Journal of Machine Learning Research 5 (2004) 1391–1415 Submitted 3/04; Published 10/04

The Entire Regularization Path for the Support Vector Machine

Trevor Hastie HASTIE@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Saharon Rosset SROSSET@US.IBM.COM
IBM Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598, USA

Robert Tibshirani TIBS@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Ji Zhu JIZHU@UMICH.EDU
Department of Statistics
University of Michigan
439 West Hall
550 East University
Ann Arbor, MI 48109-1092, USA

Editor: Nello Cristianini

Abstract

The support vector machine (SVM) is a widely used tool for classification. Many efficient implementations exist for fitting a two-class SVM model. The user has to supply values for the tuning parameters: the regularization cost parameter, and the kernel parameters. It seems a common practice is to use a default value for the cost parameter, often leading to the least restrictive model. In this paper we argue that the choice of the cost parameter can be critical. We then derive an algorithm that can fit the entire path of SVM solutions for every value of the cost parameter, with essentially the same computational cost as fitting one SVM model. We illustrate our algorithm on some examples, and use our representation to give further insight into the range of SVM solutions.

Keywords: support vector machines, regularization, coefficient path

1. Introduction

In this paper we study the support vector machine (SVM)(Vapnik, 1996; Schölkopf and Smola, 2001) for two-class classification. We have a set of n training pairs x_i, y_i, where $x_i \in \mathbb{R}^p$ is a p-vector of real-valued predictors (attributes) for the ith observation, and $y_i \in \{-1, +1\}$ codes its binary response. We start off with the simple case of a linear classifier, where our goal is to estimate a linear decision function

$$f(x) = \beta_0 + \beta^T x, \tag{1}$$

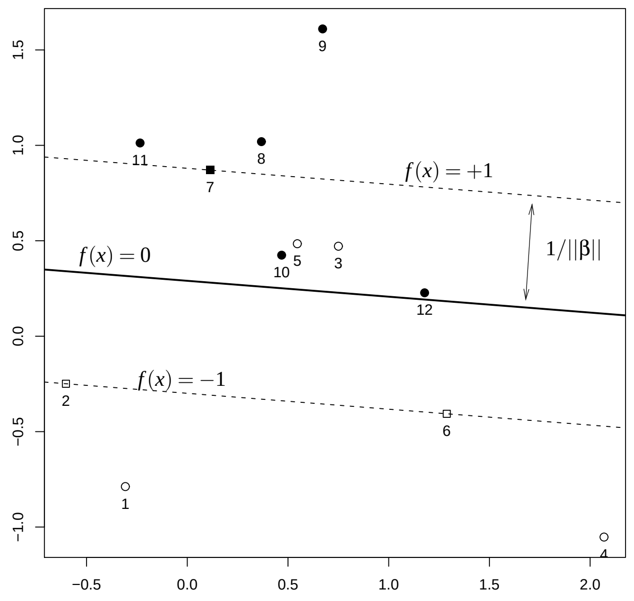


Figure 1: A simple example shows the elements of a SVM model. The "+1" points are solid, the "-1" hollow. $C = 2$, and the width of the soft margin is $2/||\beta|| = 2 \times 0.587$. Two hollow points $\{3,5\}$ are misclassified, while the two solid points $\{10,12\}$ are correctly classified, but on the wrong side of their margin $f(x) = +1$; each of these has $\xi_i > 0$. The three square shaped points $\{2,6,7\}$ are exactly on the margin.

and its associated classifier

$$\text{Class}(x) = \text{sign}[f(x)]. \tag{2}$$

There are many ways to fit such a linear classifier, including linear regression, Fisher's linear discriminant analysis, and logistic regression (Hastie et al., 2001, Chapter 4). If the training data are linearly separable, an appealing approach is to ask for the decision boundary $\{x : f(x) = 0\}$ that maximizes the margin between the two classes (Vapnik, 1996). Solving such a problem is an exercise in convex optimization; the popular setup is

$$\min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 \text{ subject to, for each i: } y_i(\beta_0 + x_i^T\beta) \geq 1. \tag{3}$$

A bit of linear algebra shows that $\frac{1}{||\beta||}(\beta_0 + x_i^T\beta)$ is the signed distance from x_i to the decision boundary. When the data are not separable, this criterion is modified to

$$\min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n}\xi_i, \tag{4}$$

$$\text{subject to, for each } i\text{: } y_i(\beta_0 + x_i^T\beta) \quad \geq \quad 1 - \xi_i.$$

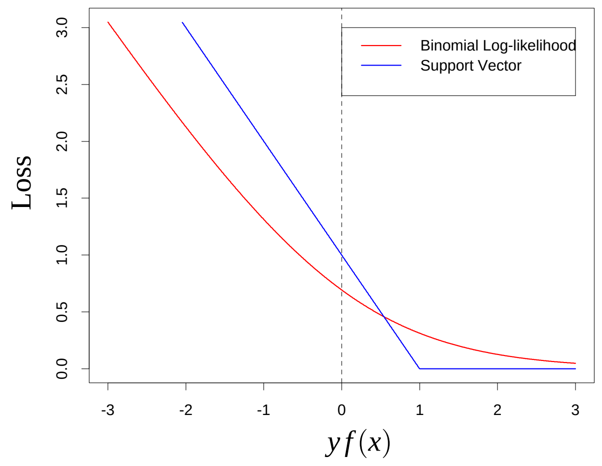


Figure 2: The hinge loss penalizes observation margins $yf(x)$ less than $+1$ linearly, and is indifferent to margins greater than $+1$. The negative binomial log-likelihood (deviance) has the same asymptotes, but operates in a smoother fashion near the *elbow* at $yf(x) = 1$.

Here the ξ_i are non-negative slack variables that allow points to be on the wrong side of their "soft margin" ($f(x) = \pm 1$), as well as the decision boundary, and C is a cost parameter that controls the amount of overlap. Figure 1 shows a simple example. If the data are separable, then for sufficiently large C the solutions to (3) and (4) coincide. If the data are not separable, as C gets large the solution approaches the minimum overlap solution with largest margin, which is attained for some finite value of C.

Alternatively, we can formulate the problem using a *Loss* + *Penalty* criterion (Wahba et al., 2000; Hastie et al., 2001):

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} [1 - y_i(\beta_0 + \beta^T x_i)]_+ + \frac{\lambda}{2}||\beta||^2. \tag{5}$$

The regularization parameter λ in (5) corresponds to $1/C$, with C in (4). Here the *hinge loss* $L(y, f(x)) = [1 - yf(x)]_+$ can be compared to the negative binomial log-likelihood $L(y, f(x)) = \log[1 + \exp(-yf(x))]$ for estimating the linear function $f(x) = \beta_0 + \beta^T x$; see Figure 2.

This formulation emphasizes the role of regularization. In many situations we have sufficient variables (e.g. gene expression arrays) to guarantee separation. We may nevertheless avoid the maximum margin separator ($\lambda \downarrow 0$), which is governed by observations on the boundary, in favor of a more regularized solution involving more observations.

This formulation also admits a class of more flexible, nonlinear generalizations

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda J(f), \tag{6}$$

where $f(x)$ is an arbitrary function in some Hilbert space $\mathcal{H}$, and $J(f)$ is a functional that measures the "roughness" of f in $\mathcal{H}$.

The nonlinear *kernel* SVMs arise naturally in this context. In this case $f(x) = \beta_0 + g(x)$, and $J(f) = J(g)$ is a norm in a Reproducing Kernel Hilbert Space of functions $\mathcal{H}_K$ generated by a

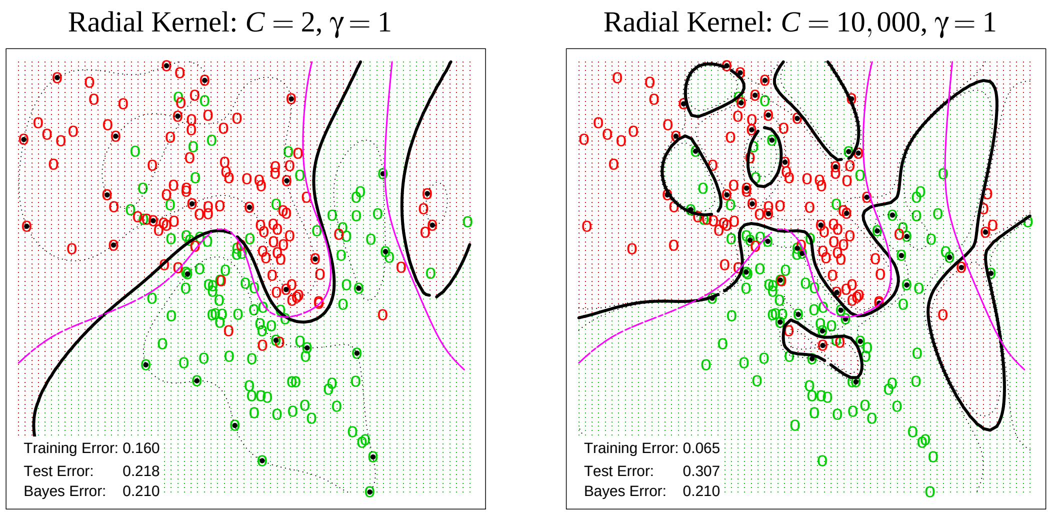


Figure 3: Simulated data illustrate the need for regularization. The 200 data points are generated from a pair of mixture densities. The two SVM models used radial kernels with the scale and cost parameters as indicated at the top of the plots. The thick black curves are the decision boundaries, the dotted curves the margins. The less regularized fit on the right overfits the training data, and suffers dramatically on test error. The broken purple curve is the optimal Bayes decision boundary.

positive-definite kernel $K(x,x')$. By the well-studied properties of such spaces (Wahba, 1990; Evgeniou et al., 1999), the solution to (6) is finite dimensional (even if $\mathcal{H}_K$ is infinite dimensional), in this case with a representation $f(x) = \beta_0 + \sum_{i=1}^n \theta_i K(x,x_i)$. Consequently (6) reduces to the finite form

$$\min_{\beta_0,\theta} \sum_{i=1}^n L[y_i, \beta_0 + \sum_{j=1}^n \theta_i K(x_i,x_j)] + \frac{\lambda}{2} \sum_{j=1}^n \sum_{j'=1}^n \theta_j \theta_{j'} K(x_j, x'_j). \tag{7}$$

With L the hinge loss, this is an alternative route to the kernel SVM; see Hastie et al. (2001) for more details.

