Shrinkage Estimator Generalizations of Proximal Support Vector Machines

Deepak K. Agarwal
AT&T Labs-Research
180 Park Avenue
Florham Park, NJ 07932, USA
dagarwal@research.att.com

ABSTRACT

We give a statistical interpretation of Proximal Support Vector Machines (PSVM) proposed at KDD2001 as linear approximaters to (nonlinear) Support Vector Machines (SVM). We prove that PSVM using a linear kernel is identical to ridge regression, a biased-regression method known in the statistical community for more than thirty years. Techniques from the statistical literature to estimate the tuning constant that appears in the SVM and PSVM framework are discussed. Better shrinkage strategies that incorporate more than one tuning constant are suggested. For nonlinear kernels, the minimization problem posed in the PSVM framework is equivalent to finding the posterior mode of a Bayesian model defined through a Gaussian process on the predictor space. Apart from providing new insights, these interpretations help us attach an estimate of uncertainty to our predictions and enable us to build richer classes of models. In particular, we propose a new algorithm called PSVMMIX which is a combination of ridge regression and a Gaussian process model. Extension to the case of continuous response is straightforward and illustrated with example datasets.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Correlation and regression analysis, Stochastic processes.

General Terms

Algorithms.

Keywords

Bayesian models, classification, correlation, kernel, bias-variance tradeoff, regression.

1. INTRODUCTION

SIGKDD 02 Edmonton, Alberta, Canada

Fung and Mangasarian[13] introduced proximal support vector machine (PSVM) classification as a variation on classical support vector machines (SVM). Their method is based on replacing an inequality by an equality in the defining structure of the SVM framework, and also on replacing absolute error by squared error in the defining minimization problem. The result of these changes is to replace a nonlinear algorithm with complexity similar to that of linear programming to a linear least squares algorithm. The authors claim that this great decrease in computing complexity is achieved with no discernible loss of classification accuracy and robustness. The standard SVM has both "linear" and "nonlinear" variants, where in the latter the matrix product AB is replaced by a nonlinear kernel function $K(A, B)$ in part of the defining framework. They also provide a nonlinear variant of PSVM that can accept an arbitrary kernel $K()$ but still computes as a noniterative least squares algorithm.

In this paper we show that PSVM with a linear kernel is equivalent to ridge regression, a technique that has been extensively studied by statisticians [4, 7, 17, 19, 28]. With nonlinear kernels, we show PSVM is equivalent to a Bayesian model defined through a Gaussian process on the predictor space with the kernel determining the correlation function of the process. The separating hyperplane in this case is given by the posterior predictive distribution of the foregoing Bayesian model.

These statistical interpretations provide evidence that some commonly used data mining algorithms are simple shrinkage estimation techniques used in statistics. Shrinkage estimation is known to be a powerful variance reduction technique and has been successfully applied to data mining problems (see [10] for an example). Some other commonly used variance reduction techniques are bagging [5] and boosting (See [12] and references therein). All these methods inflate bias and work well when the reduction in variance is substantial compared to the increase in bias, leading to an overall reduction in mean squared error. Moreover, working in a statistical framework enables us to attach a measure of uncertainty to our predictions. In particular, we can obtain 95% predictive intervals for each point in our test set. This can potentially help the data analyst build better algorithms. For instance, if most predictive intervals of misclassified points are statistically significant, it is perhaps indicative of overfitting. On the other hand, if most of these predictive intervals are insignificant, it perhaps means we require a better algorithm or need to explore the predictor space more thoroughly.

Another benefit of our statistical interpretation is that it provides guidance into building richer classes of models. By using combinations of linear and nonlinear kernels, we introduce a new algorithm, PSVMMIX, and indicate how it could be implemented by a slight modification of PSVM. Similarly we show how the methodology easily extends itself to regression with continuous response variables.

The paper is organized as follows: In Section 2 we prove the equivalence of PSVM with linear kernel and ridge regression and discuss some commonly used statistical methods used to determine the tuning parameter [17, 28] which may prove useful when implementing SVMs or PSVMs for data mining tasks. In Section 3 we show that the minimization task posed by PSVM with nonlinear kernels is equivalent to finding the posterior mode of a Bayesian model. Section 4 discusses PSVMMIX followed by data examples in Section 5. Section 6 concludes the paper with some discussion and scope for future work.

A word about our notation. All vectors will be column vectors. The transpose of a matrix or a vector will be indicated by the superscript 'T'. For an $m \times n$ matrix A and $n \times k$ matrix B, $K(A, B)$ is an $m \times k$ matrix. In particular, if x and y are vectors, $K(x^T, y)$ is a real number, $K(x^T, A^T)$ is a row vector and $K(A, A^T)$ is an $m \times m$ matrix. For a matrix A, $A_{i.}$ and $A_{.j}$ will denote the ith row and jth column respectively. All probability distributions will be denoted by a single function $f(.)$. For example, $f(y)$ will denote the probability distribution of a random variable Y, $f(y; z, w)$ will denote the conditional distribution of Y given (Z, W). $E(.)$ and $V(.)$ will denote the expectation and variance of a probability distribution respectively. So $E(g(Y); Z, W)$ denotes the expectation $\int g(y)f(y; z, w)dy$ and $V(g(Y); Z, W)$ denotes the variance $\int (g(y) - E(g(Y); Z, W))^2 f(y; z, w)dy$ of $g(Y)$ with respect to the conditional distribution of Y given (Z, W). For any two random variables (X, Y), marginalizing over Y will mean obtaining the marginal density $f(x)$ by integrating out y from the joint density $f(x, y)$. $X \sim N(m, V)$ means the random variable X follows a normal or Gaussian probability distribution with expectation m, dispersion matrix V and density given by $f(x) = 1/\sqrt{}((2\pi)^p |V|)exp(-\frac{1}{2}(x - m)^T V^{-1}(x - m))$, where p is the dimension of X. For each x, $|x|$ will denote the euclidean distance of x from the origin.

2. PSVM WITH LINEAR KERNELS ARE RIDGE REGRESSIONS

The following development shows that the linear version of PSVM is equivalent to ridge regression where the dependent variable, z, is defined as +1 or -1 depending on the class assignment of each of the m items in the training set. From equation (9) of [13], the objective is to minimize,

$$\begin{array}{l} \nu y^T y/2 + (w^T w + \gamma^2)/2, \\ \text{subject to} \quad D(Aw - e\gamma) + y \quad = \quad e, \end{array} \tag{1}$$

where,
D is a diagonal $m \times m$ matrix with $\text{diag}(D) = z$;
A is the $m \times n$ matrix of numeric predictors (features);
e is an $m \times 1$ vector of all 1's;
w is an $n \times 1$ vector of coefficients, one for each feature, to be determined;
γ is a scalar, to be determined;
ν is an arbitrary positive tuning constant for the algorithm that plays a role whenever perfect classification is impossible;
y is an $m \times 1$ vector of absolute "residuals" where the elements of y are defined implicitly by the equality in equation (1).

