Journal of Machine Learning Research 3 (2003) 1229-1243 Submitted 5/02; Published 3/03

Dimensionality Reduction via Sparse Support Vector Machines

Jinbo Bi BIJ2@RPI.EDU
Kristin P. Bennett BENNEK@RPI.EDU
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
Mark Embrechts EMBREM@RPI.EDU
Department of Decision Science and Engineering Systems
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
Curt M. Breneman BRENEC@RPI.EDU
Minghu Song SONGM@RPI.EDU
Department of Chemistry
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Editors: Isabelle Guyon and André Elisseeff

Abstract

We describe a methodology for performing variable ranking and selection using support vector machines (SVMs). The method constructs a series of sparse linear SVMs to generate linear models that can generalize well, and uses a subset of nonzero weighted variables found by the linear models to produce a final nonlinear model. The method exploits the fact that a linear SVM (no kernels) with ℓ_1-norm regularization inherently performs variable selection as a side-effect of minimizing capacity of the SVM model. The distribution of the linear model weights provides a mechanism for ranking and interpreting the effects of variables. Starplots are used to visualize the magnitude and variance of the weights for each variable. We illustrate the effectiveness of the methodology on synthetic data, benchmark problems, and challenging regression problems in drug design. This method can dramatically reduce the number of variables and outperforms SVMs trained using all attributes and using the attributes selected according to correlation coefficients. The visualization of the resulting models is useful for understanding the role of underlying variables.

Keywords: Variable Selection, Dimensionality Reduction, Support Vector Machines, Regression, Pattern Search, Bootstrap Aggregation, Model Visualization

1. Introduction

Variable selection refers to the problem of selecting input variables that are most predictive of a given outcome. Appropriate variable selection can enhance the effectiveness and domain interpretability of an inference model. Variable selection problems are found in many supervised and unsupervised machine learning tasks including classification, regression, time series prediction, clustering, etc. We shall focus on supervised regression tasks, but the general methodology can be extended to any inference task that can be formulated as an ℓ_1-norm SVM, such as classification and novelty detection (Campbell and Bennett, 2000, Bennett and Bredensteiner, 1997). The objective of variable selection is two-fold: improving prediction performance (Kittler, 1986) and enhancing understanding of the underlying concepts in the induction model.

Our variable selection methodology for SVMs was created to address challenging problems in Quantitative Structural-Activity Relationships (QSAR) analysis. The goal of QSAR analysis is to predict the bioactivity of molecules. Each molecule has many potential descriptors (300-1000) that may be highly correlated with each other or irrelevant to the target bioactivity. The bioactivity is known for only a few molecules (30-200). These issues make model validation challenging and overfitting easy. The results of the SVMs are somewhat unstable – small changes in the training and validation data or on model parameters may produce rather different sets of nonzero weighted attributes (Breneman et al., 2002). Our variable selection and ranking methodology exploits this instability. Computational costs are not a primary issue in our experiments due to lack of data. Our method is based on sparse SVMs, so we call the algorithm VS-SSVM for Variable Selection via Sparse SVMs.

Variable selection is a search problem, with each state in the search space specifying a subset of the possible attributes of the task. Exhaustive evaluation of all variable subsets is usually intractable. Genetic algorithms, population-based learning, and related Bayesian methods have been commonly used as search engines for the variable selection process (Inza et al., 1999, Yang and Honavar, 1997, Kudo et al., 2000). Particularly for SVMs, a variable selection method was introduced (Weston et al., 2000) based on finding the variables that minimize bounds on the leave-one-out error for classification. The search of variable subsets can be efficiently performed by a gradient descent algorithm. The method, however, was limited to separable classification problems, and thus is not directly applicable to the regression problems examined in this paper. Guyon et al. proposed another variable selection method for classification by recursively eliminating the input variables that decrease the margin the least (Guyon et al., 2002). A generic wrapper approach based on sensitivity analysis has been applied to kernel SVM regression (SVR) (Embrechts et al., 2001) but it is more computationally intensive than our proposed approach.

