

Variable Selection for SVM via Shrinkage Methods

Hao Helen Zhang, Jeongyoun Ahn, Xiaodong Lin, and Cheolwoo Park

1 Introduction

Modern technologies in many scientific fields allow us to produce and store large data sets with ever increasing sample sizes and dimensions, which often include some superfluous variables or information. Effective variable selection procedures are thus desired to improve both accuracy and interpretability of the learning techniques (Kittler (1986))

Support vector machine (SVM) algorithm is a large-margin classifier with mathematical tractability and geometrical interpretation, and has shown successful performances in real-world classification problems. However, the standard SVM can suffer from the appearance of redundant variables, since its decision rule utilizes all the input variables without discrimination. To conduct simultaneous variable selection and classification in the SVM, we study shrinkage methods which systematically threshold small coefficients to zero thus reduce dimensionality of the input space automatically. We first review the L_1 SVM, then propose the SCAD SVM which employ a nonconvex shrinkage penalty in the SVM learning. Compared with the L_1 SVM, the SCAD SVM is shown to build a more compact classifier while achieving a higher classification accuracy. A sequential quadratic program algorithm is developed to minimize the non-differentiable and nonconvex objective function in the SCAD SVM by solving a series of linear equation systems. The new method is finally applied to a microarray gene expression data and a metabolism data, and both examples demonstrate its potential for analyzing “high dimensional and low sample size” data.

2 Support Vector Machine

In a standard two-class classification problem, a set of training data $(\mathbf{x}_i, y_i), \dots, (\mathbf{x}_n, y_n)$ are observed, where the input $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ and the output $y_i \in \{+1, -1\}$ is the class label. The learning task is to train a classification rule $f : \mathcal{X} \rightarrow \{+1, -1\}$ which will be used to predict the labels for new inputs. In a statistical framework, the data points are independent and identical realizations of a random pair (\mathbf{X}, Y) following the joint distribution $P(\mathbf{X}, Y)$. Denote the conditional probability $p(\mathbf{x}) = \text{Prob}(Y = +1 | \mathbf{X} = \mathbf{x})$. In decision theory, if we use the 0-1 loss $l[f(\mathbf{x}), y] = 1$ if $yf(\mathbf{x}) < 0$; $= 0$ otherwise, the optimal rule which minimizes the expected loss $El[f(\mathbf{X}, Y)]$ is $\text{sign}[p(\mathbf{x}) - 1/2]$. We often call it the Bayes rule.

The introduction of support vector machines (SVMs) [Boser et al. (1992), Vapnik (1995), Vapnik (1998)] has been an important innovation in machine learning. The SVM was first proposed as a large-margin classifier which separates two linearly-separable classes by maximizing the geometrical margin between them. To handle non-separable classification problems, the soft-margin SVM was developed by introducing slack variables to control the upper bound of the misclassification error. The linear SVM is a regularization method which finds $f(\mathbf{x}) = b + \mathbf{w} \cdot \mathbf{x}$ such that

$$\min_{b, \mathbf{w}} \frac{1}{n} \sum_{i=1}^n [1 - y_i f(\mathbf{x}_i)]_+ + \lambda \|\mathbf{w}\|^2, \quad (2.1)$$

where b is the constant, \mathbf{w} is the directional vector, and λ is the tuning parameter. The loss function $[1 - yf]_+$ is the so-called hinge loss function. For classifying the data with complicated structures where a linear separation is not plausible, the nonlinear SVM maps the data \mathbf{x} in the original input space into a higher q -dimensional feature space \mathcal{H} (q can be infinite), and then implements a linear classification in \mathcal{H} . Let $h(\mathbf{x}) = \{h_1(\mathbf{x}), \dots, h_q(\mathbf{x})\}$ be a dictionary of basis functions in \mathcal{H} . The nonlinear SVM rule is obtained by solving

$$\min_{b, \mathbf{w}} \frac{1}{n} \sum_{i=1}^n [1 - y_i (b + \mathbf{w} \cdot h(\mathbf{x}_i))]_+ + \lambda \|\mathbf{w}\|^2. \quad (2.2)$$

In many algorithms, the only quantities required are of the form $h(\mathbf{x}_i) \cdot h(\mathbf{x}_j)$ and thus a kernel function satisfying $K(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_i) \cdot h(\mathbf{x}_j)$ is often used for computational convenience. This technique is referred as the “kernel trick”. Lin (2002) showed that the SVM solution directly targets on the Bayes rule and is Fisher consistent.

However, the solution of the SVM utilizes all the input covariates without discrimination, which can be harmful for prediction in the appearance of redundant variables. Hastie et al. (2001) gave an illustrative example on this. Several approaches for variable selection in SVMs have been suggested, which either combine (Furey et al. (2000); Grandvalet and Canu (2002)) or integrate (Chaple et al. (2002); Weston et al. (2000); Bi et al. (2003); Rakotomamonjy (2003)) variable selection with SVMs. Guyon et al. (2002) further developed the recursive feature elimination (RFE) algorithm which successively eliminates features by training a sequence of SVM classifiers. Different from all the methods mentioned above, we consider the shrinkage methods to conduct variable selection in the SVM.

3 Shrinkage Methods for SVM

In literature, several methods have been suggested to replace the L_2 norm $\|\mathbf{w}\|^2$ in (2.2) with some shrinkage-type penalty on \mathbf{w} . The penalty automatically produces sparse solutions and

thus reduces the model complexity. Here is one class of shrinkage methods:

$$\begin{aligned}
 \text{hard thresholding} \quad L_0(\mathbf{w}) &= \sum_{j=1}^d I(w_j \neq 0), \\
 L_1 \text{ penalty} \quad L_1(\mathbf{w}) &= \sum_{j=1}^d |w_j|, \\
 L_q \text{ penalty} \quad L_q(\mathbf{w}) &= \left(\sum_{j=1}^d |w_j|^q \right)^{1/q}, \quad q > 0, \\
 L_\infty \text{ penalty} \quad L_\infty(\mathbf{w}) &= \max_j |w_j|.
 \end{aligned}$$

In Figure 1, the penalty functions $L_0, L_{0.5}, L_1$ and L_2 are plotted. The L_0 penalty is the entropy penalty, and it gives a hard-thresholding rule by shrinking small coefficients to zero while keeping large coefficients intact. Though the L_0 is the best theoretical choice for shrinkage purpose, its discontinuity form makes the optimization hard and tends to give unstable solutions. It is known that the L_q function is a soft-thresholding penalty only if $q \leq 1$ (Bradley and Mangasarian (1998)). This explains why the standard SVM which uses the L_2 penalty can not select variables. In Figure 1, all the penalty functions except the L_2 penalty are not differentiable at the origin, which in fact is a necessary condition for a function to be a thresholding penalty (Fan and Li (2001)).