We convert this into the statisticians' notation for ridge regression using the following equivalencies. Let
$r = Dy$ (interpreted as signed residuals);
$r^T r = y^T y$ (since $D^2 = I$);
$z = De$ (e is a vector of 1s);
$Aw + e\mu + r = z$ **(multiply both sides of equality in (1) by D; $\mu = -\gamma$);
$X = [e \quad A]$ (prepend a column of 1s to A);
$\beta = [\mu \quad w]$ (μ is the coefficient of the constant term);
$X\beta = e\mu + Aw$ (from the previous two definitions);
$r = z - X\beta$ (from **, justifying calling r the signed residual);
$\beta^T \beta = w^T w + \mu^2$ (squared length of coefficient vector).

Finally, we see that the entire objective in (1) can be written as a ridge regression:
minimize $r^T r + \beta^T \beta/\nu$, subject to $r = z - X\beta$.

A Bayesian interpretation of ridge regression is given by the following model:

$$f(z; \beta) \sim N(X\beta, \sigma^2 I); f(\beta) \sim N(0, \nu\sigma^2 I); \tag{2}$$

where σ^2 is the variance of the residuals and I is an identity matrix. Note that the minimization problem posed in (1) is exactly equivalent to finding the posterior mode of β in (2). In fact, the posterior distribution of β is given by

$$f(\beta; z, \nu, \sigma^2) \sim N(LX^T z, \sigma^2 L), \tag{3}$$

where $L = (X^T X + \frac{1}{\nu} I)^{-1}$.

The separating hyperplane for a point x_0 not in the training sample is $x_0^T E(\beta; z) = x_0^T (X^T X + \frac{1}{\nu} I)^{-1} X^T z$ and hence the predictive variance $V(x_0)$ at x_0 is given by

$$V(x_0) = \sigma^2 x_0^T L X^T X L x_0. \tag{4}$$

Note that the predictive variance involves an unknown parameter σ^2 which can be estimated quite reliably in data mining applications by the predictive mean squared error for the test dataset. Due to the Gaussian assumption, a 95% predictive confidence interval at x_0 is given by

$$x_0^T E(\beta; z) \bar{+} 1.96\sqrt{}(V(x_0)).$$

The prior precision $k = 1/\nu$ is sometimes called the "ridge parameter" and determines by how much the least squares coefficients are shrunk toward 0. If $\nu \to \infty$ (or equivalently $k \to 0$), we have a diffuse prior on β and ridge regression coincides with ordinary least squares. On the other extreme, if $\nu \to 0$ (or equivalently $k \to \infty$), $\beta \to 0$ and hence the predictors don't play any role at all in classification resulting in a totally useless classifier. A Bayesian might even have a subjective prior distribution for β, perhaps leading to shrinkage toward values other than $\beta = 0$. Also, the magnitude of shrinkage for all parameters need not be the same. For instance, if the matrix of numeric predictors is composed of all possible terms defined by a polynomial of degree p ($p > 1$), we may want to shrink the higher order terms more than the lower order terms. In most data mining applications, the foregoing Bayesian interpretation allows the analyst the flexibility to try different shrinkage

strategies perhaps based on domain knowledge instead of using just one tuning parameter. Such strategies are effective if the data is linearly separable but contaminated with a lot of noise. As an extreme case, we can have a different ridge parameter for each predictor. However, this will lead to an algorithm with too many tuning constants that will be difficult to implement in practice. In a Bayesian framework one can assume the ridge parameters have a common probability law i.e. $\beta \sim N(0, \sigma^2 \text{diag}(\nu_1, \cdots, \nu_n))$ and the $\nu_i's$ are shrunken further by assuming they are identically and independently distributed with common distribution $f(\nu; a)$ where a is known. For instance, a possible choice of $f(\nu; a = \{C_1, C_2\})$ is a uniform distribution on $[C_1, C_2]$ or even a uniform distribution on a unidimensional grid.

Finally we remark that extension to the case of continuous response variable is trivial in the statistical framework. Ridge regression was originally invented by statisticians as an alternative to ordinary least squares. They argued that even though the ridge estimator was biased, a small bias can often be compensated by much lower variance, such that the overall mean square error of the ridge estimator is smaller than it's ordinary least squares counterpart. Ridge estimation has been generalized for binary response data by [21, 20]. References [23, 27, 25] extend it further for generalized linear models. These models are fitted by means by an iterative procedure which is computationally more intensive. A subtle point to remember while implementing ridge regression for continuous response is to make sure that the response variable is centered at zero. An alternate strategy is to assume $[\mu : w] \sim N((\alpha, 0)^T, \nu\sigma^2 I)$ where α is some measure of central tendency of the response variable (generally the sample mean or sample median).

2.1 Estimating the tuning parameter

In most data mining applications, the ridge parameter ν is chosen by cross-validation, in which some part of the training sample is held back and the value that best predicts this held-back data is our estimate. While there's nothing wrong with this grid search in low dimensions, getting the right scale for the parameter (or the interval of grid search) might turn out to be a difficult task in most data mining applications. The ridge regression literature discusses a plethora of methods to estimate ν from the training data. In practice, it may be a good idea to use them first to get the correct scale for the parameter followed by cross-validation. We discuss some of these methods now and point out the relevant literature.

The simplest method is to graph the coefficients as functions of the ridge parameter k leading to a choice of ν that seem to give "sensible" coefficient estimates. See [30] for further discussion on these plots called "ridge trace plots" in the statistics literature. For data mining applications, although these trace plots may not give us the exact estimate of ν, it certainly helps us to decide on a sensible range $[C1, C2]$ in which to confine our grid search to. While [13] mention estimating ν by using a validation sample, they do not say how they conduct the grid search. The authors in [7] compare several methods of estimating k through a simulation study. Their RIDGM method uses an empirical Bayes approach to estimate k. Efron and Morris in their discussion of [7] suggest a further refinement based on EBMLE discussed in [11]. Reference [17] compares ten different methods of estimating k using a Monte Carlo study and identify three estimators - HKB, RIDGM, and GHW as giving the best overall performance. We give a brief description of these estimators below.

2.2 Hoerl, Kennard and Baldwin estimator

If n is the number of predictors, the HKB estimator is given by $k = n\sigma^2/\beta^T\beta$. Let $\hat{\beta}$ denote the ordinary least squares (OLS) estimator of β and $\hat{\sigma^2} = (z - X\hat{\beta})^T(z - X\hat{\beta})/(n - m - 1)$, an estimate of σ^2. Let $\hat{\beta(k^*)}$ denote the ridge estimate of β at $k = k^*$. At the t^{th} iteration, $k_t = n\hat{\sigma^2}/\hat{\beta(k_{t-1})}^T\hat{\beta(k_{t-1})}$ with $k_0 = n\hat{\sigma^2}/\hat{\beta}^T\hat{\beta}$. The iterated algorithm has been implemented by applying a 10^{-4} convergence criterion to successive k values and defaulting to least squares estimator if convergence is not obtained in 30 iterations.