Variable selection methods are often divided along two lines: filter and wrapper methods (Kohavi and John, 1997). The filter approach of selecting variables serves as a preprocessing step to the induction. The main disadvantage of the filter approach is that it totally ignores the effects of the selected variable subset on the performance of the induction algorithm. The wrapper method searches through the space of variable subsets using the estimated accuracy from an induction algorithm as the measure of "goodness" for a particular variable subset. Thus, the variable selection is being "wrapped around" a particular induction algorithm. These methods have encountered some success with induction tasks, but they can be very computationally expensive for tasks with a large number of variables.

Our approach (VS-SSVM) consists largely of two consecutive parts: variable selection and nonlinear induction. The selection of variables serves as a preprocessing step to the final kernel SVR induction. The variable selection itself is performed by wrapping around linear SVMs (no kernels) with sparse norm regularization. Such sparse linear SVMs are constructed to both identify variable subsets and assess their relevance in a computationally cheaper way compared with a direct wrap around nonlinear SVMs. However, the variable selection by linear SVMs and the final nonlinear SVM inference are tightly coupled since they both employ the same loss function. Our method is similar in spirit to the Least Absolute Shrinkage and Selection Operator (LASSO) method (Tibshirani, 1996) but is specifically targeted to SVR with the ε-insensitive loss function.

This article is organized as follows. In Section 2, we review sparse SVMs with ℓ_1-norm regularization, specifically, the sparse ν-SVR. Section 3 provides details on the VS-SSVM algorithm based on sparse linear SVMs. Sections 4 and 5 compare VS-SSVM with stepwise dimensionality reduction and correlation coefficient ranking methods on synthetic data and Boston Housing data. Model visualization is also explored in Section 5 to reveal domain insights. Computational results on real-life QSAR data are included in Section 6.

2. Sparse Support Vector Machines

In this section, we investigate sparse SVMs. Consider the regression problem as finding a function $f^* \in \mathcal{F} = \{f : \mathbb{R}^n \to \mathbb{R}\}$ that minimizes the regularized risk functional (Boser et al., 1992, Vapnik, 1995, Smola, 1998): $R[f] := \mathbf{P}[f] + C\frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i, f(\mathbf{x}_i))$, where $L(\cdot)$ is a loss function. Usually the ε-insensitive loss $L_\varepsilon(y, f(\mathbf{x})) = \max\{|y - f(\mathbf{x})| - \varepsilon, 0\}$ is used in SVR. $\mathbf{P}[\cdot]$ is a regularization operator and C is called the regularization parameter. For linear functions, $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} + b$, the regularization operator in classic SVMs is the squared ℓ_2-norm of the normal vector $\mathbf{w}$. Nonlinear functions are produced by mapping $\mathbf{x}$ to $\Phi(\mathbf{x})$ in a feature space via the kernel function k and constructing linear functions in the feature space. A linear function in feature space corresponds to a nonlinear function in the original input space. The optimal solution $\mathbf{w}$ to SVMs can be expressed as a support vector expansion $\mathbf{w} = \sum \alpha_i \Phi(\mathbf{x}_i)$. Thus, the regression function can be equivalently expressed as a kernel expansion $f(\mathbf{x}) = \sum \alpha_i \Phi(\mathbf{x}_i)'\Phi(\mathbf{x}) + b = \sum \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$. Classic SVMs are quadratic programs (QPs) in terms of α.

Solving QPs is typically computationally more expensive than solving linear programs (LPs). SVMs can be transformed into LPs as in Bennett (1999), Breiman (1999) and Smola et al. (1999). This is achieved by regularizing with a sparse norm, e.g. the ℓ_1-norm. This technique is also used in basis pursuit (Chen et al., 1995), parsimonious least norm approximation (Bradley et al., 1998), and LASSO (Tibshirani, 1996). Instead of choosing the "flattest" function as in classic SVR, we directly apply the ℓ_1-norm to the coefficient vector α in the kernel expansion of f. The regularized risk functional is then specified as

$$R[f] := \sum_{i=1}^{\ell} |\alpha_i| + C\frac{1}{\ell}\sum_{i=1}^{\ell} L_\varepsilon(y_i, f(\mathbf{x}_i)). \tag{1}$$

This is referred to as a "sparse" SVM because the optimal solution $\mathbf{w}$ is usually constructed based on fewer training examples $\mathbf{x}_i$ than in classic SVMs and thus the function f requires fewer kernel entries $k(\mathbf{x}_i, \mathbf{x})$.