It seems that the regularization parameter C (or λ) is often regarded as a genuine “nuisance” in the community of SVM users. Software packages, such as the widely used *SVM^{light}* (Joachims, 1999), provide default settings for C, which are then used without much further exploration. A recent introductory document (Hsu et al., 2003) supporting the `LIBSVM` package does encourage grid search for C.

Figure 3 shows the results of fitting two SVM models to the same simulated data set. The data are generated from a pair of mixture densities, described in detail in Hastie et al. (2001, Chapter 2).[1] The radial kernel function $K(x,x') = \exp(-\gamma||x-x'||^2)$ was used, with $\gamma = 1$. The model on the left is more regularized than that on the right ($C = 2$ vs $C = 10,000$, or $\lambda = 0.5$ vs $\lambda = 0.0001$), and

1. The actual training data and test distribution are available from `http:// www-stat.stanford.edu/ElemStatLearn`.

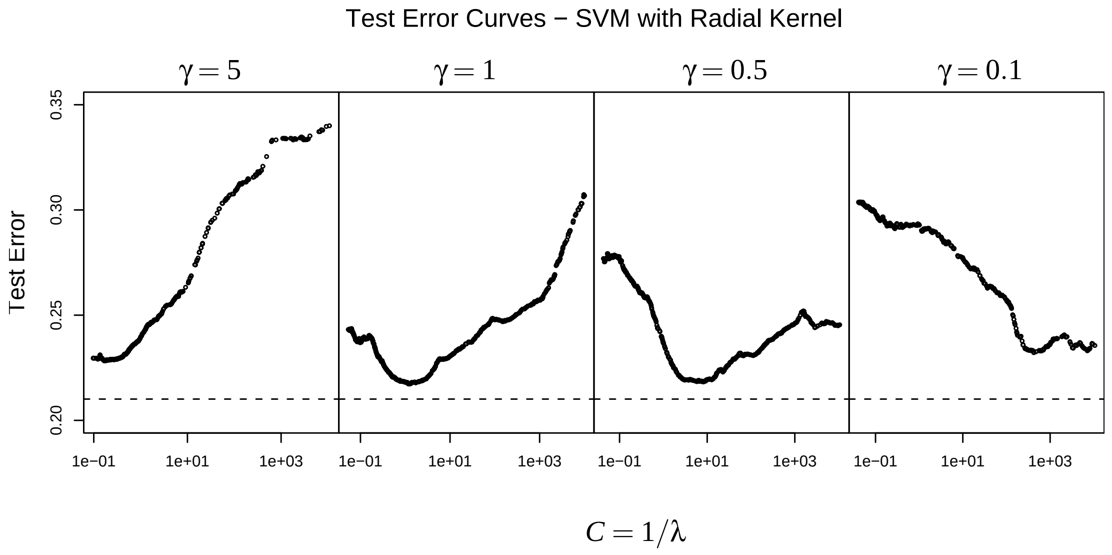


Figure 4: Test error curves for the mixture example, using four different values for the radial kernel parameter γ. Small values of C correspond to heavy regularization, large values of C to light regularization. Depending on the value of γ, the optimal C can occur at either end of the spectrum or anywhere in between, emphasizing the need for careful selection.

performs much better on test data. For these examples we evaluate the test error by integration over the lattice indicated in the plots.

Figure 4 shows the test error as a function of C for these data, using four different values for the kernel scale parameter γ. Here we see a dramatic range in the correct choice for C (or $\lambda = 1/C$); when $\gamma = 5$, the most regularized model is called for, and we will see in Section 6 that the SVM is really performing kernel density classification. On the other hand, when $\gamma = 0.1$, we would want to choose among the least regularized models.

One of the reasons that investigators avoid extensive exploration of C is the computational cost involved. In this paper we develop an algorithm which fits the *entire path* of SVM solutions $[\beta_0(C), \beta(C)]$, for all possible values of C, with essentially the computational cost of fitting a single model for a particular value of C. Our algorithm exploits the fact that the Lagrange multipliers implicit in (4) are piecewise-linear in C. This also means that the coefficients $\beta(C)$ are also piecewise-linear in C. This is true for all SVM models, both linear and nonlinear kernel-based SVMs. Figure 8 on page 1406 shows these Lagrange paths for the mixture example. This work was inspired by the related “Least Angle Regression” (LAR) algorithm for fitting LASSO models (Efron et al., 2004), where again the coefficient paths are piecewise linear.

These speedups have a big impact on the estimation of the accuracy of the classifier, using a validation dataset (e.g. as in K-fold cross-validation). We can rapidly compute the fit for each test data point for any and all values of C, and hence the generalization error for the entire validation set as a function of C.

In the next section we develop our algorithm, and then demonstrate its capabilities on a number of examples. Apart from offering dramatic computational savings when computing multiple solutions (Section 4.3), the nature of the path, in particular at the boundaries, sheds light on the action of the kernel SVM (Section 6).

2. Problem Setup

We use a criterion equivalent to (4), implementing the formulation in (5):

$$\begin{aligned} &\min_{\beta,\beta_0} \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\beta^T\beta \\ &\text{subject to } 1 - y_i f(x_i) \le \xi_i;\ \xi_i \ge 0;\ f(x) = \beta_0 + \beta^T x. \end{aligned} \tag{8}$$

Initially we consider only linear SVMs to get the intuitive flavor of our procedure; we then generalize to kernel SVMs.

We construct the Lagrange primal function

$$L_P: \quad \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\beta^T\beta + \sum_{i=1}^{n} \alpha_i(1 - y_i f(x_i) - \xi_i) - \sum_{i=1}^{n} \gamma_i \xi_i \tag{9}$$

and set the derivatives to zero. This gives

$$\frac{\partial}{\partial\beta}: \quad \beta = \frac{1}{\lambda}\sum_{i=1}^{n} \alpha_i y_i x_i, \tag{10}$$

$$\frac{\partial}{\partial\beta_0}: \quad \sum_{i=1}^{n} y_i \alpha_i = 0, \tag{11}$$

$$\frac{\partial}{\partial\xi_i}: \quad \alpha_i = 1 - \gamma_i, \tag{12}$$

along with the KKT conditions

$$\alpha_i(1 - y_i f(x_i) - \xi_i) = 0, \tag{13}$$

$$\gamma_i \xi_i = 0. \tag{14}$$

We see that $0 \le \alpha_i \le 1$, with $\alpha_i = 1$ when $\xi_i > 0$ (which is when $y_i f(x_i) < 1$). Also when $y_i f(x_i) > 1$, $\xi_i = 0$ since no cost is incurred, and $\alpha_i = 0$. When $y_i f(x_i) = 1$, α_i can lie between 0 and 1.[2]

We wish to find the entire solution path for all values of $\lambda \ge 0$. The basic idea of our algorithm is as follows. We start with λ large and decrease it toward zero, keeping track of all the events that occur along the way. As λ decreases, $||\beta||$ increases, and hence the width of the margin decreases (see Figure 1). As this width decreases, points move from being inside to outside the margin. Their corresponding α_i change from $\alpha_i = 1$ when they are inside the margin ($y_i f(x_i) < 1$) to $\alpha_i = 0$ when they are outside the margin ($y_i f(x_i) > 1$). By continuity, points must linger on the margin ($y_i f(x_i) = 1$) while their α_i decrease from 1 to 0. We will see that the $\alpha_i(\lambda)$ trajectories are piecewise-linear in λ, which affords a great computational savings: as long as we can establish the break points, all

2. For readers more familiar with the traditional SVM formulation (4), we note that there is a simple connection between the corresponding Lagrange multipliers, $\alpha'_i = \alpha_i/\lambda = C\alpha_i$, and hence in that case $\alpha'_i \in [0, C]$. We prefer our formulation here since our $\alpha_i \in [0, 1]$, and this simplifies the definition of the paths we define.

values in between can be found by simple linear interpolation. Note that points can return to the margin, after having passed through it.

It is easy to show that if the $\alpha_i(\lambda)$ are piecewise linear in λ, then both $\alpha'_i(C) = C\alpha_i(C)$ and $\beta(C)$ are piecewise linear in C. It turns out that $\beta_0(C)$ is also piecewise linear in C. We will frequently switch between these two representations.

We denote by $\mathcal{I}_+$ the set of indices corresponding to $y_i = +1$ points, there being $n_+ = |\mathcal{I}_+|$ in total. Likewise for $\mathcal{I}_-$ and n_-. Our algorithm keeps track of the following sets (with names inspired by the hinge loss function in Figure 2):

- $\mathcal{E} = \{i : y_i f(x_i) = 1,\ 0 \le \alpha_i \le 1\}$, $\mathcal{E}$ for Elbow,
- $\mathcal{L} = \{i : y_i f(x_i) < 1,\ \alpha_i = 1\}$, $\mathcal{L}$ for Left of the elbow,
- $\mathcal{R} = \{i : y_i f(x_i) > 1,\ \alpha_i = 0\}$, $\mathcal{R}$ for Right of the elbow.