2.3 RIDGM estimator

Let $G^T X^T X G = \text{diag}(\lambda_1, \cdots, \lambda_n)$ denote the spectral decomposition of $X^T X$ and $\gamma = G^T\beta$. The RIDGM approach is to choose k so that

$$\sum \hat{\gamma_i^2}/(1/k + 1/\lambda_i) = \hat{\sigma^2} n, \tag{5}$$

where $\hat{\gamma} = G^T\beta$. In practice, the solution is obtained by computing the left hand side of equation (5) for a mesh of k values.

2.4 Golub, Heath, and Wahba estimator

Both HKB and RIDGM estimators depend explicitly on least squares estimates of β which may not be available in practice due to singularity of the feature matrix. GHW estimator is attractive in the sense that it does not depend on the OLS estimates. A solution for k is determined by minimizing

$$\begin{aligned} &V(k) = |[I - C(k)]z|^2/(trace[I - C(k)])^2, \\ &\text{where} \quad C(k) = X(X^T X + kI)^{-1} X^T. \end{aligned} \tag{6}$$

In practice, the minimization is implemented by evaluating V(k) for a mesh of k values. The computations are facilitated by the fact that $(X^T X + kI)^{-1} = XG(\text{diag}(1/(\lambda_1 + k), \cdots, 1/(\lambda_n + k)))G^T X^T$ requiring just one matrix inversion for all values of k.

2.5 Bayesian estimate

Finally, it is possible to get a Bayesian estimate by putting a prior on ν and carrying out the estimation process using Markov Chain Monte Carlo (MCMC) algorithms. The MCMC algorithm most commonly used in statistics is Gibbs sampling, popularized by [15]. Such an approach can easily incorporate richer shrinkage strategies as discussed earlier in Section 2. However, MCMC algorithms are iterative in nature and it may be hard to scale them up to massive data mining tasks. With linear kernels, the complexity of MCMC depends mainly on n (the number of predictors). If n is not too large (say < 200), it is feasible to run an MCMC algorithm on the entire training dataset to estimate all the shrinkage parameters in the model. Needless to say, one should only consider taking such an approach if the number of tuning parameters involved is large. For large number of predictors, if one has to rely on MCMC techniques, it should be done after marginalizing over β in (2) i.e. $z \sim N(0, \sigma^2(I + \nu X X^T))$. Since the dispersion matrix

is of order $m \times m$, for large m the MCMC algorithm should be implemented on a small subsample of the training set.

3. PSVM WITH NONLINEAR KERNELS

We now give a statistical interpretation of PSVM with nonlinear kernels. From equation (18) of [13], the objective is to minimize (w.r.t. u, μ)

$$\begin{array}{c} \nu r^T r/2 + (u^T u + \mu^2)/2, \\ \text{subject to} \quad r = z - AA^T Du - \mu e. \end{array} \tag{7}$$

Fung and Mangasarian[13] convert this into an algorithm for nonlinear kernel K by replacing AA^T with $K(A, A^T)$ in equation (7). Motivated by [22, 24], a more general version of this minimization problem can be obtained by replacing $u^T u$ in equation(7) with $u^T Hu$, where H is a positive definite matrix. In fact, if we replace w in equation (9) of [13] by $A^T Du$ in their equation (18), we should have $u^T DAA^T u$ in equation (7). We restate the general version of the minimization problem: minimize(w.r.t. u, μ)

$$\begin{array}{c} \nu r^T r/2 + (u^T D^T HDu + \mu^2)/2, \\ \text{subject to} \quad r = z - KDu - \mu e. \end{array} \tag{8}$$

Consider the transformation $\theta(A) = K(A, A^T)Du$. Then equation (8) can be rewritten in terms of θ as minimize (w.r.t. θ, μ)

$$\begin{array}{c} \nu r^T r/2 + (\theta(A)^T K^{-1}(A, A^T) HK^{-1}(A, A^T)\theta + \mu^2)/2, \\ \text{subject to} \quad r = z - \theta - \mu e. \end{array} \tag{9}$$

This is immediately seen to be equivalent to finding the posterior mode of the following Bayesian model:

$$\begin{array}{c} f(z; \theta(A), \mu) \sim N(\mu e + \theta(A), \sigma^2 I); \\ f(\theta(A), \mu) = f(\theta(A)) f(\mu); \\ f(\theta(A)) \sim N(0, \nu\sigma^2 K(A, A^T) H^{-1} K(A, A^T)); \\ f(\mu) \sim N(0, \nu\sigma^2). \end{array} \tag{10}$$

If we assume $H = K^{-1}$ (we will get this with PSVM if we make the correction to equation (18) of [13] as discussed earlier), θ is a mean zero Gaussian process on the predictor space $\mathcal{X}$ with correlation between any two points $x, y \in \mathcal{X}$ given by $K(x^T, y)$. We will stick to this formulation in the rest of the paper unless mentioned otherwise. The tuning parameter ν has a different interpretation in this context which we explain. For simplicity, let's assume that $K(x^T, x)$ is constant for each $x \in \mathcal{X}$. This is true for all kernels $K(x^T, y)$ which depend on x and y only through $|x - y|$. Then the variance of the Gaussian process is $\nu\sigma^2$ where σ^2 is the variance of the residuals. Thus, ν is the ratio of the variance of the Gaussian process to that of the residuals. In fact, obtaining better performance with a small value of ν would indicate the Gaussian process fails to explain most of the variability in the data indicating that we need a better model. Note that the constant term μ also shrinks toward zero as in ridge regression. This motivates us to replace μe with $M\beta$ in the foregoing model where M is a known feature matrix and $\beta \sim N(0, \nu\sigma^2 I)$. With this extension, large values of ν combine nonlinearity with ordinary least squares. With small values of ν, the effect of the Gaussian process weakens and ridge regression takes over. If we always want ridge regression to work irrespective of the influence of the Gaussian process (e.g. if M is nearly singular), we need to introduce an additional tuning constant into our algorithm by assuming $\beta \sim N(0, \nu\nu_\beta\sigma^2 I)$. We will discuss fitting these richer classes of models later in the paper. To obtain the separating hyperplane at a point x not in the training sample, note that marginalizing over (θ, μ), we obtain

$$f(z(x), z) \sim N(0, \nu\sigma^2(\tfrac{1}{\nu} I + K(B, B^T) + ee^T)), \tag{11}$$

where $B^T = (x^T : A^T)$.

