x^\# + (\gamma - w'x^\#)w/2||w||$ is not covered (as it is greater than x). Moreover, any point in the hypercuboid $x_i^\# \leq x_i < y_i$ is not covered by the rules. Hence the volume of the uncovered region is at least $\prod_{i=1}^{n} (y_i - x_i^\#)$, which is a contradiction of the previous part of the theorem. Hence, the point $x^\#$ gets covered after a finite number of iterations.

6. NUMERICAL TESTING

To show the effectiveness of our rule extraction algorithm, we performed experiments in five real-world datasets. Three of the datasets are publicly available datasets from the UCI Machine Learning Repository [14]: Wisconsin Diagnosis Breast Cancer (WDBC), Ionosphere, and Cleveland heart. The fourth dataset is a dataset related to the nontraditional authorship attribution problem related to the federalist papers [9] and the fifth dataset is a dataset used for training in a computer aided detection (CAD) lung nodule detection algorithm, we refer to this set as the Lung CAD dataset. Experiments for the five datasets were performed to test the capability of algorithm 4.1 to cover training points correctly classified by the SVM hyperplane. For each experiment, we obtained a separating hyperplane using the 1−norm linear programming SVM (LP-SVM) formulation as described in equation (1). The state of the art optimization software CPLEX was used to solve the corresponding linear programming problems. Ten-fold cross validation was used as a tuning procedure to determine the SVM parameter ν. In All the experiments, the resulting hyperplane classifier was sparse, this means that the set $\{w_i \text{ s.t. } w_i \neq 0, \ 1 \leq i \leq n\}$ was "small", this was expected because of the effect of the 1−norm regularization term on the coefficients w_i. Having a sparse hyperplane implies that the dimensionality of the training dataset can be reduced by discarding the features corresponding to $w_i = 0$ since they do not play any role in the classification.

Once the hyperplane was obtained we applied algorithm 4.1 using one of the two criteria for optimal rules described in subsections 4.1 and 4.2. The first criteria is based in finding rules that maximizes the volume of the region covered by the rule, we will refer to this variant of algorithm 4.1 as Volume Maximization (**VM**). The second criteria is to find rules that attempt to cover a many points of the training set as possible. We will call this variant of algorithm 4.1 Point Coverage Maximization (**PCM**).

Results for both VM and PCM are reported in Tables 1 and 2 including: total number of optimization problems solved, total execution time, total number of extracted rules and percentage of correctly classify points by the hyperplane that were covered by the extracted rules.
It is important to note that the results reported included only rules that covered more than one point. We considered that rules that covered only one point did not have any generalization capability and therefore were discarded. In general, the algorithm can be tuned to discard rules that do not cover enough points according to a number predefined by the user.

Empirical results on the five datasets as reported in Tables 1 and 2 show the effectiveness of both the VM and PCM variants of our proposed algorithm. In most cases our algorithms covered more of 90% of the training points using only a few rules. As was expected, the VM variant seems to solve more "easy" optimization problems and generate more rules. On the other hand, the PCM variant solved fewer optimizations problems (linear programming problems) but that were slightly harder to solve, generating fewer rules.

Note that Tables 1 and 2 appear at the end of this paper (after the references). Next, we will discuses in more detail the results obtained for the WDBC dataset and the lung-cad dataset since medical diagnosis applications is of special interest to us.

6.1 WDBC dataset

The first experiment relates to the publicly available WDBC dataset that consists of 683 patient data. The classification task associated with this dataset is to diagnose breast masses based solely on a Fine Needle Aspiration (FNA). Doctors identified nine visually assessed characteristics or attributes of an FNA sample which they considered relevant to diagnosis (for more detail please refer to [12]). After applying the LP-SVM algorithm and discarding the features corresponding to the $w_i = 0$, we ended up with a hyperplane classifier in 5 dimensions that achieved 95.0% tenfold testing set correctness. After applying our **ExtractRules-PCM** algorithm to cover the 214 points in A_- correctly classified by the hyperplane we obtained a total of 7 non empty, non-singleton rules that cover 99.8% of the points. Similarly we obtained a total of 3 non empty, non-singleton rules that cover 98.1% of points in A_+ correctly classified by the hyperplane. For example after using the fact that all the features values are integers between 1 and 10 we obtained the following rule that covered 383 of the 435 positive training points:

$$\begin{aligned}(Cell\ Size \leq 3) \quad \wedge \quad & (Bare\ Nuclei \leq 1) \\ & \wedge\ (Normal\ Nucleoli \leq 7)\end{aligned}$$
$$\Rightarrow \ mass\ is\ benign$$