3. Initialization

We need to establish the initial state of the sets defined above. When λ is very large (∞), from (10) $\beta = 0$, and the initial values of β_0 and the α_i depend on whether $n_- = n_+$ or not. If the classes are balanced, one can directly find the initial configuration by finding the most extreme points in each class. We will see that when $n_- \neq n_+$, this is no longer the case, and in order to satisfy the constraint (11), a quadratic programming algorithm is needed to obtain the initial configuration.

In fact, our `SvmPath` algorithm can be started at any intermediate solution of the SVM optimization problem (i.e. the solution for any λ), since the values of α_i and $f(x_i)$ determine the sets $\mathcal{L}$, $\mathcal{E}$ and $\mathcal{R}$. We will see in Section 6 that if there is no intercept in the model, the initialization is again trivial, no matter whether the classes are balanced or not. We have prepared some MPEG movies to illustrate the two special cases detailed below. The movies can be downloaded at the web site `http://www-stat.stanford.edu/~hastie/Papers/svm/MOVIE/`.

3.1 Initialization: $n_- = n_+$

Lemma 1 *For λ sufficiently large, all the $\alpha_i = 1$. The initial $\beta_0 \in [-1, 1]$ — any value gives the same loss $\sum_{i=1}^n \xi_i = n_+ + n_-$.*

Proof Our proof relies on the criterion and the KKT conditions in Section 2. Since $\beta = 0$, $f(x) = \beta_0$. To minimize $\sum_{i=1}^n \xi_i$, we should clearly restrict β_0 to $[-1, 1]$. For $\beta_0 \in (-1, 1)$, all the $\xi_i > 0$, $\gamma_i = 0$ in (12), and hence $\alpha_i = 1$. Picking one of the endpoints, say $\beta_0 = -1$, causes $\alpha_i = 1$, $i \in \mathcal{I}_+$, and hence also $\alpha_i = 1$, $i \in \mathcal{I}_-$, for (11) to hold. ■

We also have that for these early and large values of λ

$$\beta = \frac{1}{\lambda}\beta^* \text{ where } \beta^* = \sum_{i=1}^n y_i x_i. \tag{15}$$

Now in order that (11) remain satisfied, we need that one or more positive and negative examples hit the elbow *simultaneously*. Hence as λ decreases, we require that $\forall i\ y_i f(x_i) \le 1$ or

$$y_i \left[\frac{\beta^{*T} x_i}{\lambda} + \beta_0\right] \le 1 \tag{16}$$

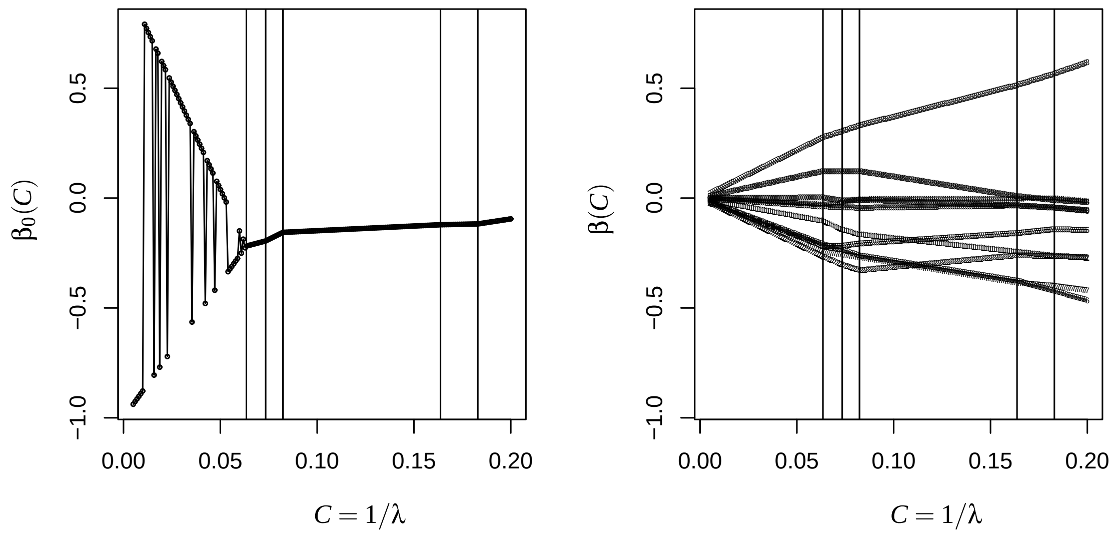


Figure 5: The initial paths of the coefficients in a small simulated dataset with $n_- = n_+$. We see the zone of allowable values for β_0 shrinking toward a fixed point (20). The vertical lines indicate the breakpoints in the piecewise linear coefficient paths.

or

$$\beta_0 \leq 1 - \frac{{\beta^*}^T x_i}{\lambda} \quad \text{for all } i \in I_+ \tag{17}$$

$$\beta_0 \geq -1 - \frac{{\beta^*}^T x_i}{\lambda} \quad \text{for all } i \in I_-. \tag{18}$$

Pick $i_+ = \arg\max_{i \in I_+} {\beta^*}^T x_i$ and $i_- = \arg\min_{i \in I_-} {\beta^*}^T x_i$ (for simplicity we assume that these are unique). Then at this point of entry and beyond for a while we have $\alpha_{i_+} = \alpha_{i_-}$, and $f(x_{i_+}) = 1$ and $f(x_{i_-}) = -1$. This gives us two equations to solve for the initial point of entry λ_0 and β_0, with solutions

$$\lambda_0 = \frac{{\beta^*}^T x_{i_+} - {\beta^*}^T x_{i_-}}{2}, \tag{19}$$

$$\beta_0 = -\left(\frac{{\beta^*}^T x_{i_+} + {\beta^*}^T x_{i_-}}{{\beta^*}^T x_{i_+} - {\beta^*}^T x_{i_-}} \right). \tag{20}$$

Figure 5 (left panel) shows a trajectory of $\beta_0(C)$ as a function of C, for a small simulated data set. These solutions were computed directly using a quadratic-programming algorithm, using a

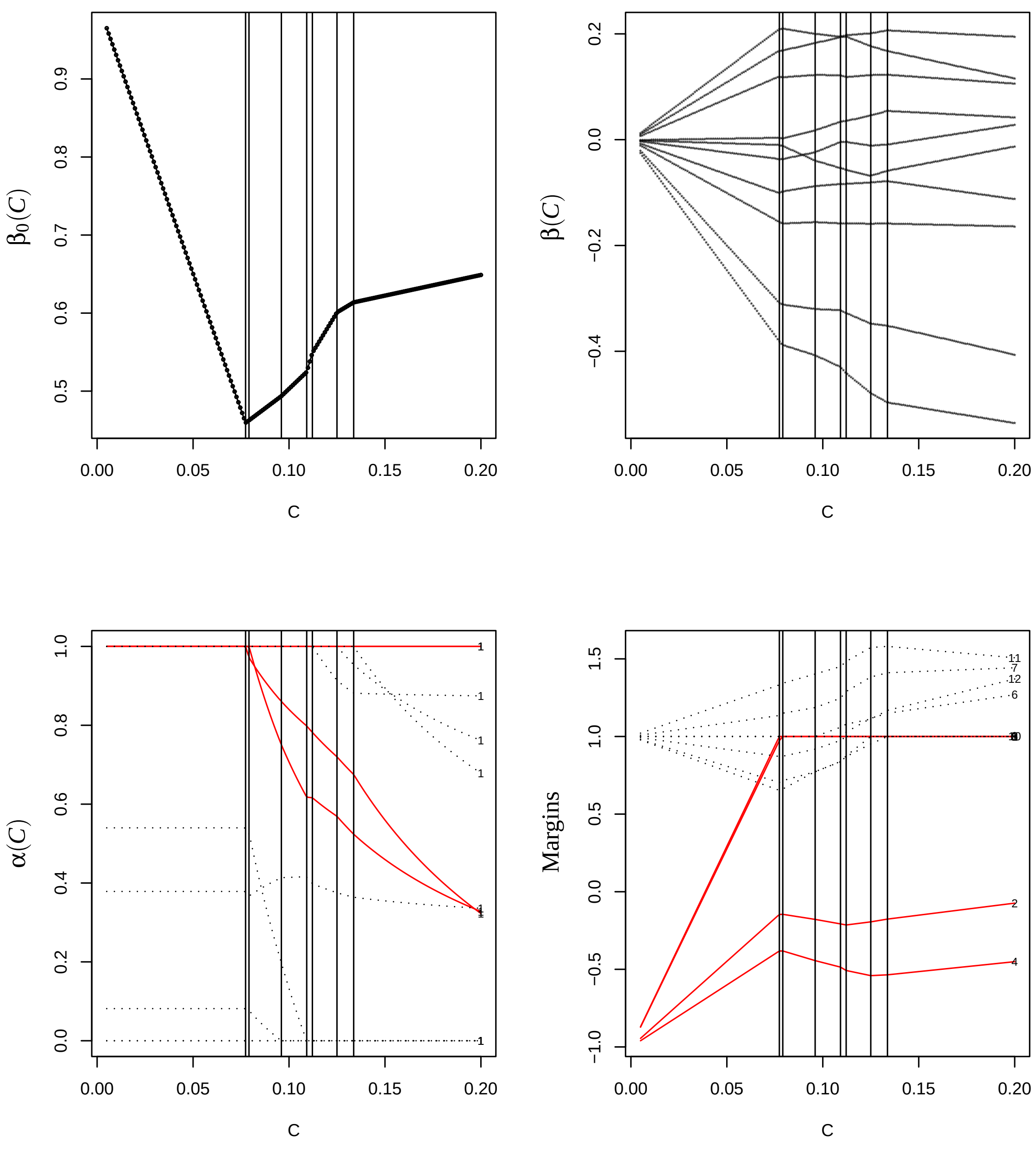


Figure 6: The initial paths of the coefficients in a case where $n_- < n_+$. All the n_- points are misclassified, and start off with a margin of -1. The α_i^* remain constant until one of the points in I_- reaches the margin. The vertical lines indicate the breakpoints in the piecewise linear $\beta(C)$ paths. Note that the $\alpha_i(C)$ are *not* piecewise linear in C, but rather in $\lambda = 1/C$.

predefined grid of values for λ. The arbitrariness of the initial values is indicated by the zig-zag nature of this path. The breakpoints were found using our exact-path algorithm.