Hence, the separating hyperplane and the predictive variance at x are given by the following equations:

$$\begin{array}{l} E(z(x); z) = K(x^T, A^T)v + e^T v, \\ \text{where} \quad v = (\tfrac{1}{\nu} I + K(A, A^T) + ee^T)^{-1} z. \\ V(z(x); z) = \sigma^2(1 + 2\nu - \nu(((K(x^T, A^T) + e^T) \\ (I/\nu + K(A, A^T) + ee^T)^{-1}(K(x, A) + e))). \end{array} \tag{12}$$

A conservative estimate of σ^2 can be obtained by taking the ratio of predictive MSE to $(1 + 2\nu)$ enabling us to compute predictive confidence intervals. Note that we only need to perform an extra matrix multiplication to compute the predictive variance.

3.1 Richer classes of models: PSVMMIX

As discussed earlier, we can enrich the class of statistical models by replacing μe in equation (10) with $M\beta$, where M is a known feature matrix of order $m \times L$ and assuming $\beta \sim N(0, \nu_\beta\nu\sigma^2 I)$. Note that the predictors used in forming M may or may not overlap with those used to form K. We shall call this the PSVMMIX(ν_β) model. A special important case is the PSVMMIX(1) model in which the shrinkage of the linear and nonlinear terms are controlled by a single tuning constant ν. With $M = e$, PSVMMIX(1) reduces to PSVM with nonlinear kernel. The separating hyperplane and the predictive variance at a point x not in the training sample are given as follows:
Let x_{oM} and x_{oK} denote subsets of x that enter the linear and nonlinear part of the algorithm respectively. Then,

$$\begin{array}{l} E(z(x); z) = K(x_{oK}^T, A^T)v + \nu_\beta x_{oM}^T M^T v, \\ \text{where} \quad v = (\tfrac{1}{\nu} I + K + \nu_\beta MM^T)^{-1} z. \\ V(z(x); z) = \sigma^2(1 + \nu\nu_\beta x_{oM}^T x_{oM} + \nu \\ -\nu\kappa^T(I/\nu + K + \nu_\beta MM^T)^{-1}\kappa, \\ \text{where} \quad \kappa^T = K(x_{oK}^T, A^T) + \nu_\beta x_{oM}^T M^T. \end{array} \tag{13}$$

Fitting PSVMMIX requires an additional tuning parameter ν_β. Ideally, if we have l tuning parameters, they should be estimated in one sweep by conducting a grid search in an l dimensional space. In practice, it may be a good idea to first fit PSVMMIX(1) followed by a one dimensional grid search for ν_β in a neighborhood of ν optimal for PSVMMIX(1). So far, the latter has worked well for us.

4. FITTING PSVMMIX TO A LARGE TRAINING SET

Computing the separating hyperplane from equation (13) requires the inversion of an $m \times m$ matrix $V = (\frac{1}{\nu} I + K + MM^T)$. For large values of m the inversion of $Q = (\frac{1}{\nu} I + K)$ is usually facilitated by using compactly supported kernel functions (See [16] and references therein). These kernels make K a sparse matrix with a much smaller bandwidth b_w than m enabling us to compute the inverse using mb_w^2 flops. However, the presence of MM^T completely destroys this sparsity. We now express $E(z(x); z)$ and $V(z(x); z)$ in

terms of Q enabling us to exploit the sparsity of K for large m. Equation (13) can be rewritten as follows:

$$E(z(x);z) = E(E(z(x);z,\beta);z) =$$
$$x^T E(z;\beta) + K(x^T, A^T)Q^{-1}(z - ME(z;\beta)),$$
$$\text{where } E(z;\beta) = (M^T Q^{-1} M + \tfrac{1}{\nu_\beta} I)^{-1} M^T Q^{-1} z,$$
$$\text{and}$$
$$V(z(x);z) = E(V(z(x);z.\beta),z) + V(E(z(x);z,\beta),z). \quad (14)$$

Assuming the number of predictors L involved in M is small, the expressions in (14) can be evaluated easily once we have inverted Q using efficient inversion routines for sparse matrices (See [18] for details on using these routines in MATLAB). The limit on the size of L is not too restrictive for practical applications. Using a very large L would kill the contribution of the Gaussian process to a large extent and we may end up approximately implementing a linear version of PSVM. On the other hand, if we shrink the linear terms too much by making ν_β small, we end up approximately implementing a nonlinear version of PSVM and unnecessarily wasting computational time in the process.

4.1 Implementation Details

Since Q is positive definite, the actual estimation process is carried out by using a sparse Cholesky decomposition of Q; i.e. $Q = GG^T$ where G is a lower triangular matrix and solving a system of linear equation $Qx = b$ for a bunch of unknown b's. Once the factorization is known, solving this system for any known b involves m forward substitutions followed by m backward substitutions. All our computations were done on an SGI server with 195 MHz processor using sparse matrix routines in the meschach library, a numerical library of C routines for performing calculations on matrices and vectors [29]. A pseudo code for implementing PSVMMIX for known values of ν, s (tuning parameter associated with the kernel) and ν_β at a point x_0 not in the training sample is given below. The two main functions used are spCHfactor(P) (returns the Cholesky factorization of a sparse input matrix P) and spCHsolve(W, w) (returns the solution of $Px = w$ using the sparse Cholesky factorization W = spCHfactor(P) of P). The corresponding functions for dense matrices are CHfactor and CHsolve. %*% will denote matrix multiplication and t(.) will denote the transpose operator for vectors and matrices.

1. G = spCHfactor(Q);
2. xx = t(M)%*%spCHsolve(G,z);
3. for(i in 1:L) $OUT_{.i}$ = spCHsolve(G,$M_{.i}$);
4. WORK = t(M)%*%OUT + $(1/\nu_\beta)$*I;
5. temp = CHfactor(WORK);
6. Betaest = CHsolve(temp,xx);
7. ee = z - M%*%Betaest;
8. xx = spCHsolve(G,ee);
9. $kk = K(x_{oK}^T, A^T)$;
10. Predictor = t(x_{oM})%*%Betaest + kk%*%xx;

Note that in step 5 we have to do a Cholesky factorization of an $L \times L$ dense matrix. This shows clearly why we can't afford to use too many predictors in the linear part of PSVMMIX. To implement a grid search for (ν, s, ν_β), note that at each fixed value of (ν, s), steps 1 to 3 are implemented once followed by looping steps 4 to 10 over values of ν_β.