6.2 The Lung CAD dataset

The second experiment relates to a set of data used in a computer aided detection (Lung CAD) system for pulmonary nodule detection on thin slice multidetector CT scans. The Lung CAD algorithm performs the following processing steps: a) lung segmentation; b) candidate generation; c) feature calculation at each candidate location; d) classification and e) presenting CAD findings to a physician for review. The task of the candidate generation step, is to reduce the search space by quickly generating a list of suspicious locations for different types of nodules at a high sensitivity without considering the specificity. For this, shape based characteristics are used to generate a candidate list. For each candidate in the list a set of features is calculated. Those features are based on the intensity, the shape, the curvature, and the location. The goal of the last processing step is to increase the specificity without decreasing the sensitivity by pruning the list of candidates. For this, a classifier is used. Our dataset consists on 274 candidates represented by 34 numerical features. Each datapoint corresponds to a candidate labeled as a nodule or not a nodule. The LP-SVM algorithm generated a classifier in only 5 features with 82.5% tenfold testing set correctness. Our **ExtractRules-PCM** algorithm extracted a total of 10 nonempty, non-singleton rules that cover 94% of positive training points correctly classified for the hyperplane, similarly we obtained a total of only 5 nonempty, non-singleton rules that cover 98.4% of the points in A_+ correctly classified for the hyperplane.

7. CONCLUSION & FUTURE DIRECTIONS

We have described an efficient algorithm for converting any arbitrary linear classifier into a rule set that can be easily interpreted by humans. We presented two variants of our algorithm based on different criteria for selecting "optimal rules". One main advantage of our algorithm is that it only involves solving relatively simple optimization problems in a few variables. We also discussed various properties and provided a detailed convergence analysis of the algorithm. Empirical results on several real-world data sets demonstrate the efficacy and speed of our method.

We plan to extend our numerical results to include comparisons to other rule-based classification methods. We are also considering other mathematical programming formulations where the rules can overlap since overlapping rules may have an advantage that may depend on the specific problem.

An interesting extension of this work would be to combine our rule extraction algorithm with a recently proposed Knowledge-based SVM [8] to design an incremental algorithm to handle massive amounts of data. The algorithm could "compress" training data in the form of rules obtained form different "chunks" and then integrate all the obtained rules into a Knowledge-based SVM.

The incorporation of the feature selection into the rule extraction problem is also a possibility we are exploring at this moment. This approach would generate rules that depend on different features instead of depending on the same preselected subset of features.

So far we have focused on developing rule sets that are human interpretable models that are equivalent to the original linear classifier. An equally important use of our method would be to provide an explanation of the classification for a new unlabeled (test) example. The most obvious way is to present the user with the specific rule that includes the test example. For instance, when working with physicians, we have found that an explanation of a classification label which is in terms of a bounding hypercube, is far more understandable than "explaining" a label because some weighted sum of the variables is less than some constant.
The interesting case arises when no rule covers the test example. The obvious extension to execute **ExtractRules** on the region I which contains the test example, until a covering rule is found. However, the resulting rule may cover a very small volume around the test example, rendering the explanation useless. An alternate approach is not to build

Table 1: Results using maximal area formulation, # of optimization problems solved, total execution time (in seconds), Number of rules and % of correctly classified points covered are shown for both classes A_- and A_+ on six datasets

Data Set $m \times n, card(A_-), card(A_+)$ # of features	# prob. solved $\hat{A_-}$ $\hat{A_+}$	Time $\hat{A_-}$ $\hat{A_+}$	# of points $\hat{A_-}$ $\hat{A_+}$	# rules $\hat{A_-}$ $\hat{A_+}$	Coverage % $\hat{A_-}$ $\hat{A_+}$
Lung CAD $274 \times 34, 137, 137$ 5	33 43	0.14 0.17	124 102	18 20	97.6 % 100.0 %
WPBC $683 \times 9, 444, 239$ 5	53 12	0.23 0.11	214 435	28 9	100.0% 100.0 %
Ionosphere $351 \times 34, 225, 126$ 6	46 29	0.17 0.19	70 224	19 11	100.0 % 100.0 %
Cleveland $297 \times 13, 214, 83$ 6	102 68	0.30 0.20	53 195	10 22	79.3 % 98.4 %
Federalist $106 \times 70, 50, 56$ 6	22 23	0.17 0.19	50 52	4 6	90.0 % 92.00 %

a rule set that is equivalent to the entire classifier, but instead to revise the original problem defined in (8) to extract just one rule – the largest possible hypercube (rule) which contains the test example. Such a rule, however, may not have much explanatory value because in most cases the test example will lie on one of the surfaces of the hypercube.