The classic SVR approach has two hyper-parameters C and ε. The tube parameter ε can be difficult to select as one does not know beforehand how accurately the function will fit. The ν-SVR (Schölkopf et al., 2000, Smola et al., 1999) was developed to automatically adjust the tube size, ε, by using a parameter $\nu \in (0, 1]$. The parameter ν provides an upper bound on the fraction of error examples and a lower bound on the fraction of support vectors. To form the ν-SVR LP, we rewrite $\alpha_j = u_j - v_j$ where $u_j,\ v_j \geq 0$. The solution has either u_j or v_j equal to 0, depending on the sign of α_j, so $|\alpha_j| = u_j + v_j$. Let the training data be $(\mathbf{x}_i, y_i),\ i = 1, \cdots, \ell$ where $\mathbf{x}_i \in \mathbb{R}^n$ with the j^{th}

component of $\mathbf{x}_i$ denoted as $x_{ij},\ j = 1, \cdots, n$. The LP is formulated in variables $\mathbf{u}$, $\mathbf{v}$, b, ε, ξ and η as

$$\begin{array}{lll}
\min & \sum_{j=1}^{\ell} (u_j + v_j) + C\frac{1}{\ell}\sum_{i=1}^{\ell}(\xi_i + \eta_i) + C\nu\varepsilon & \\
\text{such that} & y_i - \sum_{j=1}^{\ell}(u_j - v_j)k(\mathbf{x}_i, \mathbf{x}_j) - b \leq \varepsilon + \xi_i, & i = 1, \ldots, \ell, \\
& \sum_{j=1}^{\ell}(u_j - v_j)k(\mathbf{x}_i, \mathbf{x}_j) + b - y_i \leq \varepsilon + \eta_i, & i = 1, \ldots, \ell, \\
& u_j,\ v_j,\ \xi_i,\ \eta_i,\ \varepsilon \geq 0, & i, j = 1, \ldots, \ell.
\end{array} \tag{2}$$

LP (2) provides the basis of both our variable selection and modeling methods. To select variables effectively, we employ a sparse linear SVR which is formulated from LP (2) simply by replacing $k(\mathbf{x}_i, \mathbf{x}_j)$ by x_{ij} with index i running over examples and index j running over variables. The optimal solution is then given by $\mathbf{w} = \mathbf{u} - \mathbf{v}$. To construct the final nonlinear model, we use the sparse nonlinear SVR LP(2) with a nonlinear kernel such as the RBF kernel $k(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{z}\|^2}{\sigma^2}\right)$. The optimal solution is then given by $\alpha = \mathbf{u} - \mathbf{v}$.

3. The VS-SSVM Algorithm

We briefly describe the VS-SSVM algorithm in this section. The VS-SSVM algorithm consists of 5 essential components: 1. A linear model with sparse $\mathbf{w}$ constructed by solving a linear SVM LP to obtain a subset of variables nonzero-weighted in the linear model; 2. An efficient search for optimal hyper-parameters C and ν in the linear SVM LP using "pattern search"; 3. The use of bagging to reduce the variability of variable selection; 4. A method for discarding the least significant variables by comparing them to "random" variables; 5. A nonlinear regression model created by training and bagging the LPs (2) with RBF kernels on the final subset of variables selected. We shall explain the various components in more detail in this section.[1] Pseudocode for the first four steps is given in Algorithm 1 in the appendix. Algorithm 2 in the appendix describes the final nonlinear regression modeling algorithm. In Section 5, we describe how further filtering of variables can be achieved by visualizing the bagged solutions and applying rules to the bagged models.