Memory requirements for the algorithm are facilitated by use of sparse kernels. Instead of storing an $m \times m$ matrix Q we have to only store the non-zero entries of the sparse matrix Q which is of order $O(mb_w)$. Loosely speaking, we can think of b_w as the maximum number of nearest neighbors used by the training algorithm. Storing the dense matrix M is of order $O(mL)$ and hence the total memory requirement is of order $O(m \max(L, b_w))$. The main computationally intensive tasks performed by the algorithm are the inversion of Q which is of order $O(mb_w^2)$ and inversion of the $L \times L$ dense matrix in step 5 of the psuedo-code which is of order $O(L^3)$. Hence, for small values of b_w and L, the algorithm scales linearly both in space and time approximately. Note that the predictors used in the nonlinear part only enter the algorithm through the pairwise distance matrix used to compute the kernel matrix K which is computed at the beginning. Thus, we can have a very large number of predictors in the nonlinear part without affecting the performance of the algorithm both in terms of time and space. In section 5.6 we illustrate the performance of the algorithm on a dataset consisting of large number of observations. We have recently applied the algorithm to a dataset consisting of a large number of predictors (1200 approximately). However, we won't report that analysis here due to space constraint but can provide details on request.

Table 1: Training and ten-fold cross validation accuracy for the Ionosphere dataset for two different models

Model	Training set accuracy	Test set accuracy
PSVM	93.73%	89.14%
PSVM2	94.59%	90.57%

5. DATA EXAMPLES

5.1 Ionosphere Data

This publicly available dataset from the UCI Machine Learning Repository [26] consists of 351 observations and 34 predictors. All the predictors are continuous. The response variable is binary. We fitted a PSVM model with linear kernel using 69 predictors $[1, x_1, \cdots, x_{34}, x_1^2, \cdots, x_{34}^2]$. The value of ν was estimated by doing a grid search for $k \in \{1, 2, \cdots, 50\}$. Our final estimator was $\nu = 1/37$. We also computed the GHW estimate for ν which was 1/14 in this case. Figure 1 shows separate ridge trace plots (β vs $k = 1/\nu$). The trace plots for the linear and quadratic terms are different. The latter looks like a funnel with a bigger radius indicating it may require more shrinkage. We fitted a second model which we call PSVM2 with two distinct shrinkage parameters ν_1 and ν_2 for the linear and quadratic terms. The parameters were estimated by means of a two dimensional grid search. The estimated values in this case were $\nu_1 = 1/5$, $\nu_2 = 1/25$ indicated by vertical lines in figure 1. The ten-fold cross-validation accuracy was computed by randomly dividing the entire data into 10 random subsets of equal size (one of them had an extra observation) and conglomerating 9 parts into a training set while the 10th part was used as a test set. This was done 10 times (each part taken as a test set once) and the average accuracy reported in table 1. PSVM2 gives us better test set accuracy than the PSVM model in this case.

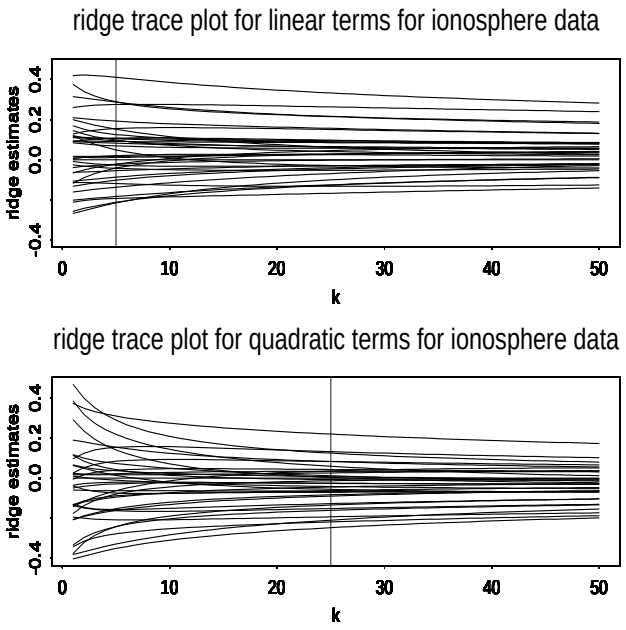

Figure 1: Ridge Trace Plots for the Ionosphere Data. Estimated shrinkage parameters are indicated by vertical lines.

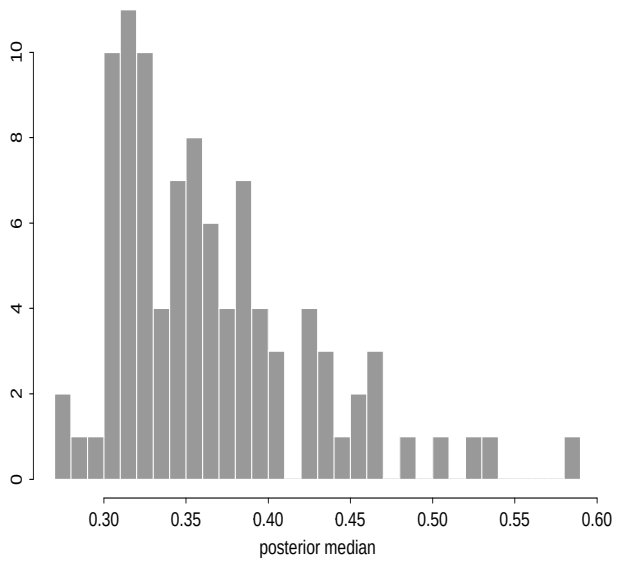

Figure 2: Posterior median of ν's for the adult dataset

5.2 Adult dataset

We analyzed the "Adult" dataset, which is publicly available from the Silicon Graphics website [1]. Before proceeding with the analysis, all records with missing information were removed. A $2/3, 1/3$ training and test data split was used as discussed in [1]. Our training set consisted of 30161 observations while the test dataset consisted of 15060 records. A total of 96 predictors were used. We performed a fully Bayesian analysis with a linear kernel assuming a separate shrinkage parameter for each component of β. A priori, we assumed the common distribution of $\nu's$ is uniform on a grid spanning from .01 to 6 with intervals of length .01 between .01 and 1 and of length .1 between 1 and 6. The grid design was constructed after examining the ridge trace plots of the predictors. Figure 2 shows the histogram of the posterior medians of ν_i's. In 95% of the cases, the medians were between .3 and .53, indicating there is not much variability among the ν_i's and we would not gain much by using a different shrinkage parameter for each component of β. This was further endorsed by the test accuracy which was 83.84%, close to those reported for SVM, PSVM and other standard classifiers (Table 1 in [13])