A more satisfactory explanation for a test sample may be provided by a rule where the example lies well within the interior of the rule, far away from the bounding spaces. The rule that provides the "optimal" explanation, can be created by drawing a normal from the test sample to the hyperplane, and the intersection of the normal with the hyperplane defines the corner of a uniquely defined bounding hypercube (rule), which centrally contains the test sample. Additionally, we can provide a confidence associated with the explanation (rule); ideally the explanation rule should cover all training examples in A_+ (A_-), contain only all positive (negative) training samples, be as large as possible (the volume ratio with respect to the rule created by **ExtractRules**), and for the test sample to be as far from the hyperplane. All these factors may be used to adjust the confidence associated with the rule (for the specific test sample) by weighting it using some scoring scheme. In general, these criteria may be applied to any explanatory rule, not just the "optimal" explanatory rules created as defined above.

8. REFERENCES

[1] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

[2] Dimitri P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20:221–246, 1982.

[3] F. Beyer, L. Zierott, J. Stoeckel, W. Heindel, and D. Wormanns. Computer-assisted detection (cad) of pulmonary nodules at mdct: Can cad be used as concurrent reader? In *Proceeding of the 11th European Congress of Radiology*, Viena, Austria, March 2005. To appear.

[4] E.H.Shortliffe B.G.Buchanan. *Rule-Based Expert Systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984.

[5] 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.

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

[7] 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.

[8] G. Fung, O. L. Mangasarian, and J. Shavlik. Knowledge-based support vector machine classifiers. Technical Report 01-09, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, November 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-09.ps, NIPS 2002 Proceedings, to appear.

[9] Glenn Fung. The disputed federalist papers: Svm feature selection via concave minimization. In *TAPIA '03: Proceedings of the 2003 conference on Diversity in computing*, pages 42–46. ACM Press, 2003.

Table 2: Results using the maximal coverage formulation, # of optimization problems solved, total execution time (in seconds), Number of rules and % of correctly classified points covered are shown for both classes A_- and A_+ on six datasets

Data Set	# prob. solved	Time	# of points	# rules	Coverage %
$m \times n, card(A_-), card(A_+)$	$\hat{A_-}$	$\hat{A_-}$	$\hat{A_-}$	$\hat{A_-}$	$\hat{A_-}$
# of features	$\hat{A_+}$	$\hat{A_+}$	$\hat{A_+}$	$\hat{A_+}$	$\hat{A_+}$
Lung CAD	5	0.09	124	4	98.4 %
$274 \times 34, 137, 137$	10	0.16	102	9	94.1 %
5					
WPBC	10	0.17	214	7	98.1%
$683 \times 9, 444, 239$	3	0.14	435	3	99.8 %
5					
Ionosphere	11	0.17	70	7	87.2 %
$351 \times 34, 225, 126$	5	0.09	224	2	98.2 %
6					
Cleveland	32	0.33	53	7	81.1 %
$297 \times 13, 214, 83$	11	0.19	195	11	97.44 %
6					
Federalist	2	0.08	50	2	100.0 %
$106 \times 70, 50, 56$	13	0.25	52	11	96.2 %
6					

[10] F. J. Kurfes. Neural networks and structured knowledge: Rule extraction and applications. *Applied Intelligence (Special Issue)*, 12(1-2):7–13, 2000.

[11] 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.

[12] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.

[13] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

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

[15] Haydemar Nuñez, Cecilio Angulo, and Andreu Catal. Rule extraction from support vector machines. In *ESANN'2002 proceedings - European Symposium on Artificial Neural Networks*, pages 107–112. d-side, 2002.

[16] K. Preston. Computer processing of biomedical images. *Computer*, 9:54–68, 1976.

[17] A. Tickle R. Andrews, J. Diederich. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8:373–389, 1995.

[18] L. B. Lusted R. S. Ledley. Reasoning foundations of medical diagnosis. *Science*, 130:9–21, 1959.

[19] J. Roehrig. The promise of cad in digital mammography. *European Journal of Radiology*, 31:35–39, 1999.

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

[21] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

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