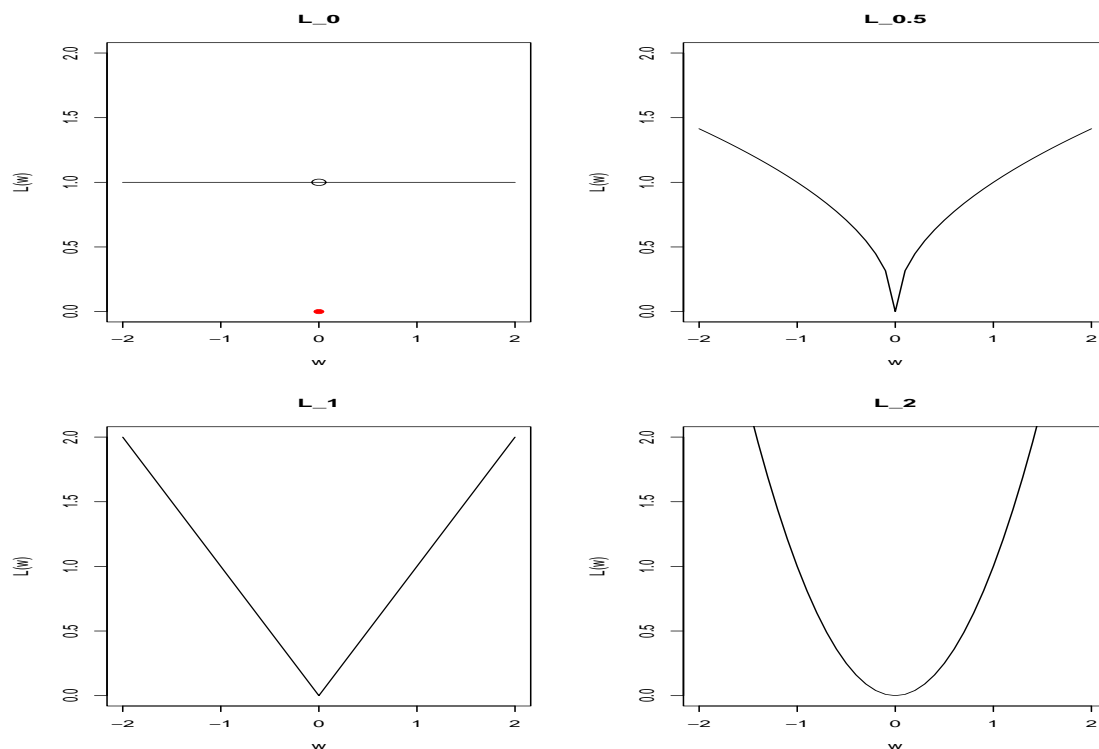


Figure 1: Various Shrinkage-type Penalty Functions

3.1 The L_1 SVM

In the statistics literature, the linear regression model with the L_1 penalty is known as the LASSO. Tibshirani (1996) gave a thorough study of the method for variable selection. In machine learning, the L_1 SVM is first suggested by Bradley and Mangasarian (1998) and it solves

$$\min_{b, \mathbf{w}} \frac{1}{n} \sum_{i=1}^n [1 - y_i(b + \mathbf{w} \cdot \mathbf{x}_i)]_+ + \lambda \sum_{j=1}^d |w_j|. \quad (3.1)$$

The L_1 penalty gives a soft-thresholding rule and yields a directional vector $\hat{\mathbf{w}}$ with many zero components. Different from the standard SVM which requires solving a linearly-constrained quadratic programming, the problem (3.1) leads to a linear programming problem:

$$\begin{aligned} \min_{\xi, \mathbf{u}, \mathbf{v}} \quad & \mathbf{1}'\xi + \lambda \mathbf{1}'(\mathbf{u} + \mathbf{v}) \\ \text{s.t.} \quad & Y(b\mathbf{1} + X\mathbf{w}) + \xi \geq \mathbf{1}, \\ & \mathbf{u}, \mathbf{v}, \xi \geq \mathbf{0}, \end{aligned} \quad (3.2)$$

where $Y = \text{diag}[y_1, \dots, y_n]$, $X_{n \times d}$ is the design matrix with the i -th row being the input vector \mathbf{x}_i . Zhu et al. (2003) studied the solution property of the L_1 SVM by considering an equivalent optimization problem of (3.1)

$$\begin{aligned} \min_{b, \mathbf{w}} \quad & \frac{1}{n} \sum_{i=1}^n [1 - y_i(b + \mathbf{w} \cdot \mathbf{x}_i)]_+ \\ \text{s.t.} \quad & \|\mathbf{w}\| = |w_1| + \dots + |w_d| \leq s, \end{aligned} \quad (3.3)$$

where $s > 0$ is the tuning parameter which has the same role as λ . Actually (3.3) is the Lagrange version of the optimization problem (3.1). They first argued that the solution of (3.3) $\mathbf{w}(s)$ is a piece-wise linear function in the tuning parameter s , and then proposed an efficient algorithm to compute the whole solution path, which facilitates adaptive selection of the tuning parameter s .

Very recently, Fung and Mangasarian (2004) developed a fast Newton algorithm NLPSVM to solve the dual problem of (3.2) by only using a linear equation solver. In addition, the NLPSVM algorithm is very effective to classify problems with a high dimensional input space and even when $d \gg n$. In our numerical studies, we use the MATLAB codes in Fung and Mangasarian (2004) to implement the L_1 SVM.