5.3 Spiral Dataset

We analyzed the spiral dataset [16] using a Gaussian kernel $K(x^T, y) = exp(-s|x-y|^2)$. The dataset consists of 194 black and white points intertwined in the shape of a spiral. All accuracy results for this dataset are reported using the leave-one-out method. Private communication with the authors of [13] led us to use $\nu = 10^5$ and $s = 4.0$. This gave us a classification accuracy of 76.3% which is quite poor compared to the performance of some other data mining algorithms on this dataset. For instance, [14] report an accuracy of 88.14%. On reducing the value of s to .4, the accuracy went up to 98.97%. Intuitively, since $K(x^T, y)$ denotes the correlation between x and y (which is a decreasing function of $|x-y|$), reducing the value of s is equivalent to increasing the number of nearest neighbors. Thus, reducing s from 4.0 to .4 is roughly equivalent to a 3 fold increase in the number of nearest neighbors. On reducing s to .04 (approximately 10 fold increase in the number of nearest neighbors compared to $s = 4.0$), the accuracy drops down to 46.39%. Apart from inducing positive association between like colored points, such a small value of s also induce positive association between opposite colored points resulting in degraded performance. With $s = .4$, there is strong correlation between like colored points and negligible correlation between points of different colors. This simple dataset is a good example where statistical intuition helps us better understand a problem and achieve excellent performance. Figure 3 shows the 95% predictive intervals for each of the 194 points. None of the predictive intervals contain 0 showing that all our predictions are statistically significant. Also, as we move away from the centroid $(0, 0)$ of the dataset, predictive variability tends to become larger. This is a common phenomenon in most stat estimation problems - accuracy diminishes near the boundary of the predictor space.

5.4 Boston housing data

To illustrate the extension of PSVM to continuous response data, we analyzed the Boston housing data publicly available from UCI machine learning repository [26]. This data was also analyzed in [8] using Support vector regression machines. Our response variable here is MEDV, median house price. The dataset consists of 12 continuous predictors and one nominal valued predictor called CHAS which indicates the proximity of a house to Charles river. The latter was not used by [8] in their analysis. We randomly parti-

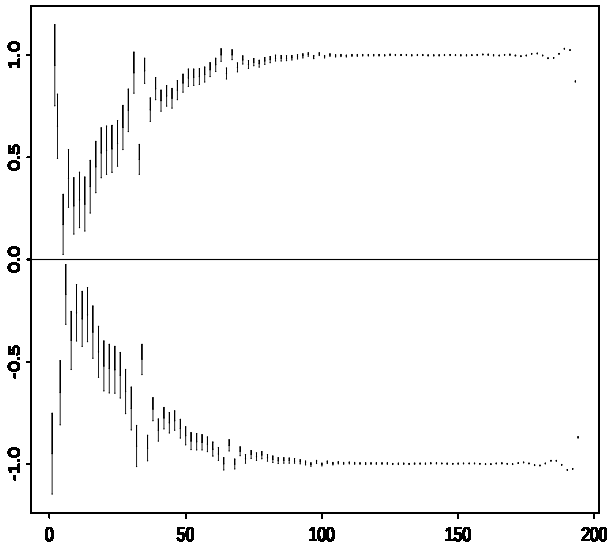

Figure 3: 95% predictive confidence intervals for spiral data. Horizontal axis denotes the observation id sorted in descending order by distance from the centroid $(0, 0)$.

tioned the database consisting of 506 records into a training sample (401 records), validation sample(80 records) and test sample (25 records) as in [8]. All continuous predictors were centered at 0 and scaled by its standard deviation. We first fit PSVM using only the continuous valued predictors and a Gaussian kernel. Values of (ν, s) were chosen by considering a mesh of values $\{\ 10, 20, 30, \cdots, 100\ \}$x $\{.005, .01, \cdots, 1\ \}$ and choosing the pair that gives the minimum prediction error on the validation sample. For PSVM excluding CHAS, our estimates were (40,.085). Next, we centered CHAS at 0 and scaled it to have a standard deviation of 1. We now fit PSVM with all 13 predictors. Estimates of ν and s in this case were 40 and .035 respectively. We fit another version of PSVM without centering or scaling CHAS. Our parameter estimates in this case were 40 and .045 respectively. Finally, we fit PSVMMIX by putting CHAS in the linear part and the continuous predictors in the nonlinear part. We centered our response variable at 0. On conducting cross-validation on the validation set we observed that for each fixed (ν, s), the value of ν_β was almost equal to 0 showing that presence of CHAS in the linear part does not help much. We tried another version without centering the response variable. Our estimates in this case were $\nu = 50, s = .085$ and $\nu_\beta = .015$. Table 2 report training and test MSE's for all the models. PSVM without CHAS does better than SVR. On introducing CHAS (centered and scaled) into PSVM, the performance deteriorates. Without any transformation, PSVM with CHAS does well compared to PSVM without CHAS. PSVMMIX (response centered at 0) gives results almost identical to PSVM (without CHAS) as discussed above. In all cases, the MSE on the training set is smaller than the test set which is expected. However, the training MSE relative to test MSE for PSVM without CHAS is smaller compared to PSVM's with CHAS. The training MSE for SVR was not reported by [8] for comparison. Surprisingly, we get better performance on the test set shrinking the response toward 0.

Table 2: Predictive and Training set MSE's for the Boston housing data

Model	TrainMSE	Pred MSE
SVR	–	7.2
PSVM(w/o CHAS)	2.423	6.359
PSVM(with CHAS) centered and scaled	4.152	6.659
PSVM(with CHAS) (original scale)	3.671	5.548
PSVMMIX (response centered)	2.423	6.106
PSVMMIX (response on original scale)	3.964	2.457

Table 3: BUPA Liver data

Model	Training set accuracy	Test set accuracy
PSVM	78.95%	72.65%
PSVMMIX	78.95%	72.46%

5.5 BUPA Liver Dataset

The BUPA Liver Disorders dataset from Irvine Machine Learning Database Repository [26] consists of 345 data points with 6 continuous predictors and a selector field used to split the data into 2 sets with 145 and 200 instances respectively. We fitted two models to this dataset, PSVM with a Gaussian kernel and PSVMMIX with all the 6 predictors in the linear and nonlinear part. Tuning parameters were all selected by means of a small validation set. In table 3 we report the ten-fold cross validation accuracy for the two models. Adding all the predictors both in the linear and nonlinear part does not help much here. We get almost identical accuracies but end up wasting computational time fitting PSVMMIX since it involves an additional Cholesky decomposition of an $L \times L$ matrix with $L = 7$ (6 predictors and one intercept).

5.6 PSVMMIX on a large dataset

To exemplify fitting PSVMMIX to large datasets, we used a database consisting of 15277 transactions of single-family homes sold in Dallas, Texas during 1996. The primary source of information is the Dallas Central Appraisal District (DCAD). Attached to each house in Dallas are its structural characteristics and address. The addresses were geocoded enabling us to attach a latitude and longitude to each house in our database. Each house was assigned to an independent school district (ISD). The data had a total of 13 ISD's. The natural logarithm of selling price of houses was used as our response variable since it looked more Gaussian than the raw selling price verified using quantile plots (See figure 4). We give a list of predictors in our database with a brief description of each:

Continuous predictors:
1)LAT and LONG - Geocoded co-ordinates. 2)LIVAREA - total living area in square feet.