3.2 Initialization: $n_+ > n_-$

In this case, when $\beta = 0$, the optimal choice for β_0 is 1, and the loss is $\sum_{i=1}^n \xi_i = n_-$. However, we also require that (11) holds.

Lemma 2 *With* $\beta^*(\alpha) = \sum_{i=1}^n y_i \alpha_i x_i$, *let*

$$\{\alpha_i^*\} = \arg\min_{\alpha} ||\beta^*(\alpha)||^2 \tag{21}$$

$$s.t.\ \alpha_i \in [0,1] \text{ for } i \in I_+,\ \alpha_i = 1 \text{ for } i \in I_-, \text{ and } \sum_{i \in I_+} \alpha_i = n_- \tag{22}$$

Then for some λ_0 *we have that for all* $\lambda > \lambda_0$, $\alpha_i = \alpha_i^*$, *and* $\beta = \beta^*/\lambda$, *with* $\beta^* = \sum_{i=1}^n y_i \alpha_i^* x_i$.

Proof The Lagrange dual corresponding to (9) is obtained by substituting (10)–(12) into (9) (Hastie et al., 2001, Equation 12.13):

$$L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2\lambda} \sum_{i=1}^n \sum_{i'=1}^n \alpha_i \alpha_{i'} y_i y_{i'} x_i x_{i'}. \tag{23}$$

Since we start with $\beta = 0$, $\beta_0 = 1$, all the I_- points are misclassified, and hence we will have $\alpha_i = 1 \forall i \in I_-$, and hence from (11) $\sum_{i=1}^n \alpha_i = 2n_-$. This latter sum will remain $2n_-$ for a while as β grows away from zero. This means that during this phase, the first term in the Lagrange dual is constant; the second term is equal to $-\frac{1}{2\lambda}||\beta^*(\alpha)||^2$, and since we maximize the dual, this proves the result. ■

We now establish the "starting point" λ_0 and β_0 when the α_i start to change. Let β^* be the fixed coefficient direction corresponding to α_i^* (as in (15)):

$$\beta^* = \sum_{i=1}^n \alpha_i^* y_i x_i. \tag{24}$$

There are two possible scenarios:

1. There exist two or more elements in I_+ with $0 < \alpha_i^* < 1$, or
2. $\alpha_i^* \in \{0,1\}\ \forall i \in I_+$.

Consider the first scenario (depicted in Figure 6), and suppose $\alpha_{i_+}^* \in (0,1)$ (on the margin). Let $i_- = \arg\min_{i \in I_-} \beta^{*T} x_i$. Then since the point i_+ remains on the margin until an I_- point reaches its margin, we can find

$$\lambda_0 = \frac{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}}{2}, \tag{25}$$

identical in form to to (19), as is the corresponding β_0 to (20).

For the second scenario, it is easy to see that we find ourselves in the same situation as in Section 3.1—a point from I_- and one of the points in I_+ with $\alpha_i^* = 1$ must reach the margin simultaneously. Hence we get an analogous situation, except with $i_+ = \arg\max_{i \in I_+^1} \beta^{*T} x_i$, where I_+^1 is the subset of I_+ with $\alpha_i^* = 1$.

3.3 Kernels

The development so far has been in the original feature space, since it is easier to visualize. It is easy to see that the entire development carries through with "kernels" as well. In this case $f(x) = \beta_0 + g(x)$, and the only change that occurs is that (10) is changed to

$$g(x_i) = \frac{1}{\lambda}\sum_{j=1}^{n} \alpha_j y_j K(x_i, x_j),\ i = 1, \ldots, n, \tag{26}$$

or $\theta_j(\lambda) = \alpha_j y_j/\lambda$ using the notation in (7).

Our initial conditions are defined in terms of expressions $\beta^{*T} x_{i_+}$, for example, and again it is easy to see that the relevant quantities are

$$g^*(x_{i^+}) = \sum_{j=1}^{n} \alpha_j^* y_j K(x_{i_+}, x_j), \tag{27}$$

where the α_i^* are all 1 in Section 3.1, and defined by Lemma 2 in Section 3.2.

Hereafter we will develop our algorithm for this more general kernel case.

4. The Path

The algorithm hinges on the set of points $\mathcal{E}$ sitting at the elbow of the loss function — i.e on the margin. These points have $y_i f(x_i) = 1$ and $\alpha_i \in [0,1]$. These are distinct from the points $\mathcal{R}$ to the right of the elbow, with $y_i f(x_i) > 1$ and $\alpha_i = 0$, and those points $\mathcal{L}$ to the left with $y_i f(x_i) < 1$ and $\alpha_i = 1$. We consider this set at the point that an event has occurred. The event can be either:

1. The initial event, which means 2 or more points start at the elbow, with their initial values of $\alpha \in [0,1]$.
2. A point from $\mathcal{L}$ has just entered $\mathcal{E}$, with its value of α_i initially 1.
3. A point from $\mathcal{R}$ has reentered $\mathcal{E}$, with its value of α_i initially 0.
4. One or more points in $\mathcal{E}$ has left the set, to join either $\mathcal{R}$ or $\mathcal{L}$.

Whichever the case, for continuity reasons this set will stay stable until the next event occurs, since to pass through $\mathcal{E}$, a point's α_i must change from 0 to 1 or vice versa. Since all points in $\mathcal{E}$ have $y_i f(x_i) = 1$, we can establish a path for their α_i.

Event 4 allows for the possibility that $\mathcal{E}$ becomes empty while $\mathcal{L}$ is not. If this occurs, then the KKT condition (11) implies that $\mathcal{L}$ is balanced w.r.t. +1s and -1s, and we resort to the initial condition as in Section 3.1.

We use the subscript ℓ to index the sets above immediately after the ℓth event has occurred. Suppose $|\mathcal{E}_\ell| = m$, and let α_i^ℓ, β_0^ℓ and λ_ℓ be the values of these parameters at the point of entry. Likewise f^ℓ is the function at this point. For convenience we define $\alpha_0 = \lambda\beta_0$, and hence $\alpha_0^\ell = \lambda_\ell \beta_0^\ell$.

Since

$$f(x) = \frac{1}{\lambda}\left(\sum_{j=1}^{n} y_j \alpha_j K(x, x_j) + \alpha_0\right), \tag{28}$$

for $\lambda_\ell > \lambda > \lambda_{\ell+1}$ we can write

$$\begin{aligned} f(x) &= \left[f(x) - \frac{\lambda_\ell}{\lambda} f^\ell(x)\right] + \frac{\lambda_\ell}{\lambda} f^\ell(x) \\ &= \frac{1}{\lambda}\left[\sum_{j\in\mathcal{E}_\ell}(\alpha_j - \alpha_j^\ell)y_j K(x,x_j) + (\alpha_0 - \alpha_0^\ell) + \lambda_\ell f^\ell(x)\right]. \end{aligned} \tag{29}$$

The second line follows because all the observations in $\mathcal{L}_\ell$ have their $\alpha_i = 1$, and those in $\mathcal{R}_\ell$ have their $\alpha_i = 0$, for this range of λ. Since each of the m points $x_i \in \mathcal{E}_\ell$ are to stay at the elbow, we have that

$$\frac{1}{\lambda}\left[\sum_{j\in\mathcal{E}_\ell}(\alpha_j - \alpha_j^\ell)y_i y_j K(x_i,x_j) + y_i(\alpha_0 - \alpha_0^\ell) + \lambda_\ell\right] = 1,\ \forall i \in \mathcal{E}_\ell. \tag{30}$$

Writing $\delta_j = \alpha_j^\ell - \alpha_j$, from (30) we have

$$\sum_{j\in\mathcal{E}_\ell}\delta_j y_i y_j K(x_i,x_j) + y_i\delta_0 = \lambda_\ell - \lambda,\ \forall i \in \mathcal{E}_\ell. \tag{31}$$

Furthermore, since at all times $\sum_{i=1}^n y_i\alpha_i = 0$, we have that

$$\sum_{j\in\mathcal{E}_\ell} y_j\delta_j = 0. \tag{32}$$

Equations (31) and (32) constitute $m+1$ linear equations in $m+1$ unknowns δ_j, and can be solved.

Denoting by $\mathbf{K}_\ell^*$ the $m\times m$ matrix with ijth entry $y_i y_j K(x_i,x_j)$ for i and j in $\mathcal{E}_\ell$, we have from (31) that

$$\mathbf{K}_\ell^*\delta + \delta_0\mathbf{y}_\ell = (\lambda_\ell - \lambda)\mathbf{1}, \tag{33}$$

where $\mathbf{y}_\ell$ is the m vector with entries $y_i,\ i \in \mathcal{E}_\ell$. From (32) we have

$$\mathbf{y}_\ell^T\delta = 0. \tag{34}$$

We can combine these two into one matrix equation as follows. Let

$$\mathbf{A}_\ell = \begin{pmatrix} 0 & \mathbf{y}_\ell{}^T \\ \mathbf{y}_\ell & \mathbf{K}_\ell^* \end{pmatrix},\quad \delta^a = \begin{pmatrix}\delta_0 \\ \delta\end{pmatrix},\ \text{and}\quad \mathbf{1}^a = \begin{pmatrix} 0 \\ \mathbf{1}\end{pmatrix}, \tag{35}$$

then (34) and (33) can be written

$$\mathbf{A}_\ell\delta^a = (\lambda_\ell - \lambda)\mathbf{1}^a. \tag{36}$$

If $\mathbf{A}_\ell$ has full rank, then we can write

$$\mathbf{b}^a = \mathbf{A}_\ell{}^{-1}\mathbf{1}^a, \tag{37}$$

and hence

$$\alpha_j = \alpha_j^\ell - (\lambda_\ell - \lambda)b_j,\ j \in \{0\}\cup\mathcal{E}_\ell. \tag{38}$$

Hence for $\lambda_{\ell+1} < \lambda < \lambda_\ell$, the α_j for points at the elbow proceed *linearly in* λ. From (29) we have

$$f(x) = \frac{\lambda_\ell}{\lambda}\left[f^\ell(x) - h^\ell(x)\right] + h^\ell(x), \tag{39}$$

where

$$h^\ell(x) = \sum_{j \in \mathcal{E}_\ell} y_j b_j K(x, x_j) + b_0. \tag{40}$$

Thus the function itself changes in a piecewise-inverse manner in λ.