3.2 SVM via the SCAD penalty

Fan and Li (2001) showed, in the linear regression context, though the L_1 penalty produces sparse solutions, its solutions can be biased for large coefficients because larger penalties

are imposed on larger coefficients. They illustrated that a good penalty function should result in an estimator with three properties: (a) unbiasedness: the resulting estimator is nearly unbiased when the true unknown parameter is large; (b) sparsity: the resulting estimator is a thresholding rule, which automatically sets small estimated coefficients to zero; (c) continuity: the resulting estimator is continuous in data. Notice none of the $L_q (q > 0)$ penalty satisfies all the three conditions simultaneously. When $q > 1$ the solution is not sparse; when $0 < q < 1$ the solution is not continuous. In regression settings, they then proposed the smoothly clipped absolute deviation (SCAD) penalty which meets all the three conditions. In this paper, we consider employing the SCAD penalty in the SVM context for simultaneous classification and variable selection. In particular, we will study the performances of the SCAD SVM on high dimensional low sample size data.

Mathematically, the SCAD penalty function is defined as

$$p_\lambda(w) = \begin{cases} \lambda|w| & \text{if } |w| \leq \lambda \\ -\frac{(|w|^2 - 2a\lambda|w| + \lambda^2)}{2(a-1)} & \text{if } \lambda < |w| \leq a\lambda, \\ \frac{(a+1)\lambda^2}{2} & \text{if } |w| > a\lambda, \end{cases} \quad (3.4)$$

where $a > 2$ and $\lambda > 0$ are two tuning parameters. The function p_λ is symmetric and a quadratic spline function with two knots at λ and $a\lambda$. In Figure 2, we plot the SCAD function with $a = 3$ and $\lambda = 0.4$. Except its singularity at the origin, the function $p_\lambda(w)$ has a continuous first-order derivative.

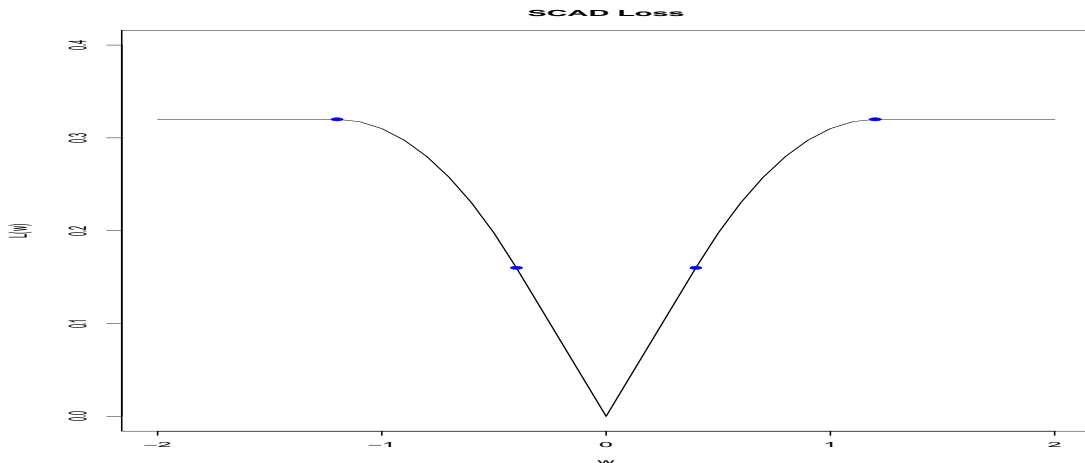


Figure 2: SCAD loss functions with $\lambda = 0.5, a = 3$.

The SCAD function has the same form as the L_1 penalty for small coefficients. However, for large coefficients, the SCAD always applies a constant penalty while the L_1 penalty

increases linearly as the coefficient increases. It is this distinct feature that guards the SCAD penalty against causing possible biases for estimating large coefficients. For the linear classification, the SCAD SVM solves

$$\min_{b, \mathbf{w}} \frac{1}{n} \sum_{i=1}^n [1 - y_i f(\mathbf{x}_i)]_+ + \sum_{j=1}^d p_\lambda(|w_j|), \quad \text{where } f(\mathbf{x}) = b + \sum_{j=1}^d w_j x_j. \quad (3.5)$$

The objective function in (3.5) consists of two parts: the data-fitting part represented by the hinge loss and the model parsimony part represented by the SCAD penalty on the directional vector \mathbf{w} . The parameter λ balances the trade-off between data-fitting and model parsimony. If λ is too small, the procedure tends to overfit the training data and gives a classifier with little sparsity; if λ is too large, the resulted classifier is very sparse but may have a poor discriminating power.

For the nonlinear SVM, the classifier $f(\mathbf{x}) = b + \sum_{j=1}^q w_j h_j(\mathbf{x})$ represented in terms of the basis functions in \mathcal{H} , solves

$$\min_{b, \mathbf{w}} \frac{1}{n} \sum_{i=1}^n [1 - y_i f(\mathbf{x}_i)]_+ + \sum_{j=1}^q p_\lambda(|w_j|) \quad (3.6)$$

In practice, the feature space \mathcal{H} should have a sufficiently rich structure to ensure a good approximation to the true separating hyperplane. For example, a space generated by

$$\{x_1, x_2, x_1^2, x_2^2, x_1 x_2\}$$

in \mathbb{R}^2 should be used when the separating boundary has a quadratic form. In complicated problems, the space \mathcal{H} containing more flexible functions such as high-order polynomials, should be considered.

Interestingly, when $d > n$, a linear classifier often shows a better performance than a nonlinear classifier in many applications (Hastie et al. (2001)), even though nonlinear classifiers are known to be more flexible. This fact is related to the asymptotic results given in Hall et al. (2004), where they showed that when $d \gg n$, under a mild assumption for the data distribution, all the pairwise distances between each pair of data points are approximately identical to each other so that all the data points form an n -simplex. Therefore a linear classifier becomes a natural choice to discriminate two simplices.

4 The SCAD SVM Algorithm

Denote the objective function in (3.5) by $A(b, \mathbf{w})$. Since the hinge loss function is not differentiable at zero and the SCAD penalty is not convex in \mathbf{w} , many standard optimization

packages fail to solve (3.5). In this section, we propose an iterative algorithm to implement the SCAD SVM efficiently and show that only a series of linear equation systems need to be solved.