Sparse linear models: Sparse linear models are constructed using the following LP:

$$\begin{array}{ll}
\min & \sum_{j=1}^{n} (u_j + v_j) + C\frac{1}{\ell}\sum_{i=1}^{\ell}(\xi_i + \eta_i) + C\nu\varepsilon \\
\text{such that} & y_i - \sum_{j=1}^{n}(u_j - v_j)x_{ij} - b \leq \varepsilon + \xi_i, \quad i = 1, \ldots, \ell, \\
& \sum_{j=1}^{n}(u_j - v_j)x_{ij} + b - y_i \leq \varepsilon + \eta_i, \quad i = 1, \ldots, \ell, \\
& u_j,\ v_j,\ \xi_i,\ \eta_i,\ \varepsilon \geq 0, \quad i = 1, \ldots, \ell, \quad j = 1, \ldots, n.
\end{array} \tag{3}$$

Let $\mathbf{w} = \mathbf{u} - \mathbf{v}$ be the solution to the linear SVR LP (3). The magnitude and sign of the component w_j indicates the effect of the j^{th} variable on the model. If $w_j > 0$, the variable contributes to y; if $w_j < 0$, the variable reduces y. The ℓ_1-norm of $\mathbf{w}$ inherently enforces sparseness of the solution. Roughly speaking, the vectors further from the coordinate axes are "larger" with respect to the ℓ_1-norm than with respect to ℓ_p-norms with $p > 1$. For example, consider the vectors

1. More details are available at the website *http://www.rpi.edu/~bennek/featsele.html*

$(1,0)$ and $(1/\sqrt{2},1/\sqrt{2})$. For the ℓ_2-norm, $\|(1,0)\|_2 = \|(1/\sqrt{2},1/\sqrt{2})\|_2 = 1$, but for the ℓ_1-norm, $1 = \|(1,0)\|_1 < \|(1/\sqrt{2},1/\sqrt{2})\|_1 = \sqrt{2}$. The degree of sparsity of the solution $\mathbf{w}$ depends on the regularization parameter C and the tube parameter ν in LP(3).

Pattern search: Since the hyper-parameters play a crucial role in our variable selection approach, we optimize them using a pattern search approach. This optimization is automatically performed based on validation set results by applying the derivative-free pattern search method (Dennis and Torczon, 1994) in the C-ν search space. For each choice of C-ν, LP (3) generates a linear model based on the training data. Then the resulting model is applied to the validation data and evaluated using the statistic $Q^2 = \frac{\sum(y_i-\hat{y}_i)^2}{\sum(y_i-\bar{y})^2}$, the mean squared error scaled by the variance of the response, where $\hat{y}_i$ is the prediction of y_i for the i^{th} validation example and $\bar{y}$ is the mean of the actual responses. The pattern search method optimizes this validation Q^2 over the C-ν space. A good range for the hyper-parameters may be problem-specific, but we prefer to a generic approach applicable to most datasets. Hence a reasonably large range is adopted to produce the C-ν space, specifically, $C \in [e^{-2} = 0.1353, e^{10} = 22026]$ and $\nu \in [0.02, 0.6]$.

The pattern search algorithm is embedded in Algorithm 1 as a sub-routine. Each iteration of a pattern search algorithm starts with a center (initially randomly chosen), samples other points around the center in the search space, and calculates objective values of each neighboring point until it finds a point with objective value less than that of the center. The algorithm then moves the center to the new minimizer. If all the points around the center fail to bring a decrease to the objective, the search step (used to determine the neighboring points) is reduced by half. This search continues until the search step gets sufficiently small, thus ensuring convergence to a local minimizer. For a full explanation of pattern search for SVR see Momma and Bennett (2002).

Variability reduction: The optimal weight vectors $\mathbf{w}$ for LP (3) exhibit considerable variance due to local minima in the pattern search, the small dataset size, and changes in validation data. Different partitions of data may produce very different answers. No individual model can be considered completely reliable, especially for QSAR data. Thus "bootstrap aggregation" or "bagging" is used to make the procedure more stable (Breiman, 1996). In our experiments, models were constructed based on $T = 20$ random partitions to produce distinct weight vectors. There are several schemes to combine models. We took the superset of nonzero weighted variables obtained in any of the 20 different partitions – the "bagged" subset of the variables. Bagging can augment the performance of various individual models due to the reduced variance of the "bagged" model (Breiman, 1996). For problems with large variance, regression based on the average usually outperforms any single model. We use bagging for both variable selection and nonlinear SVR modeling.