If $\mathbf{A}_\ell$ does not have full rank, then the solution paths for some of the α_i are not unique, and more care has to be taken in solving the system (36). This occurs, for example, when two training observations are identical (tied in x and y). Other degeneracies can occur, but rarely in practice, such as three different points on the same margin in $\mathbb{R}^2$. These issues and some of the related updating and downdating schemes are an area we are currently researching, and will be reported elsewhere.

4.1 Finding $\lambda_{\ell+1}$

The paths (38)–(39) continue until one of the following events occur:

1. One of the α_i for $i \in \mathcal{E}_\ell$ reaches a boundary (0 or 1). For each i the value of λ for which this occurs is easily established from (38).

2. One of the points in $\mathcal{L}^\ell$ or $\mathcal{R}^\ell$ attains $y_i f(x_i) = 1$. From (39) this occurs for point i at

$$\lambda = \lambda_\ell \left(\frac{f^\ell(x_i) - h^\ell(x_i)}{y_i - h^\ell(x_i)} \right). \tag{41}$$

By examining these conditions, we can establish the largest $\lambda < \lambda_\ell$ for which an event occurs, and hence establish $\lambda_{\ell+1}$ and update the sets.

One special case not addressed above is when the set $\mathcal{E}$ becomes empty during the course of the algorithm. In this case, we revert to an initialization setup using the points in $\mathcal{L}$. It must be the case that these points have an equal number of +1's as -1's, and so we are in the balanced situation as in 3.1.

By examining in detail the linear boundary in examples where $p = 2$, we observed several different types of behavior:

1. If $|\mathcal{E}| = 0$, than as λ decreases, the orientation of the decision boundary stays fixed, but the margin width narrows as λ decreases.

2. If $|\mathcal{E}| = 1$ or $|\mathcal{E}| = 2$, but with the pair of points of opposite classes, then the orientation typically rotates as the margin width gets narrower.

3. If $|\mathcal{E}| = 2$, with both points having the same class, then the orientation remains fixed, with the one margin stuck on the two points as the decision boundary gets shrunk toward it.

4. If $|\mathcal{E}| \geq 3$, then the margins and hence $f(x)$ remains fixed, as the $\alpha_i(\lambda)$ change. This implies that $h^\ell = f^\ell$ in (39).

4.2 Termination

In the separable case, we terminate when $\mathcal{L}$ becomes empty. At this point, all the ξ_i in (8) are zero, and further movement increases the norm of β unnecessarily.

In the non-separable case, λ runs all the way down to zero. For this to happen without f "blowing up" in (39), we must have $f^\ell - h^\ell = 0$, and hence the boundary and margins remain fixed at a point where $\sum_i \xi_i$ is as small as possible, and the margin is as wide as possible subject to this constraint.

4.3 Computational Complexity

At any update event ℓ along the path of our algorithm, the main computational burden is solving the system of equations of size $m_\ell = |\mathcal{E}_\ell|$. While this normally involves $O(m_\ell^3)$ computations, since $\mathcal{E}_{\ell+1}$ differs from $\mathcal{E}_\ell$ by typically one observation, inverse updating/downdating can reduce the computations to $O(m_\ell^2)$. The computation of $h^\ell(x_i)$ in (40) requires $O(nm_\ell)$ computations. Beyond that, several checks of cost $O(n)$ are needed to evaluate the next move.

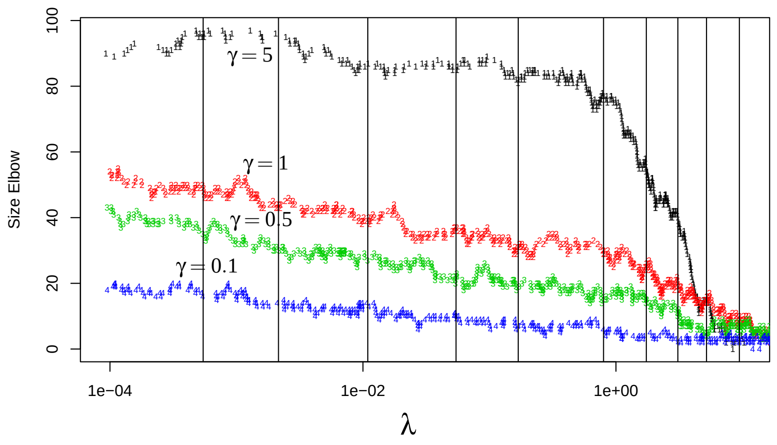


Figure 7: The elbow sizes $|\mathcal{E}_\ell|$ as a function of λ, for different values of the radial-kernel parameter γ. The vertical lines show the positions used to compare the times with `libsvm`.

We have explored using partitioned inverses for updating/downdating the solutions to the elbow equations (for the nonsingular case), and our experiences are mixed. In our R implementations, the computational savings appear negligible for the problems we have tackled, and after repeated updating, rounding errors can cause drift. At the time of this publication, we in fact do not use updating at all, and simply solve the system each time. We are currently exploring numerically stable ways for managing these updates.

Although we have no hard results, our experience so far suggests that the total number Λ of moves is $O(k\min(n_+, n_-))$, for k around $4-6$; hence typically some small multiple c of n. If the average size of $\mathcal{E}_\ell$ is m, this suggests the total computational burden is $O(cn^2m + nm^2)$, which is similar to that of a single SVM fit.

Our R function `SvmPath` computes all 632 steps in the mixture example ($n_+ = n_- = 100$, radial kernel, $\gamma = 1$) in 1.44(0.02) secs on a Pentium 4, 2Ghz Linux machine; the `svm` function (using the optimized code `libsvm`, from the R library `e1071`) takes 9.28(0.06) seconds to compute the solution at 10 points along the path. Hence it takes our procedure about 50% more time to compute the entire path, than it costs `libsvm` to compute a typical single solution.

We often wish to make predictions at new inputs. We can also do this efficiently for all values of λ, because from (28) we see that (modulo $1/\lambda$), these also change in a piecewise-linear fashion in λ. Hence we can compute the entire fit path for a single input x in $O(n)$ calculations, plus an additional $O(nq)$ operations to compute the kernel evaluations (assuming it costs $O(q)$ operations to compute $K(x,x_i)$).

5. Examples

In this section we look at three examples, two synthetic and one real. We examine our running mixture example in some more detail, and expose the nature of quadratic regularization in the kernel feature space. We then simulate and examine a scaled-down version of the $p \gg n$ problem—many more inputs than samples. Despite the fact that perfect separation is possible with large margins, a heavily regularized model is optimal in this case. Finally we fit SVM path models to some microarray cancer data.

5.1 Mixture Simulation

In Figure 4 we show the test-error curves for a large number of values of λ, and four different values for γ for the radial kernel. These λ_ℓ are in fact the *entire* collection of change points as described in Section 4. For example, for the second panel, with $\gamma = 1$, there are 623 change points. Figure 8 [upper plot] shows the paths of all the $\alpha_i(\lambda)$, as well as [lower plot] a few individual examples. An MPEG movie of the sequence of models can be downloaded from the first author's website.

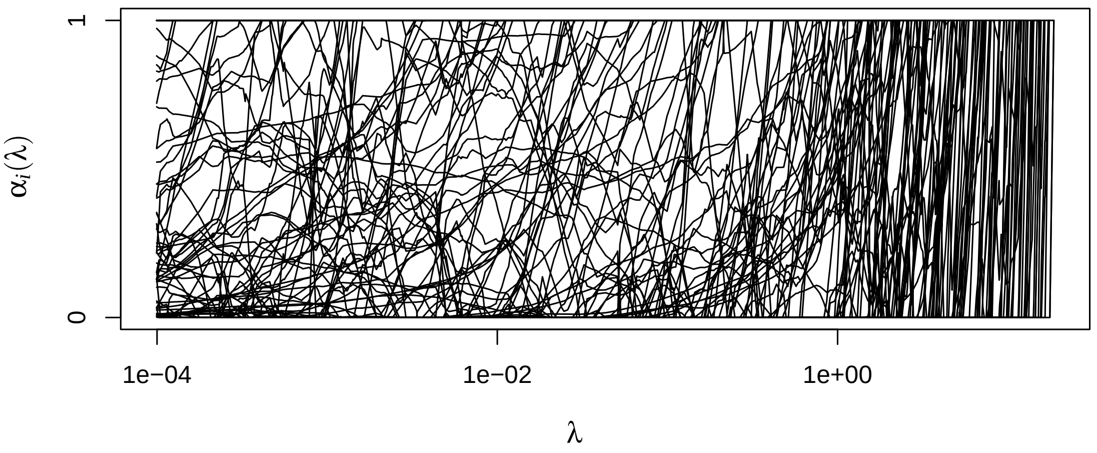
We were at first surprised to discover that not all these sequences achieved zero training errors on the 200 training data points, at their least regularized fit. In fact the minimal training errors, and the corresponding values for γ are summarized in Table 1. It is sometimes argued that the implicit

γ	5	1	0.5	0.1
Training Errors	0	12	21	33
Effective Rank	200	177	143	76

Table 1: The number of minimal training errors for different values of the radial kernel scale parameter γ, for the mixture simulation example. Also shown is the effective rank of the 200×200 Gram matrix $\mathbf{K}_\gamma$.

feature space is "infinite dimensional" for this kernel, which suggests that perfect separation is always possible. The last row of the table shows the effective rank of the kernel *Gram* matrix $\mathbf{K}$ (which we defined to be the number of singular values greater than 10^{-12}). This 200×200 matrix has elements $\mathbf{K}_{i,j} = K(x_i, x_j),\ i, j = 1, \ldots, n$. In general a full rank $\mathbf{K}$ is required to achieve perfect separation. Similar observations have appeared in the literature (Bach and Jordan, 2002; Williams and Seeger, 2000).