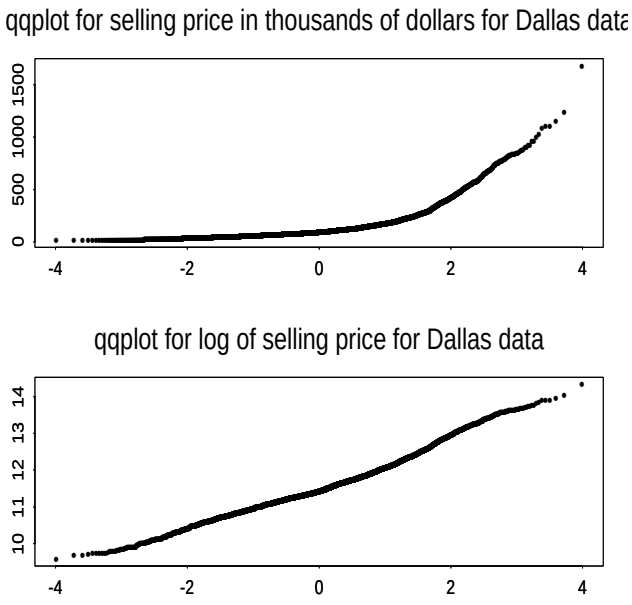

Figure 4: qqplots for selling price on the original and log scale for Dallas data

3)AGE - Age of the dwelling in decades.
4)BATHS - Number of baths in the house. Two half bathrooms are counted as one.

Nominal predictors:
5) ISD - Independent School District to which the house belongs. There are 13 such districts in the database.
6) pool - A 0/1 valued variable with 1 denoting the presence of a pool in the house.
7)wetbar - A 0/1 valued variable with 1 denoting presence.

Thus, including a dummy variable for each school district, we have a total of 20 predictors in our database. All continuous predictors were centered at zero and scaled by their respective standard deviation. Motivated by [2], we decided to use LAT and LONG in the nonlinear part of PSVMMIX with the remaining 18 predictors used in the linear part. To make Q sparse, we worked with the following compactly supported kernel [16]

$$\tilde{K}(x^T, y) = exp(-s|x-y|^2)\max((1-\tfrac{|x-y|}{\tilde{\theta}})^{\tilde{\nu}}, 0) \quad (15)$$

where $\tilde{\nu} \geq (n+1)/2$ and $\tilde{\theta} > 0$. Note that smaller values of $\tilde{\theta}$ will encourage sparsity. The value of $\tilde{\nu}$ controls the rate at which the correlations approach zero.

To fit PSVMMIX using kernel defined by equation (15) requires estimating constants $\nu, s, \tilde{\nu}, \tilde{\theta}$ and ν_β. Fitting PSVMMIX to several small subsamples from the training and validation set using a Gaussian kernel helped us narrow down our grid search for ν, s and ν_β. For the current dataset, we decided to use $s \in \{.1, .6, .9, 1.0\}$, $\nu \in \{20, 50, 100\}$ and $\nu_\beta \in \{.3, .7, 1.0\}$. Since we have n = 2 predictors in the nonlinear part, $\tilde{\nu}$ must be ≥ 1.5. We used $\tilde{\nu} = 2$. If N_i denotes the number of non-zero correlations of the ith data point in the training set (i.e. number of non-zero entries in the ith row of $K(A, A^T)$ or the number of nearest neighbors for the ith data point), bandwidth b_w is given by $b_w = \max_i N_i$. The computational time required to do a Cholesky factorization of Q increases approximately quadratically with b_w. Also, the memory requirement is proportional to the number of non-zero entries in Q. From equation (15), it's clear that $\tilde{\theta}$ is an increasing function of b_w. Higher values of $\tilde{\theta}$ would give us good predictions but increase computational time. We divided our dataset into three random subsets viz a training set consisting of 10000 points, a test set consisting of 3277 points and a validation set consisting of 2000 points. With $\tilde{\theta} = .001D$ (where D is the maximum pairwise distance), the value of b_w was 27. Our grid search for s, ν and ν_β on the validation set was done for this value of b_w to speed up computations and we finally decided to use $s = .1, \nu = 50$ and $\nu_\beta = .3$. Fixing s, ν, ν_β and $\tilde{\nu}$ at .1,50,.3 and 2 respectively, we conducted a grid search to select a "sensible" value of $\tilde{\theta}$. Table 4 reports the predictive mse and computational time for different values of b_w computed for the validation set. Based on the results, we decided to use $\tilde{\theta} = .02D$. In table 5 we report the predictive MSE for PSVMMIX(ν_β). Table 6 gives coefficient estimates and p-values for some important predictors in the linear part. Increase in living area, presence of more bathrooms and presence of a pool all lead to an increase in house price. On the other hand, house prices tend to depreciate with age. Wetbar turns out to be statistically insignificant.

Table 4: Predictive MSE and computational time in seconds for different bandwidths for Dallas data.

θ/D	b_w	predMSE	time
.025	349	.196	752.77
.020	257	.217	392.19
.015	162	.243	142.58
.01	88	.275	44.56
.005	38	.317	16.53
.001	27	1.333	15.1

Table 5: Predictive MSE for test and training dataset for Dallas data

Model	Training set accuracy	Test set accuracy
PSVMMIX($\nu_\beta = .3$)	.0013	.1655

Table 6: Coefficient estimates for PSVMMIX for Dallas data. Numbers in parantheses are estimated standard deviations rounded to three decimal places

coeff	estimate	p-value
LIVAREA	.198 (.002)	0.000
AGE	-.030 (.002)	0.00
BATH	.044 (.003)	0.00
POOL	.086 (.003)	0.00
WETBAR	-.002 (.004)	.559

6. DISCUSSION AND FUTURE WORK

We have given a statistical interpretation of PSVM with linear and nonlinear kernels. These interpretations provide further evidence that shrinkage estimators are perhaps the best data mining tools around. The equivalence of ridge regression and PSVM with linear kernels enable us to take advantage of the rich statistics literature to estimate the tuning parameter more effectively, an issue that was ignored in earlier work. Better shrinkage strategies to control the bias-variance tradeoff in our predictions can be considered by increasing the number of tuning parameters. This may lead to better classification as illustrated by the Ionosphere dataset. With a small number of predictors, the tuning constants could be estimated using MCMC methods as illustrated with the adult dataset. With nonlinear kernels, we showed PSVM is equivalent to a Bayesian model defined through a Gaussian process on the predictor space with the correlations determined by the kernel. We then introduced a new class of models called PSVMMIX that were combinations of linear and nonlinear PSVM's. We demonstrated how these models could be fitted to large datasets by exploiting the sparsity in $K(A, A^T)$. However, more methodological research is needed to efficiently choose the tuning parameters for very large datasets. This is an issue that pops up even in the context of PSVM's and SVM's and needs further investigation. For instance, [9] report on an empirical study where they compare several functionals that were used to tune the parameters (called hyperparameters by them) for SVM's. In our case, we used the cross-validation MSE to tune the parameters which is an expensive functional to compute. We are currently investigating proxies that are cheap to compute and would enable us to estimate the tuning constants quicker for large datasets. One possibility is to use Kernelized Locally Linear Embeddings (KLLE) proposed in [6]. Although not exemplified, extension to the case of multicategory data with nominal categories is trivial. With k categories ($k > 2$), the classification problem is solved by implementing k 2-class problems. With ordinal categories (e.g. User satisfaction on a 3 point scale: bad, satisfactory and good) the classification problem could be solved using an ordinal response model (see [3] and references therein).