Discarding least significant variables: Sparsity of the linear model eliminates many variables in every bootstrap, but it is possible that in any given bootstrap irrelevant variables are included. Thus, we also eliminate variables by introducing random gauge variables. The intuition is that if an independent variable has even less significance than a random variable which is barely related to the response, then it may be safely deleted. The VS-SSVM Algorithm 1 first augments the data with normally distributed random variables (mean 0 and standard deviation 1). Random variables following other distributions can also be employed. See Stoppiglia and Dreyfus (2003) for a more thorough discussion of this issue. Based on the previous empirical results by Embrechts et al. (2001), we added 3 random variables with sample correlations to the response less than 0.13 in magnitude. The weights $\mathbf{w}$ for these random variables provide clues for thresholding the selection of variables based on the average of the weights on the 3 variables across all the bootstraps. Only variables with average weight greater than this average will be selected. The selection of variables can be further refined. We leave the explanation of our scheme for model visualization and further filtering of the variables until Section 5.

Nonlinear SVR models: After the final set of variables is selected, we employ Algorithm 2 in the appendix to construct the nonlinear SVR model. Nonlinear models were constructed based on $T = 10$ partitions and then averaged to produce the final model. We focus on evaluating the performance of the variable selection method more than optimizing the predictor. Hence a simple grid search was used in nonlinear SVR modeling to select hyper-parameters rather than pattern search in each fold of the bagging. In the grid search, we considered only the RBF kernels with parameter σ^2 equal to 8, 100, 150, 250, 500, 1000, 3000, 5000, and 10000. The parameter C was chosen from values between 10 and 20000 with 100 as the increment within 1000 and 1000 as the increment from 1000 to 20000, and the parameter ν from 0.1, 0.15, 0.2, 0.3, and 0.5.

4. Computational Analysis of VS-SSVM

We evaluated the computational effectiveness of VS-SSVM on synthetic data and the benchmark Boston Housing problem (Harrison and Rubinfeld, 1978). Our goal was to examine whether VS-SSVM can improve generalization. The LPs (2) and (3) formulated on training data were both solved using CPLEX version 6.6 (ILOG, 1999).

We compared VS-SSVM with a widely used method, "stepwise regression" wrapped around Generalized Linear Models (GLM) (Miller, 1990). For fair comparison, the GLM models were also bagged. Hence 20 different GLM models were generated using Splus 2000 (Venables and Ripley, 1994, McCullagh and Nelder, 1983) based on different bootstrapped samples, and the resulting models were bagged. The bagged models performed at least as well as single models, so only the bagged results are presented here. In each trial, half of the examples were held out for test and the other half were used in training. VS-SSVM and stepwise regression were run on the training data. The training data were further divided to create a validation set in each fold of the bagging scheme. The final models were applied to the hold-out test data in order to compute the test Q^2. This procedure was repeated 20 times on different training-test splits of the data for both methods.

The synthetic data set was randomly generated with solution pre-specified as follows: there are 12 independent variables and 1 response variable. The first 5 variables, $x_1, \ldots, x_5$, were drawn independently and identically distributed from the standard normal distribution. The 6^{th} variable was $x_6 = x_1 + 1$ which is correlated to x_1. The 7^{th} variable was $x_7 = x_2 x_3$ which relates to both x_2 and x_3. Five additional standard normally distributed variables were also generated, and had nothing to do with the dependent y. We call them noise variables NV_1 to NV_5. The y was calculated as $y = x_1 + 2x_2 + 3x_3 + x_4^2 + e^{x_5}$. We generated 200 examples.