This emphasizes the fact that not all features in the feature map implied by K are of equal stature; many of them are shrunk way down to zero. Alternatively, the regularization in (6) and (7) penalizes unit-norm features by the inverse of their eigenvalues, which effectively annihilates some, depending on γ. Small γ implies wide, flat kernels, and a suppression of wiggly, "rough" functions.



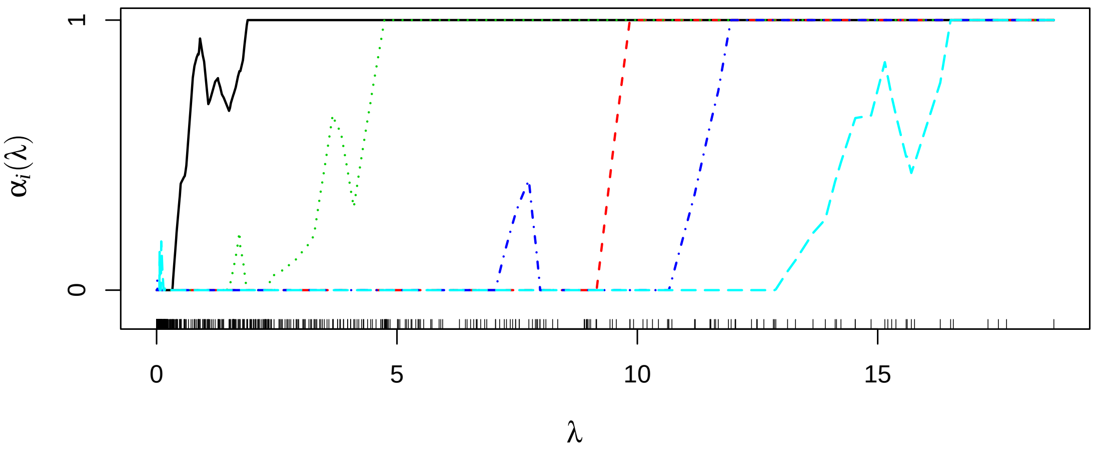


Figure 8: [Upper plot] The entire collection of piece-wise linear paths $\alpha_i(\lambda)$, $i = 1, \ldots, N$, for the mixture example. Note: λ is plotted on the log-scale. [Lower plot] Paths for 5 selected observations; λ is *not* on the log scale.

Writing (7) in matrix form,

$$\min_{\beta_0,\theta} L[\mathbf{y}, \mathbf{K}\theta] + \frac{\lambda}{2}\theta^T\mathbf{K}\theta, \tag{42}$$

we reparametrize using the eigen-decomposition of $\mathbf{K} = \mathbf{UDU}^T$. Let $\mathbf{K}\theta = \mathbf{U}\theta^*$ where $\theta^* = \mathbf{DU}^T\theta$. Then (42) becomes

$$\min_{\beta_0,\theta^*} L[\mathbf{y}, \mathbf{U}\theta^*] + \frac{\lambda}{2}\theta^{*T}\mathbf{D}^{-1}\theta^*. \tag{43}$$

Now the columns of $\mathbf{U}$ are unit-norm basis functions (in $\mathbb{R}^2$) spanning the column space of $\mathbf{K}$;

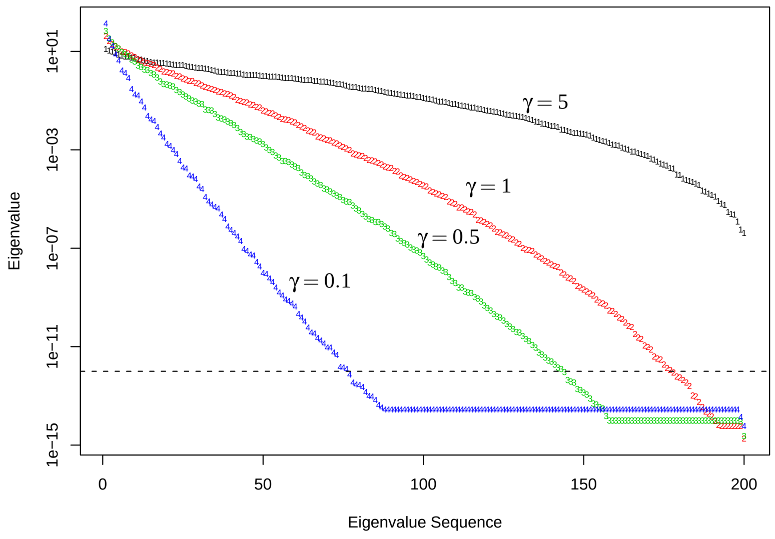


Figure 9: The eigenvalues (on the log scale) for the kernel matrices $\mathbf{K}_\gamma$ corresponding to the four values of γ as in Figure 4. The larger eigenvalues correspond in this case to smoother eigenfunctions, the small ones to rougher. The rougher eigenfunctions get penalized exponentially more than the smoother ones. For smaller values of γ, the effective dimension of the space is truncated.

from (43) we see that those members corresponding to near-zero eigenvalues (the elements of the diagonal matrix $\mathbf{D}$) get heavily penalized and hence ignored. Figure 9 shows the elements of $\mathbf{D}$ for the four values of γ. See Hastie et al. (2001, Chapter 5) for more details.

5.2 p≫n Simulation

The SVM is popular in situations where the number of features exceeds the number of observations. Gene expression arrays are a leading example, where a typical dataset has $p > 10{,}000$ while $n < 100$. Here one typically fits a linear classifier, and since it is easy to separate the data, the optimal marginal classifier is the *de facto* choice. We argue here that regularization can play an important role for these kinds of data.

We mimic a simulation found in Marron (2003). We have $p = 50$ and $n = 40$, with a 20-20 split of "+" and "-" class members. The x_{ij} are all iid realizations from a $N(0,1)$ distribution, except for the first coordinate, which has mean +2 and -2 in the respective classes.[3] The Bayes classifier in this case uses only the first coordinate of x, with a threshold at 0. The Bayes risk is 0.012. Figure 10 summarizes the experiment. We see that the most regularized models do the best here, not the maximal margin classifier.

Although the most regularized linear SVM is the best in this example, we notice a disturbing aspect of its endpoint behavior in the top-right plot. Although β is determined by all the points, the threshold β_0 is determined by the two most extreme points in the two classes (see Section 3.1). This can lead to irregular behavior, and indeed in some realizations from this model this was the case. For values of λ larger than the initial value λ_1, we saw in Section 3 that the endpoint behavior depends on whether the classes are balanced or not. In either case, as λ increases, the error converges to the estimated null error rate n_{min}/n.

This same objection is often made at the other extreme of the optimal margin; however, it typically involves more support points (19 points on the margin here), and tends to be more stable (but still no good in this case). For solutions in the interior of the regularization path, these objections no longer hold. Here the regularization forces more points to overlap the margin (support points), and hence determine its orientation.

Included in the figures are regularized linear discriminant analysis and logistic regression models (using the same λ_ℓ sequence as the SVM). Both show similar behavior to the regularized SVM, having the most regularized solutions perform the best. Logistic regression can be seen to assign weights $p_i(1-p_i)$ to observations in the fitting of its coefficients β and β_0, where

$$p_i = \frac{1}{1+e^{-\beta_0-\beta^T x_i}} \tag{44}$$

is the estimated probability of $+1$ occurring at x_i (Hastie and Tibshirani, 1990, e.g.).

- Since the decision boundary corresponds to $p(x) = 0.5$, these weights can be seen to die down in a quadratic fashion from $1/4$, as we move away from the boundary.

- The rate at which the weights die down with distance from the boundary depends on $||\beta||$; the smaller this norm, the slower the rate.

It can be shown, for separated classes, that the limiting solution ($\lambda \downarrow 0$) for the regularized logistic regression model is identical to the SVM solution: the maximal margin separator (Rosset et al., 2003).

Not surprisingly, given the similarities in their loss functions (Figure 2), both regularized SVMs and logistic regression involve more or less observations in determining their solutions, depending on the amount of regularization. This "involvement" is achieved in a smoother fashion by logistic regression.

5.3 Microarray Classification

We illustrate our algorithm on a large cancer expression data set (Ramaswamy et al., 2001). There are 144 training tumor samples and 54 test tumor samples, spanning 14 common tumor classes that

3. Here we have one important feature; the remaining 49 are noise. With expression arrays, the important features typically occur in groups, but the total number p is much larger.