Sequential programming (sequential QP) algorithm is a generalization of Newton's method for unconstrained optimization in that it finds a step away from the current point by minimizing a quadratic model of the problem. Numerous optimization packages, including NPSOL, NLPQL, OPSYC, OPTIMA, MATLAB, and SQP, are found on this approach (More and Wright (1993)). For each i , we have

$$\begin{aligned} [1 - y_i(b + \mathbf{w} \cdot \mathbf{x}_i)]_+ &= \frac{1 - y_i(b + \mathbf{w} \cdot \mathbf{x}_i)}{2} + \frac{|1 - y_i(b + \mathbf{w} \cdot \mathbf{x}_i)|}{2} \\ &= \frac{1}{2} - \frac{1}{2}y_i(b + \mathbf{w} \cdot \mathbf{x}_i) + \frac{1}{2}|y_i - (b + \mathbf{w} \cdot \mathbf{x}_i)|, \end{aligned}$$

where the second equation above holds due to the fact $y_i^2 = 1$. Given an initial value (b_0, \mathbf{w}_0) close to the minimizer of $A(b, \mathbf{w})$, we consider the following local quadratic approximations:

1.

$$|y_i - (b + \mathbf{w} \cdot \mathbf{x}_i)| \approx \frac{1}{2} \frac{[y_i - (b + \mathbf{w} \cdot \mathbf{x}_i)]^2}{|y_i - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_i)|} + \frac{1}{2}|y_i - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_i)|. \quad (4.1)$$

2. When w_{j_0} is close to zero, set $\hat{w}_j = 0$; otherwise we use

$$p_\lambda(|w_j|) \approx p_\lambda(|w_{j_0}|) + \frac{1}{2} \frac{p'_\lambda(|w_{j_0}|)}{|w_{j_0}|} (w_j^2 - w_{j_0}^2). \quad (4.2)$$

It is straightforward to show that both approximating functions keep the same gradient as their original functions at the current point (b_0, \mathbf{w}_0) . Thus minimizing the local quadratic function forms assures the convergence of the algorithm towards the correct descending direction of the original function. The local quadratic approximation of the entire objective $A(b, \mathbf{w})$ is

$$\begin{aligned} A(b, \mathbf{w}) &\approx \frac{1}{2} - \frac{1}{2n} \sum_{i=1}^n y_i(b + \mathbf{w} \cdot \mathbf{x}_i) + \frac{1}{4n} \sum_{i=1}^n |y_i - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_i)| \\ &\quad + \frac{1}{4n} \sum_{i=1}^n \frac{\{y_i - (b + \mathbf{w} \cdot \mathbf{x}_i)\}^2}{|y_i - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_i)|} + \sum_{j=1}^d \left[p_\lambda(|w_{j_0}|) + \frac{p'_\lambda(|w_{j_0}|)}{2|w_{j_0}|} (w_j^2 - w_{j_0}^2) \right]. \end{aligned}$$

Removing the terms which do not involve (b, \mathbf{w}) , we get the following equivalent objective

$$\begin{aligned} \tilde{A}(b, \mathbf{w}) &\approx - \sum_{i=1}^n \frac{y_i(b + \mathbf{w} \cdot \mathbf{x}_i)}{2n} + \sum_{i=1}^n \frac{y_i(b + \mathbf{w} \cdot \mathbf{x}_i)}{|y_i - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_i)|} \\ &\quad + \frac{1}{4n} \sum_{i=1}^n \frac{(b + \mathbf{w} \cdot \mathbf{x}_i)^2}{|y_i - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_i)|} + \sum_{j=1}^d \frac{p'_\lambda(|w_{j_0}|)}{2|w_{j_0}|} w_j^2 - \frac{1}{2n}. \end{aligned}$$

Define $\mathbf{r}_0 = [y_1/|y_1 - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_1)|, \dots, y_n/|y_n - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_n)|]$,

$$\begin{aligned} D_0 &= \frac{1}{2n} \text{diag}\left\{\frac{1}{|y_i - (b_0 + \mathbf{w}_0 \cdot \mathbf{x}_i)|}, i = 1, \dots, n\right\}, \\ Q_0 &= \text{diag}\left\{0, \frac{p'_\lambda(|w_{10}|)}{|w_{10}|}, \dots, \frac{p'_\lambda(|w_{d0}|)}{|w_{d0}|}\right\}, \end{aligned}$$

$P = \frac{1}{2n}(\mathbf{y} + \mathbf{r}_0)^T X$, and $Q = X^T D_0 X + Q_0$. Then minimizing $A(b, \mathbf{w})$ is equivalent to minimizing the quadratic function

$$\tilde{A}(b, \mathbf{w}) = \frac{1}{2} \begin{pmatrix} b \\ \mathbf{w} \end{pmatrix}^T Q \begin{pmatrix} b \\ \mathbf{w} \end{pmatrix} - P \begin{pmatrix} b \\ \mathbf{w} \end{pmatrix}. \quad (4.3)$$

The solution to (4.3) satisfies the linear equation system

$$Q \begin{pmatrix} \hat{b} \\ \hat{\mathbf{w}} \end{pmatrix} = P. \quad (4.4)$$

Therefore the SCAD SVM can be implemented by the following sequential QP algorithm:

step 1: Set $k = 1$ and specify the initial value $(b^{(1)}, \mathbf{w}^{(1)})$.

step 2: Let $(b_0, \mathbf{w}_0) = (b^{(k)}, \mathbf{w}^{(k)})$. Minimize $\tilde{A}(b, \mathbf{w})$ by solving (4.4), and denote the solution as $(b^{(k+1)}, \mathbf{w}^{(k+1)})$.

step 3: Let $k = k + 1$. Go to step 2 until convergence.

If some $w_j^{(k)}$ is close to zero, say, smaller than a certain threshold, then the j th variable is regarded as redundant. Following Fan and Li (2001), we remove the j th column from the matrix X and adjust the coefficient matrix in (4.4) as well. The iteration then continues with a smaller optimization problem. The algorithm stops when there is negligible change in $(b^{(k)}, \mathbf{w}^{(k)})$. We note that the solutions of the standard SVM are good starting values. In all of our numerical examples, this algorithm converges quickly.

5 Simulations

We compare four methods: the standard SVM, the L_1 SVM, the SCAD SVM, and another classifier called distance weighted discrimination (DWD). The DWD is a large-margin classifier developed using the second-order cone programming by Marron et al. (2004), and it does not suffer from the data piling problem as the SVM. We use the OSU SVM package (www.ece.osu.edu/maj/osu-svm) to implement the SVM and the codes of Fung and Mangasarian (2004) for the L_1 SVM. Only linear SVM are considered since our primary interest is high dimensional low sample size data.