Another issue is selecting predictors to go in the linear part and nonlinear part of PSVMMIX. Putting everything in the linear and nonlinear part may not help much as illustrated with the BUPA dataset. Increasing the number of predictors L that goes in the linear part increases computational time cubicly in L since the algorithm involves a Cholesky decomposition of an $L \times L$ dense matrix. In general, if there are predictors that are known to have a casual relationship with the response variable, our recommendation is to put them in the linear part. The coefficient estimates which we obtain at the end of the day would enable us to check whether the causality is respected by the algorithm or not. If we have a combination of nominal valued and continuous valued predictors (very common in data mining applications), our recommendation is to put all nominal valued predictors in the linear part. Centering and scaling predictors may sometime lead to bad performance as seen in the Boston Housing data example. On the other hand, it is known to work well in many situations. In complex multivariate problems, this is always an issue that pops up and there isn't any clear answer to this problem. The best thing is to try both versions and see the difference. In all our data examples with PSVMMIX(ν_β), we assumed ν_β is a scalar. However, as with the linear case, this could very well be a vector. Further research would be needed to figure out an efficient method to estimate the extra tuning parameters. We are currently working on this and hope to report on this at KDD2003.

Next, we showed that the statistical interpretations enable us to extend methodologies from the classification literature to the regression scenario without any extra effort. We also demonstrate the usefulness of attaching an estimate of variability to our predictions.

To sum up, we feel the fundamental contribution of this paper is to show that some well known KDD algorithms turn out to be equivalent to some commonly used shrinkage estimation techniques used in statistics. This equivalence opens up new research opportunities both for statisticians and computer scientists. The challenge to the statistical community is to develop methodologies that can work for massive datasets by taking advantage of the KDD literature. On the other hand, the statistical insights should help the computer science community better understand their algorithms and improve upon them.

7. ACKNOWLEDGMENTS

I thank William DuMouchel for all his support and help without which this work would not be possible. It was Bill who initiated this work by pointing out the equivalence between PSVM with linear kernel and ridge regression. I thank Daryl Pregibon for stimulating discussions from time to time that helped improve the quality of the paper. Finally, I thank Professor C.F.Sirmans and T.G.Thibedeau for providing the Dallas house price data.

8. REFERENCES

[1] US census bureau. Adult dataset. Publicly available from: www.sgi.com/Technology/mlc/db.

[2] D. Agarwal. Bayesian spatial regression analysis with large datasets, ph.d dissertation, university of connecticut, www.research.att.com/∼dagarwal. 2001.

[3] J. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88:669–679, 1993.

[4] H. Arthur and K. Robert. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

[5] L. Brieman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

[6] D. DeCoste. Visualizing mercer kernel feature spaces via kernelized locally-linear embeddings. In *The 8th International Conference on Neural Information Processing*, 2001.

[7] A. Dempster, M. Schatzoff, and N. Wermuth. A simulation study of alternatives to ordinary least squares. *Journal of the American Statistical Association*, 72:77–106, 1977.

[8] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Michael C. Mozer, Michael I. Jordan, and Thomas Petsche editors, Advances in Neural Information Processing Systems -9-*, pages 155–161. The MIT Press, Cambridge, MA, 1997.

[9] K. Duan, S. Keerthi, and A. Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Technical Report, Department of Mechanical Engineering, National University of Singapore*, 2001.

[10] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 67–76. San Francisco, CA, USA, August 2001.

[11] B. Efron and C. Morris. Data analysis using stein's estimator and its generalizations. *Journal of the American Statistical Association*, 70:311–319, 1975.

[12] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.

[13] G. Fung and O. Mangasarian. Proximal support vector machine classifiers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86. San Francisco, CA, USA, August 2001.

[14] J. Garcke and M. Griebel. Data mining with sparse grids using simplicial basis functions. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 87–96. San Francisco, CA, USA, August 2001.

[15] A. Gelfand and A. Smith. Sampling based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

[16] M. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2000.

[17] D. Gibbons. A simulation study of some ridge estimators. *Journal of the American Statistical Association*, 76:131–139, 1981.

[18] J. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356, 1992.

[19] T. Hsaing. A bayesian view on ridge regression. *The Statistician*, 24:267–268, 1975.

[20] S. le Cessie and J. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41:191–201, 1992.

[21] A. Lee and M. Silvapulle. Ridge estimation in logistic regression. *Communications in Statistics, Part B–Simulation and Computation*, 17:1231–1257, 1988.

[22] Y. Lee and O. Mangasarian. Ssvm: A smooth support vector machine. *Computational Optimization and Applications*, 20(1):to appear, 2001.

[23] M. J. Mackinnon and M. L. Puterman. Collinearity in generalized linear models. *Communications in Statistics, Part A – Theory and Methods*, 18:3463–3472, 1989.

[24] O. Mangasarian. Generalized support vector machines. In *A. Smola, P. Bartlett, B.Schölkopf, and D.Schuurmans, editors, Advances in Large Margin Classifiers*, pages 135–146. The MIT Press, Cambridge, MA, 2000.

[25] B. Marx, P. Eilers, and E. Smith. Ridge likelihood estimation for generalized linear regression. In *P. van der Heijden, W. Jensen, B. Francis, and G. Seeber, editors, Statistical Modeling*, pages 227–238. North Holland Publishing Company (Elseviers), Amsterdam, 1992.

[26] P. Murphy and D. Aha. UCI repository of machine learning databases. www.ics.uci.edu/∼mlearn/MLRepository.html.

[27] B. Segerstedt. On ordinary ridge regression in generalized linear models. *Communications in Statistics, Part A – Theory and Methods*, 21:2227–2246, 1992.

[28] G. Smith and F. Campbell. A critique of some ridge regression methods (with discussion). *Journal of the American Statistical Association*, 75:74–81, 1980.

[29] D. Stewart and Z. Leyk. Meschach: Matrix computations in C. www.netlib.org/c/meschach/.

[30] R. Thisted. Comments on 'a critique of some ridge regression methods'. *Journal of the American Statistical Association*, 75:81–86, 1980.