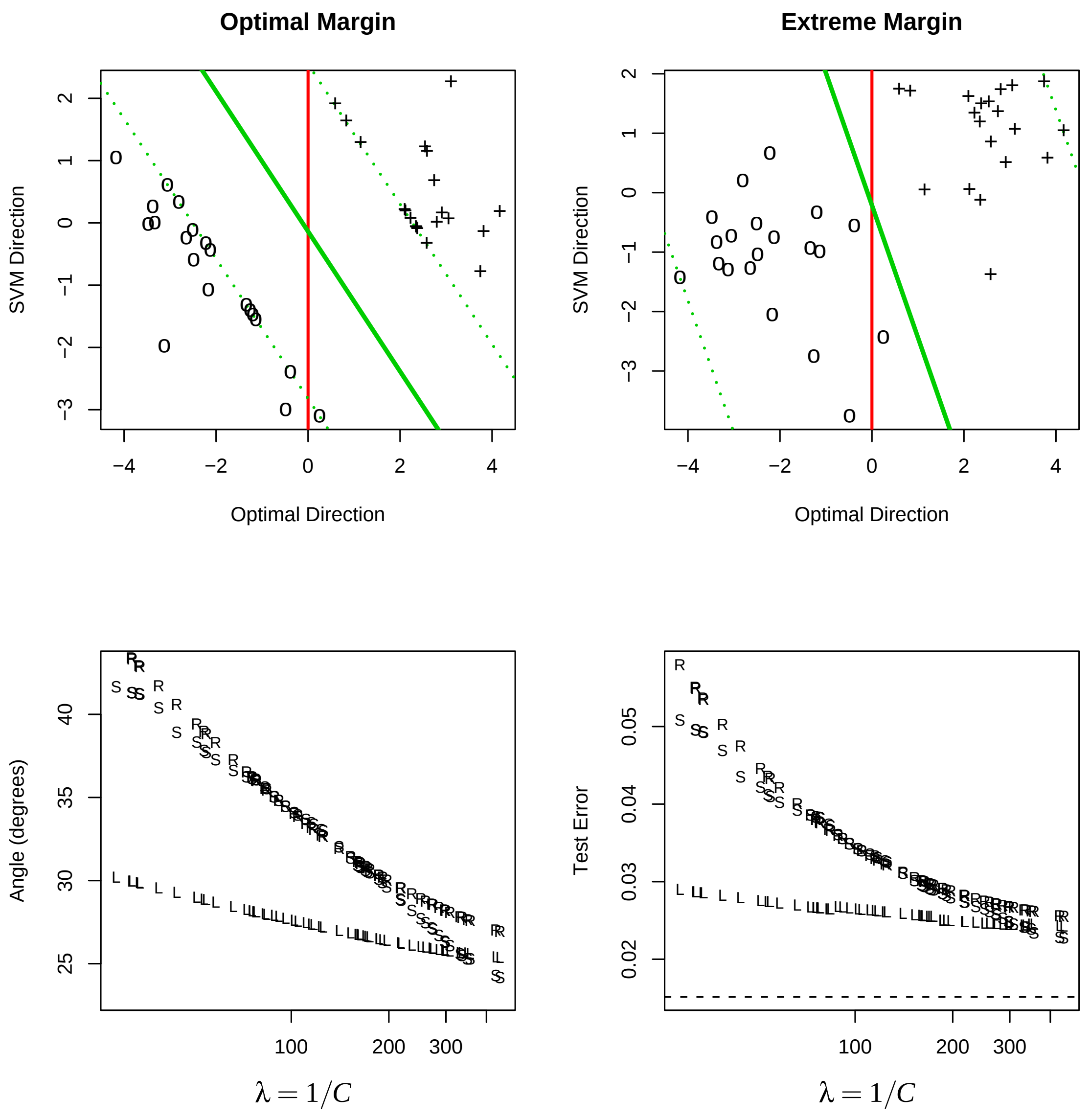


Figure 10: $p \gg n$ simulation. [Top Left] The training data projected onto the space spanned by the (known) optimal coordinate 1, and the optimal margin coefficient vector found by a non-regularized SVM. We see the large gap in the margin, while the Bayes-optimal classifier (vertical red line) is actually *expected* to make a small number of errors. [Top Right] The same as the left panel, except we now project onto the most regularized SVM coefficient vector. This solution is closer to the Bayes-optimal solution. [Lower Left] The angles between the Bayes-optimal direction, and the directions found by the SVM (S) along the regularized path. Included in the figure are the corresponding coefficients for regularized LDA (R)(Hastie et al., 2001, Chapter 4) and regularized logistic regression (L)(Zhu and Hastie, 2004), using the same quadratic penalties. [Lower Right] The test errors corresponding to the three paths. The horizontal line is the estimated Bayes rule using only the first coordinate.

account for 80% of new cancer diagnoses in the U.S.A. There are 16,063 genes for each sample. Hence $p = 16,063$ and $n = 144$. We denote the number of classes by $K = 14$. A goal is to build a classifier for predicting the cancer class of a new sample, given its expression values.

We used a common approach for extending the SVM from two-class to multi-class classification:

1. Fit K different SVM models, each one classifying a single cancer class (+1) versus the rest (-1).

2. Let $[f_1^\lambda(x), \ldots, f_K^\lambda(x)]$ be the vector of evaluations of the fitted functions (with parameter λ) at a test observation x.

3. Classify $C^\lambda(x) = \arg\max_k f_k^\lambda(x)$.

Other, more direct, multi-class generalizations exist (Rosset et al., 2003; Weston and Watkins, 1998); although exact path algorithms are possible here too, we were able to implement our approach most easily with the "one vs all" strategy above. Figure 11 shows the results of fitting this family of SVM models. Shown are the training error, test error, as well as 8-fold balanced cross-validation.[4] The training error is zero everywhere, but both the test and CV error increase sharply when the model is too regularized. The right plot shows similar results using quadratically regularized multinomial regression (Zhu and Hastie, 2004).

Although the least regularized SVM and multinomial models do the best, this is still not very good. With fourteen classes, this is a tough classification problem.

It is worth noting that:

- The 14 different classification problems are very "lop-sided"; in many cases 8 observations in one class vs the 136 others. This tends to produce solutions with all members of the small class on the boundary, a somewhat unnatural situation.

- For both the SVM and the quadratically regularized multinomial regression, one can reduce the logistics by pre-transforming the data. If $\mathbf{X}$ is the $n \times p$ data matrix, with $p \gg n$, let its singular-value decomposition be $\mathbf{UDV}^T$. We can replace $\mathbf{X}$ by the $n \times n$ matrix $\mathbf{XV} = \mathbf{UD} = \mathbf{R}$ and obtain identical results (Hastie and Tibshirani, 2003). The same transformation $\mathbf{V}$ is applied to the test data. This transformation is applied once upfront, and the transformed data is used in all subsequent analyses (i.e. K-fold cross-validation as well).

6. No Intercept and Kernel Density Classification

Here we consider a simplification of the models (6) and (7) where we leave out the intercept term β_0. It is easy to show that the solution for $g(x)$ has the identical form as in (26):

$$g(x) = \frac{1}{\lambda} \sum_{j=1}^{n} \alpha_j y_j K(x, x_j). \tag{45}$$

However, $f(x) = g(x)$ (or $f(x) = \beta^T x$ in the linear case), and we lose the constraint (11) due to the intercept term.

This also adds considerable simplification to our algorithm, in particular the initial conditions.

4. By balanced we mean the 14 cancer classes were represented equally in each of the folds; 8 folds were used to accommodate this balance, since the class sizes in the training set were multiples of 8.

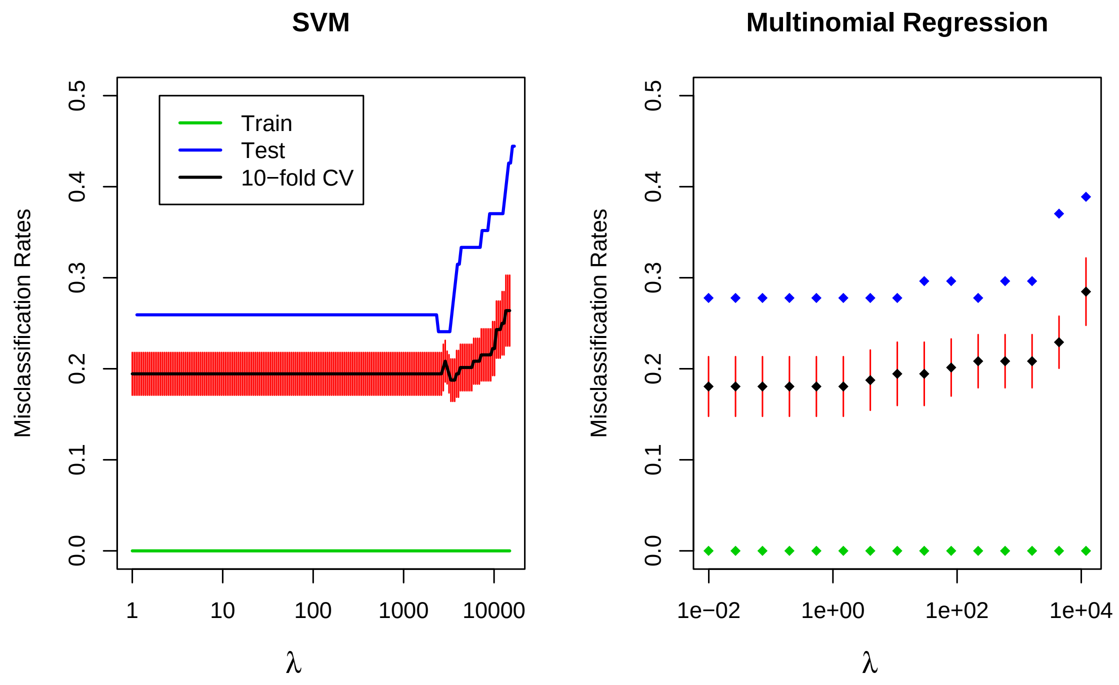


Figure 11: Misclassification rates for cancer classification by gene expression measurements. The left panel shows the the training (lower green), cross-validation (middle black, with standard errors) and test error (upper blue) curves for the entire SVM path. Although the CV and test error curves appear to have quite different levels, the region of interesting behavior is the same (with a curious dip at about $\lambda = 3000$). Seeing the entire path leaves no guesswork as to where the region of interest might be. The right panel shows the same for the regularized multiple logistic regression model. Here we do not have an exact path algorithm, so a grid of 15 values of λ is used (on a log scale).