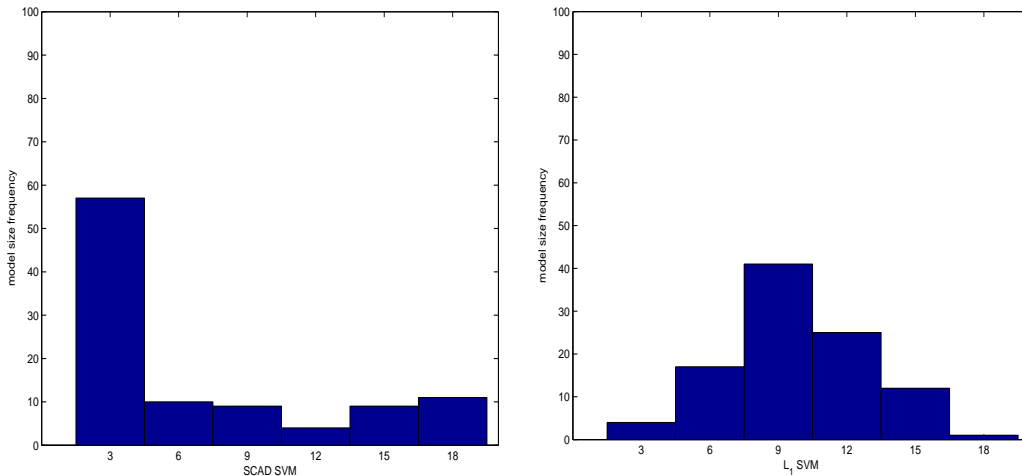
Three separate datasets are generated from the same underlying distribution for training, tuning, and testing purposes. Each method is first trained on the training set with parameters tuned using the tuning set. Then the resulted classifier is evaluated in terms of its prediction accuracy on a testing set and the number of important variables it selects correctly. The Bayes rule is also reported as the optimal rule. We conduct multiple runs for each experiment and report the average test error and the average model size given by the learning method. Fan and Li (2001) showed that the Bayes risks in regression models are not sensitive to the choice of a , and the value 3.7 has given good practical performances in various contexts. We use this value in our examples.

5.1 Example 1 - linear example

The covariate vector \mathbf{x} is 20 dimensional and generated uniformly from the unit cube $[0, 1]^{20}$. The boundary between two classes is a linear function of only first three variables: $f(\mathbf{x}) = 2x_1 + 4x_2 + 4x_3 - 4.8$. Therefore the important set is $\{x_1, x_2, x_3\}$ and the remaining seventeen variables are redundant. The points from two classes overlap at a certain degree around the boundary. The optimal error rate given by the Bayes rule is 0.2. Each of the training and tuning set contains 400 points and the testing set has 3,000 points. Table 1 summarizes the average test error and the number of variables selected by each method over 100 runs. The numbers in parentheses are the standard errors of the estimates. We observe that the SCAD SVM gives the best prediction accuracy among four methods. Overall speaking, the SCAD SVM selects a smaller number of important variables than the L_1 SVM. We also plot the histograms of the number of variables selected by the SCAD SVM and the L_1 SVM. The SCAD SVM selects the correct set $\{x_1, x_2, x_3\}$ in 54 out of 100 runs, while the L_1 SVM selects the correct set only 2 times. This shows that the SCAD SVM is able to select the correct model more frequently.

Table 1: Average test error and number of variables selected for Example 1.

	Test error	Number of selected variables
DWD	0.308 (0.040)	19.65 (0.20)
SVM	0.267 (0.015)	19.98 (0.01)
L_1 SVM	0.244 (0.017)	9.83 (0.30)
SCAD SVM	0.241 (0.027)	7.00 (0.55)



5.2 Example 2: high dimensional low sample size

we simulate a high dimension low sample size data which contains many redundant variables. This example is a modification of the example used in Weston et al. (2000). There are totally $d = 200$ variables, and only the first two are relevant. The probability of $Y = +1$ or -1 is equal. The first two variables are drawn from a mixture of Normal distributions: with probability 0.7, we have $X_1 = YN(3, 1)$ and $X_2 = N(0, 1)$; with probability 0.3, we have $X_1 = N(0, 1)$ and $X_2 = YN(3, 1)$. The remaining noise variables are generated by $X_j = N(0, 20)$, for $j = 3, \dots, 200$. We consider different settings for the training sample size: $n = 20, 30, 40, 50, 70, 100$. In each setting, we conduct thirty experiments for each method and report the average test error rate and the average number of selected variables.

Figure 3 depicts how the average test error rate changes as n increases for each method. For both the SCAD SVM and the L_1 SVM, their test errors decrease prominently compared to those of the DWD and the standard SVM. Thus it is very crucial to select important variables when too many redundant variables exist. Furthermore, the SCAD SVM consistently outperforms the L_1 SVM for all sample sizes from 20 to 100.

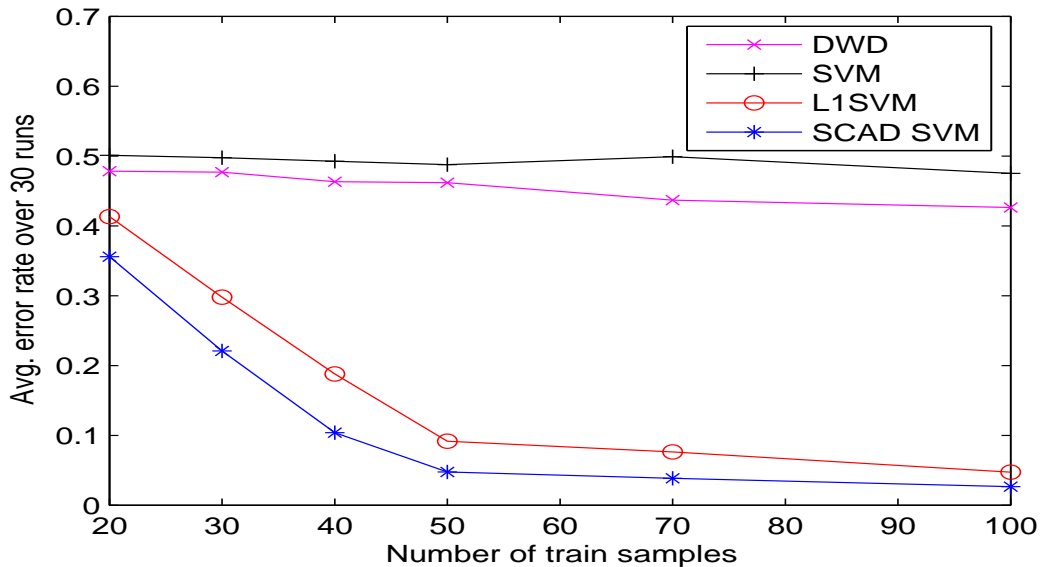


Figure 3: Average test error rate as the sample size n increases.

Table 2 shows the average number of variables selected over thirty runs for each method. Since the DWD and the standard SVM are not designed to select variables, they do not perform well in variable selection. It is observed that the SCAD SVM selects a smaller and stabler (with smaller standard errors) number of variables than the L_1 SVM in almost all cases.

Table 2: Average test error and number of variables selected for Example 2.

	n=20	n=30	n=40	n=50	n=70	n=100
DWD	99.53 (0.83)	99.83 (0.73)	102.13 (0.76)	100.23 (0.67)	98.53 (0.70)	98.97 (0.61)
SVM	64.07 (1.47)	72.67 (1.48)	81.67 (1.45)	78.93 (0.96)	82.80 (1.10)	87.10 (0.99)
L_1 SVM	15.37 (2.39)	9.10 (0.82)	12.37 (1.37)	11.17 (0.57)	12.47 (0.62)	14.03 (0.85)
SCAD SVM	8.00 (0.62)	7.90 (0.58)	7.53 (0.66)	6.47 (0.70)	4.73 (0.28)	5.27 (0.52)

In Figure 4, for one particular simulated data of size $n = 100$ from Example 3, are plotted the classification boundaries given by the Bayes rule and four methods. The symbols we used are: the DWD (dashed), the SVM (dotted), the L_1 SVM (dash-dotted), the SCAD SVM (thick solid), and the Bayes rule (thin solid). We use “o” for the points from the +1 class and

“x” for those from the -1 class. Since only X_1 and X_2 are truly relevant to the classification boundary, all the classifiers have been projected from the 200-dimensional input space to the first two-dimensional subspace. Figure 4 shows that the SCAD SVM classifier is closer to the Bayes rule than any other methods. This explains why the SCAD SVM has the smallest test error rate as shown in Figure 3.

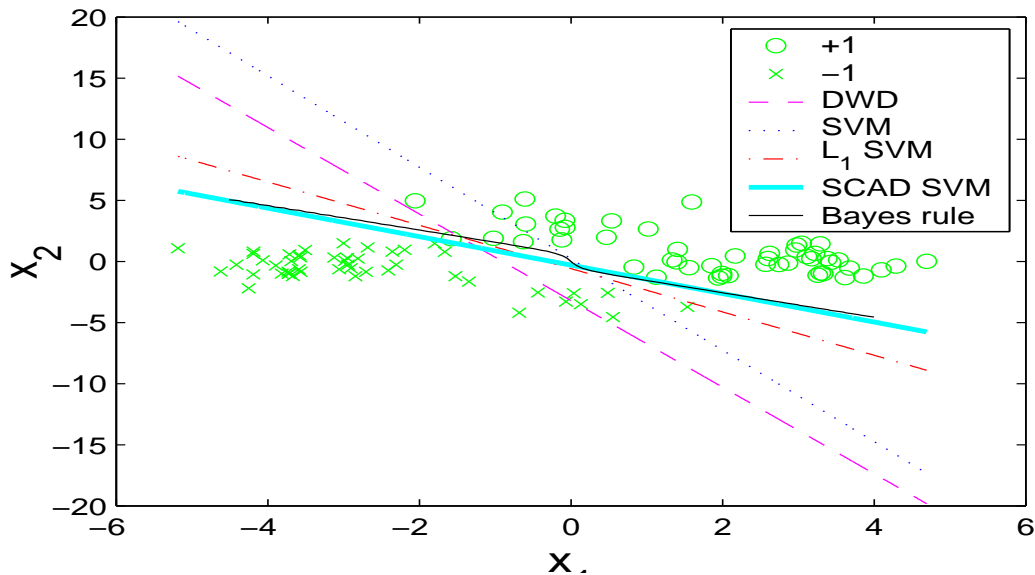


Figure 4: Bayes rule and four classification boundaries (projected onto first two dimensions) in Example 3.

6 Real Example

6.1 Gene Expression Data

In this section we use gene expression data to compare SCAD SVM with other classification methods. Three public microarray gene expression data sets are used in this section. They are from Perou et al. (2000), Veer et al. (2002), and Sotiriou et al. (2003) respectively, and for the sake of convenience we will use “Stanford”, “Rosetta”, and “Singapore” to refer them in that order. Originally the three data sets have 5,974 genes and 104 patients, 24,187 genes and 97 patients, and 7,650 genes and 99 patients, respectively. In Hu et al. (2005), three data sets are imputed for missing values, combined, and then corrected to adjust the bias since they are from three different batches. The DWD is used to the batch adjustment process; see Benito et al. (2004) for a detailed description of the systematic bias adjustment method for microarray data using the DWD. As for the gene identifier for the combined data set,

UniGene is used since it is most convenient to map the identifiers from each data set to UniGene identifier (Build 161). In case of multiple occurrences of a UCID, the median value is used.

The combined data set has 2,924 genes and totally 300 patients. Our primary interest is to select important genes and use them to classify the tissues into two different types of breast cancer. We use the source information to separate the whole data into three folds naturally. To evaluate cross-validation error of each classification rule, we train a classifier on two folds each time and test it on the remaining one. For example, the SCAD SVM is first trained on Rosetta and Singapore data, then tested on Stanford data. We refer this as the “Stanford” learning. Then we repeat this procedure for the other two learnings, the “Rosetta learning” and the “Singapore learning.” To choose the tuning parameter λ , we use ten-fold cross validation within the training set.