- It is easy to see that initially $\alpha_i = 1 \forall i$, since $f(x)$ is close to zero for large λ, and hence all points are in $\mathcal{L}$. This is true whether or not $n_- = n_+$, unlike the situation when an intercept is present (Section 3.2).

- With $f^*(x) = \sum_{j=1}^n y_j K(x, x_j)$, the first element of $\mathcal{E}$ is $i^* = \arg\max_i |f^*(x_i)|$, with $\lambda_1 = |f^*(x_{i^*})|$. For $\lambda \in [\lambda_1, \infty)$, $f(x) = f^*(x)/\lambda$.

- The linear equations that govern the points in $\mathcal{E}$ are similar to (33):

$$\mathbf{K}_\ell^* \delta = (\lambda_\ell - \lambda)\mathbf{1}, \tag{46}$$

We now show that in the most regularized case, these no-intercept kernel models are actually performing kernel density classification. Initially, for $\lambda > \lambda_1$, we classify to class +1 if $f^*(x)/\lambda > 0$, else to class -1. But

$$\begin{aligned} f^*(x) &= \sum_{j\in I_+} K(x,x_j) - \sum_{j\in I_-} K(x,x_j) \\ &= n\cdot\left(\frac{n_+}{n}\cdot\frac{1}{n_+}\sum_{j\in I_+}K(x,x_j) - \frac{n_-}{n}\cdot\frac{1}{n_-}\sum_{j\in I_-}K(x,x_j)\right) \qquad (47) \\ &\propto \pi_+h_+(x) - \pi_-h_-(x). \qquad (48) \end{aligned}$$

In other words, this is the estimated Bayes decision rule, with h_+ the kernel density (Parzen window) estimate for the + class, π_+ the sample prior, and likewise for $h_-(x)$ and π_-. A similar observation is made in Schölkopf and Smola (2001), for the model with intercept. So at this end of the regularization scale, the kernel parameter γ plays a crucial role, as it does in kernel density classification. As γ increases, the behavior of the classifier approaches that of the 1-nearest neighbor classifier. For very small γ, or in fact a linear kernel, this amounts to closest centroid classification.

As λ is relaxed, the $\alpha_i(\lambda)$ will change, giving ultimately zero weight to points well within their own class, and sharing the weights among points near the decision boundary. In the context of nearest neighbor classification, this has the flavor of "editing", a way of thinning out the training set retaining only those prototypes essential for classification (Ripley, 1996).

All these interpretations get blurred when the intercept β_0 is present in the model.

For the radial kernel, a constant term is included in span$\{K(x,x_i)\}_1^n$, so it is not strictly necessary to include one in the model. However, it will get regularized (shrunk toward zero) along with all the other coefficients, which is usually why these intercept terms are separated out and freed from regularization. Adding a constant b^2 to $K(\cdot,\cdot)$ will reduce the amount of shrinking on the intercept (since the amount of shrinking of an eigenfunction of K is inversely proportional to its eigenvalue; see Section 5). For the linear SVM, we can augment the x_i vectors with a constant element b, and then fit the no-intercept model. The larger b, the closer the solution will be to that of the linear SVM with intercept.

7. Discussion

Our work on the SVM path algorithm was inspired by earlier work on exact path algorithms in other settings. "Least Angle Regression" (Efron et al., 2002) shows that the coefficient path for the sequence of "lasso" coefficients (Tibshirani, 1996) is piecewise linear. The lasso solves the following regularized linear regression problem,

$$\min_{\beta_0,\beta}\sum_{i=1}^n(y_i-\beta_0-x_i^T\beta)^2+\lambda|\beta|, \qquad (49)$$

where $|\beta|=\sum_{j=1}^p|\beta_j|$ is the L_1 norm of the coefficient vector. This L_1 constraint delivers a sparse solution vector β_λ; the larger λ, the more elements of β_λ are zero, the remainder shrunk toward zero. In fact, any model with an L_1 constraint and a quadratic, piecewise quadratic, piecewise linear, or mixed quadratic and linear loss function, will have piecewise linear coefficient paths, which can be calculated exactly and efficiently for all values of λ (Rosset and Zhu, 2003). These models include, among others,

- A robust version of the lasso, using a "Huberized" loss function.

- The L_1 constrained support vector machine (Zhu et al., 2003).

The SVM model has a quadratic constraint and a piecewise linear ("hinge") loss function. This leads to a piecewise linear path in the dual space, hence the Lagrange coefficients α_i are piecewise linear.

Other models that would share this property include

- The ε-insensitive SVM regression model
- Quadratically regularized L_1 regression, including flexible models based on kernels or smoothing splines.

Of course, quadratic criterion + quadratic constraints also lead to exact path solutions, as in the classic case of ridge regression, since a closed form solution is obtained via the SVD. However, these paths are nonlinear in the regularization parameter.

For general non-quadratic loss functions and L_1 constraints, the solution paths are typically piecewise non-linear. Logistic regression is a leading example. In this case, approximate path-following algorithms are possible (Rosset, 2005).

The general techniques employed in this paper are known as parametric programming via active sets in the convex optimization literature (Allgower and Georg, 1992). The closest we have seen to our work in the literature employ similar techniques in incremental learning for SVMs (Fine and Scheinberg, 2002; Cauwenberghs and Poggio, 2001; DeCoste and Wagstaff, 2000). These authors do not, however, construct exact paths as we do, but rather focus on updating and downdating the solutions as more (or less) data arises. Diehl and Cauwenberghs (2003) allow for updating the parameters as well, but again do not construct entire solution paths. The work of Pontil and Verri (1998) recently came to our notice, who also observed that the lagrange multipliers for the margin vectors change in a piece-wise linear fashion, while the others remain constant.

The `SvmPath` has been implemented in the `R` computing environment (contributed library `svmpath` at CRAN), and is available from the first author's website.

Acknowledgments

The authors thank Jerome Friedman for helpful discussions, and Mee-Young Park for assisting with some of the computations. They also thank two referees and the associate editor for helpful comments. Trevor Hastie was partially supported by grant DMS-0204162 from the National Science Foundation, and grant RO1-EB0011988-08 from the National Institutes of Health. Tibshirani was partially supported by grant DMS-9971405 from the National Science Foundation and grant RO1-EB0011988-08 from the National Institutes of Health.

References

Eugene Allgower and Kurt Georg. Continuation and path following. *Acta Numerica*, pages 1–64, 1992.

Francis Bach and Michael Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS 2000)*, volume 13. MIT Press, Cambridge, MA, 2001.

Dennis DeCoste and Kiri Wagstaff. Alpha seeding for support vector machines. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 345–349. ACM Press, 2000.

Christopher Diehl and Gert Cauwenberghs. SVM incremental learning, adaptation and optimization. In *Proceedings of the 2003 International Joint Conference on Neural Networks*, pages 2685–2690, 2003. Special series on Incremental Learning.

Brad Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. Technical report, Stanford University, 2002.

Brad Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 2004. (to appear, with discussion).

Theodorus Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, Volume 13, Number 1, pages 1-50, 2000.

Shai Fine and Katya Scheinberg. Incas: An incremental active set method for SVM. Technical report, IBM Research Labs, Haifa, 2002.

Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall, 1990.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer Verlag, New York, 2001.

Trevor Hastie and Rob Tibshirani. Efficient quadratic regularization for expression arrays. Technical report, Stanford University, 2003.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003. `http://www.csie.ntu.edu.tw/∼cjlin/libsvm/`.

Thorsten Joachims. *Practical Advances in Kernel Methods — Support Vector Learning*, chapter Making large scale SVM learning practical. MIT Press, 1999. See `http://svmlight.joachims.org`.

Steve Marron. An overview of support vector machines and kernel methods. Talk, 2003. Available from author's website: `http://www.stat.unc.edu/postscript/papers/marron/Talks/`.

Massimiliano Pontil and Alessandro Verri. Properties of support vector machines. *Neural Computation*, 10(4):955–974, 1998.

S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. Mesirov, T. Poggio, W. Gerald, M. Loda, E. Lander, and T. Golub. Multiclass cancer diagnosis using tumor gene expression signature. *PNAS*, 98:15149–15154, 2001.

B. D. Ripley. Pattern recognition and neural networks. Cambridge University Press, 1996.

Saharon Rosset. Tracking curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems (NIPS 2004)*, volume 17. MIT Press, Cambridge, MA, 2005. to appear.

Saharon Rosset and Ji Zhu. Piecewise linear regularized solution paths. Technical report, Stanford University, 2003. `http://www-stat.stanford.edu/∼saharon/papers/piecewise.ps`.

Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin maximizing loss functions. In *Advances in Neural Information Processing Systems (NIPS 2003)*, volume 16. MIT Press, Cambridge, MA, 2004.

Bernard Schölkopf and Alex Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. MIT Press, 2001.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B.*, 58:267–288, 1996.

Vladimir Vapnik. *The Nature of Statistical Learning*. Springer-Verlag, 1996.

G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, 1990.

G. Wahba, Y. Lin, and H. Zhang. Gacv for support vector machines. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 297–311, Cambridge, MA, 2000. MIT Press.

J. Weston and C. Watkins. Multi-class support vector machines, 1998. URL `citeseer.nj.nec.com/8884.html`.

Christopher K. I. Williams and Matthias Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1159–1166. Morgan Kaufmann Publishers Inc., 2000.

Ji Zhu and Trevor Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 2004. (to appear).

Ji Zhu, Saharon Rosset, Trevor Hastie, and Robert Tibshirani. L1 norm support vector machines. Technical report, Stanford University, 2003.