Table 3 shows the test error in each learning for three methods. For a fair comparison, we have not included the DWD result because it was used in the pre-processing for the combined data. Not surprisingly, the DWD has the lowest average error rate 0.086 due to the reason just described. In the Stanford learning, the SCAD SVM has the lowest misclassification rate, the SVM has the worst rate. In the Rosetta learning, both the SCAD SVM and the SVM give the same best error rate. All methods do well in the Singapore learning, with the SVM being the best and the L_1 SVM the worst. Overall, the SCAD SVM gives the lowest average error rate among the three methods.

Table 3: Cross validation error rate for breast cancer data.

	Stanford	Rosetta	Singapore	Ave.
SVM	.154	.175	.051	.127
L_1 SVM	.125	.216	.081	.141
SCAD SVM	.115	.175	.061	.117

Table 4: Number of selected genes by the L_1 and SCAD SVM.

	Stanford	Rosetta	Singapore	Ave.
L_1 SVM	59	63	72	65
SCAD SVM	15	19	31	22

Table 4 gives the number of genes selected in each learning by the SCAD SVM and the L_1 SVM. The L_1 SVM selects 59 ~ 72 genes at each learning, where the SCAD SVM only selects 15 ~ 31 genes at each case. Note that the misclassification rates of the SCAD SVM

shown in Table 3 are only based on the selected 15 ~ 31 genes, which shows the very strong gene selection power of the method. Also note that the gene selection results of both methods are consistent in the sense that they both need the smallest number of genes for prediction in the Stanford learning, and both need the largest number of genes in the Singapore learning.

Figure 5 lists the UniGene identifiers of all the genes that are selected at least three times by either the SCAD SVM or the L_1 SVM. The second and third columns show the numbers of times that the gene is selected in three learnings by the SCAD SVM and the L_1 SVM respectively. The sum of these two columns is given in the fourth column and the fifth column shows whether or not the selected gene is in the list of “intrinsic” genes, which were selected using the t-test procedure by Perou et al. (2000). The last column displays the corresponding descriptive names of UniGene identifiers.

UGid	SCAD	L1	Total	Int.	Name
Hs.169946	3	3	6	Y	GATA binding protein 3
Hs.79136	3	2	5	Y	solute carrier family 39 (metal ion transporter), member 6
Hs.80420	3	2	5	Y	chemokine (C-X3-C motif) ligand 1
Hs.1657	2	3	5	Y	estrogen receptor 1
Hs.26770	2	3	5	Y	fatty acid binding protein 7, brain
Hs.1041	2	2	4	N	v-ros UR2 sarcoma virus oncogene homolog 1 (avian)
Hs.137476	2	2	4	N	paternally expressed 10
Hs.252938	2	2	4	N	low density lipoprotein-related protein 2
Hs.298654	2	2	4	N	dual specificity phosphatase 6
Hs.369508	2	2	4	N	phosphoserine phosphatase-like
Hs.412999	2	2	4	N	cystatin A (stefin A)
Hs.9795	2	2	4	Y	acyl-Coenzyme A oxidase 2, branched chain
Hs.98998	2	2	4	Y	tenascin C (hexabrachion)
Hs.2962	2	1	3	Y	S100 calcium binding protein P
Hs.442844	2	1	3	Y	fibromodulin
Hs.75256	2	1	3	N	regulator of G-protein signalling 1
Hs.111676	1	2	3	Y	protein kinase H11
Hs.2178	1	2	3	Y	histone 2, H2be
Hs.420563	1	2	3	N	NADH dehydrogenase (ubiquinone) Fe-S protein 1, 75kDa (NADH-coenzyme Q reductase)
Hs.437638	1	2	3	Y	X-box binding protein 1
Hs.458430	1	2	3	N	N-acetyltransferase 1 (arylamine N-acetyltransferase)
Hs.89603	1	2	3	Y	mucin 1, transmembrane
Hs.91448	1	2	3	N	dual specificity phosphatase 14
Hs.191842		3	3	Y	cadherin 3, type 1, P-cadherin (placental)
Hs.437457		3	3	Y	lactotransferrin
Hs.75736		3	3	Y	apolipoprotein D
Hs.79187		3	3	N	coxsackie virus and adenovirus receptor

Figure 5: Selected genes by the SCAD SVM and the L_1 SVM.

Figure 5 shows that the top gene, Hs.169946 is selected by both methods in every learning and it is also classified as an intrinsic gene. Hs.79136 and Hs.80420, both intrinsic genes, are selected three times by the SCAD SVM but only two times by the L_1 SVM. Note that 11 genes out of total 27 selected genes from the list are not intrinsic, which suggests that one should consider the multivariate gene selection approaches, such as the SCAD SVM and the

L_1 SVM, in addition to individual gene-by-gene methods such as t-test procedures.

6.2 Metabolism Data

Metabolic datasets include the quantitative measurements of all small molecule metabolites in a biological sample. Some biological studies indicate that most of the metabolites are not informative in predicting disease or non-disease outcomes (Stitt and Fernie (2003)). Consequently, hybrid methods that incorporate variable selection with classification techniques can be very effective in analyzing datasets of this sort.

This data set is provided by Metabolon Inc and we are one of the first research groups to analyze it. The data contains the metabolic profiles of 63 samples: 32 healthy subjects and 31 subjects diagnosed with a certain disease. Within the patient group, 9 subjects are taking medication and 22 are not. For each sample, its metabolic profile contains the intensity levels of 317 compounds (metabolites). Thus this data is another large d small n example. The classification error reported for SCAD-SVM is based on the leave one out cross validation. For each training set with 62 samples, cross validation is performed within the training set to tune the parameter λ .

Table 5 shows the average leave-out-one cross validation error and the number of metabolites selected by each method. Among the four methods, the SCAD SVM gives the smallest cross validation error 0.143 while the standard SVM performs worst. Moreover, the SCAD SVM selects 18 important metabolites out of 317 and the L_1 SVM selects 32 metabolites. Therefore the SCAD SVM gives the highest classification accuracy while using the fewest metabolites. This method may have great implications on metabolic studies, since one main issue from biological aspects is to identify which metabolites are more relevant to the occurrence of a certain disease.

Table 5: Cross validation error and the number of metabolites selected for metabolism data.

	Test error	Number of selected metabolites
DWD	0.159 (0.012)	315
SVM	0.190 (0.013)	307
L_1 SVM	0.174 (0.012)	32
SCAD SVM	0.143 (0.020)	18

7 Conclusion

It is known that for high dimensional classification problems, noise features and redundant variables can affect the classifier dramatically. How to combine variable selection with classification has become an imminent problem. In this paper, we have proposed and studied a new regularization technique for simultaneous classification and variable selection. Different from the previous L_1 approaches such as Fung and Mangasarian (2004) and Zhu et al. (2003), we use a non-convex penalty function in order to achieve better variable selection and more accurate classification.

The non-convexity of the penalty function introduces greater difficulties in solving the optimization. To address this problem, we have developed an iterative procedure which utilizes the second degree approximation of the hinge loss and the penalty term around the true values of the optimization coefficients. From simulation studies and real data analysis, we have found that this iterative procedure to be both efficient and accurate. Further more, through comparative studies, this approach yields significantly better results over the standard SVM and the L_1 SVM.

References

- BENITO, M., PARKER, J., DU, Q., WU, J., XIANG, D., PEROU, C. M. and MARRON, J. S. (2004). Adjustment of systematic microarray data biases. *Bioinformatics* **20** 105–144.
- BI, J., BENNETT, K. P., EMBRECHTS, M., BRENNEMAN, C. M. and SONG, M. (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research* **3** 1229–1243.
- BOSER, B. E., GUYON, I. M. and VAPNIK, V. (1992). A training algorithm for optimal margin classifiers. *Fifth Annual ACM Workshop on Computational Learning Theory, Pittsburgh, PA* 144–152.
- BRADLEY, P. S. and MANGASARIAN, O. L. (1998). Feature selection via concave minimization and support vector machines. *Proceeding 13th International Conference on Machine Learning* 82–90 San Francisco, CA.
- CHAPLLE, O., VAPNIK, V., BOUSQUET, O. and MUKHERJEE, S. (2002). Choosing kernel parameters for support vector machines. *Machine Learning* **46** 131–159.

- FAN, J. and LI, R. Z. (2001). Variable selection via penalized likelihood. *Journal of the American Statistical Association* **96** 1348–1360.
- FUNG, G. and MANGASARIAN, O. L. (2004). A feature selection newton method for support vector machine classification. *Computational Optimization and Applications Journal* **28(2)** 185–202.
- FUREY, T., CRISTIANINI, N., DUFFY, N., BEDNARSKI, D., SCHURMMER, M. and HAUSSLER, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* **16** 906–914.
- GRANDVALET, Y. and CANU, S. (2002). Adaptive scaling for feature selection in svms. *Neural Information Processing Systems* .
- GUYON, I., WESTON, J. and BARNHILL, S. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning* **46** 389–422.
- HALL, P., MARRON, J. S. and NEEMAN, A. (2004). Geometric representation of high dimension low sample size data. *Journal of Royal Statistical Society* To appear.
- HASTIE, T., TIBSHIRANI, R. and FRIEDMAN, J. (2001). *The Elements of Statistical Learning: data mining, inference, and prediction*. Springer.
- HU, Z., FAN, C., MARRON, J. S., HE, X., QAQISH, B. F., KARACA, G., LIVASY, C., CAREY, L., REYNOLDS, E., DRESSLER, L., NOBEL, A., PARKER, J., EWEND, M. G., SAWYER, L. R., XIANG, D., WU, J., LIU, Y., KARACA, M., NANDA, R., TRETIAKOVA, M., ORRICO, A. R., DREHER, D., PALAZZO, J. P., PERREARD, L., NELSON, E., MONE, M., HANSEN, H., MULLINS, M., QUACKENBUSH, J. F., OLAPADE, O. I., BERNARD, B. S. and PEROU, C. M. (2005). The molecular portraits of breast tumors are conserved across microarray platforms Submitted.
- KITTLER, J. (1986). Feature selection and extraction. In *T.Y.Young and K.-S. Fu, editors, Handbook of Pattern Recognition and Image Processing*. Academic Press, New York .
- LIN, Y. (2002). SVM and the Bayes rule in classification. *Data Mining and Knowledge Discovery* **6** 259–275.
- MARRON, J. S., TODD, M. and AHN, J. (2004). Distance weighted discrimination. Under revision.
- MORE, J. J. and WRIGHT, S. J. (1993). *Optimization Software Guide*. SIAM.

- PEROU, C., SRLIE, T., EISEN, M., VAN DE RIJN, M., JEFFREY, S., REES, C., POLLACK, J., ROSS, D., JOHNSEN, H., AKSLEN, L., FLUGE, ., PERGAMENSCHIKOV, A., WILLIAMS, C., ZHU, S., LNING, P., BRRESEN-DALE, A., BROWN, P. and BOTSTEIN, D. (2000). Molecular portraits of human breast tumors. *Nature* **406** 747–752.
- RAKOTOMAMONJY, A. (2003). Variable selection using svm-based criteria. *Journal of Machine Learning Research* **3** 1357–1370.
- SOTIRIOU, C., NEO, S., MCSHANE, L., KORN, E., LONG, P., JAZAERI, A., MARTIAT, P., FOX, S., HARRIS, A. and LIU, E. (2003). Breast cancer classification and prognosis based on gene expression profiles from a population-based study. *Proceedings of the National Academy of Sciences* **100(18)** 10393–10398.
- STITT, M. and FERNIE, A. R. (2003). From measurements of metabolites to metabolomics: an “on the fly” perspective illustrated by recent studies of carbon-nitrogen interactions. *Current Opinion in Biotechnology* **14** 136–144.
- TIBSHIRANI, R. J. (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society, B* **58** 267–288.
- VAPNIK, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.
- VAPNIK, V. N. (1998). *Statistical Learning Theory*. Wiley.
- VEER, L. V., DAI, H., VAN DE VIJVER, M., HE, Y., HART, A., MAO, M., PETERSE, H., VAN DER KOOY, K., MARTON, M., WITTEVEEN, A., SCHREIBER, G., KERKHOVEN, R., ROBERTS, C., LINSLEY, P., BERNARDS, R. and FRIEND, S. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature* **415** 530–536.
- WESTON, J., MUKHERJEE, S., CHAPELLE, O., PONTIL, M., POGGIO, T. and VAPNIK, V. (2000). Feature selection for svms. *Advances in Neural Information Processing Systems* **13** 668–674.
- ZHU, J., ROSSET, S., HASTIE, T. and TIBSHIRANI, R. (2003). 1-norm support vector machines. *Neural Information Processing Systems* **16**.