Table 1(left) shows that VS-SSVM consistently selected all the desired variables x_1 to x_5, and discarded most of the irrelevant attributes NV_i. Since the randomly-generated attributes NV_i can be correlated with the response by chance, they can not always be eliminated. Thresholding based on the maximum weight rather than the average weight on the random variables added in Algorithm 1 can eliminate all NVs, but this may also remove relevant variables. VS-SSVM selected exactly one of x_1 and $x_1 + 1$ in each fold since either could be used to represent the function, but taking the superset over all folds brought both variables into the selected set. Fortunately, such highly correlated variables can be directly filtered as in Section 5. For nonlinear SVR, it is better to have many variables than too few, and correlated variables can be beneficial when noise is presented.

Table 1: A comparison of selected variable subsets by VS-SSVM (*left*) and stepwise (*right*) within 20 trials. The variables x_1, x_2, x_3 were selected in all trials by both methods. $NV_1 \sim NV_5$ means all 5 noise variables.

Times	Variable subsets
6	$x_4,\ x_5,\ x_1+1,\ NV_2$
3	$x_4,\ x_5,\ x_1+1,\ NV_1,\ NV_5$
3	$x_5,\ x_1+1$
2	$x_4,\ x_5,\ x_1+1,\ x_2x_3,\ NV_5$
1	$x_4,\ x_5$
1	$x_4,\ x_5,\ x_1+1,\ NV_5$
1	$x_5,\ NV_2$
1	$x_5,\ x_1+1,\ NV_1,\ NV_5$
1	$x_5,\ x_1+1,\ NV_2,\ NV_3,\ NV_5$
1	$x_4,\ x_5, x_1+1,\ NV_1 \sim NV_5$

Times	Variable subsets
2	$x_4,\ x_5,\ x_2x_3,\ NV_1,\ NV_5$
2	$x_4,\ x_5,\ NV_1,\ NV_5$
2	$x_4,\ x_5,\ NV_1,\ NV_5$
2	$x_5,\ x_2x_3,\ NV_2,\ NV_4,\ NV_5$
2	$x_5,\ NV_1,\ NV_2,\ NV_4,\ NV_5$
1	$x_4,\ x_5,\ NV_2,\ NV_3,\ NV_4$
1	$x_4,\ x_5,\ NV_2,\ NV_4,\ NV_5$
1	$x_4,\ x_5,\ NV_5$
1	$x_4,\ x_5,\ x_2x_3,\ NV_4,\ NV_5$
1	$x_4,\ x_5,\ NV_1,\ NV_2,\ NV_4,\ NV_5$
1	$x_5,\ NV_2,\ NV_5$
1	$x_5,\ NV_2,\ NV_3,\ NV_4$
1	$x_5,\ x_2x_3$
1	$x_5,\ x_2x_3,\ NV_2$
1	$x_5,\ NV_3,\ NV_5$

The results for stepwise regression are summarized in Table 1(right). The Splus GLM modeling was constructed for the Gaussian family. We experimented with the quadratic form of GLM but it constructed a function of many nuisance variables like NV_2^2. Therefore GLM was restricted to linear models. Note that x_1+1 never appeared in the model, so stepwise regression seemed to handle linearly-correlated variables better than VS-SSVM. However, it was likely to pick the nonlinearly-interrelated variable x_2x_3 and the irrelevant variables NVs. Moreover it missed the desired variable x_4 more times than did VS-SSVM. The stepwise regression method is computationally expensive for problems with many attributes. The Splus GLM modeling could not be applied to the QSAR problems in Section 6 since these problems have fewer sample points than variables.

VS-SSVM generalized better than stepwise regression. Figure 1 depicts the observed versus the predicted responses of the test examples over all 20 trials on synthetic data for VS-SSVM and GLM. Each point is represented by a vertical bar with the middle point representing the mean and the length of the bar drawn according to the standard deviation of predictions for that point. VS-SSVM obtains a test Q^2 of $.0332 \pm .0027$ and stepwise GLM achieves a Q^2 of $.1287 \pm .0065$. Recall that Q^2 is proportional to the mean squared error so smaller values are better. The squared correlation coefficient, r^2, between the observed and predicted responses is also provided in the figures, and the larger r^2, the better. Figure 2 summarizes the results on the Boston Housing Data, for which all variables were selected by VS-SSVM and stepwise regression in most of the 20 trials.

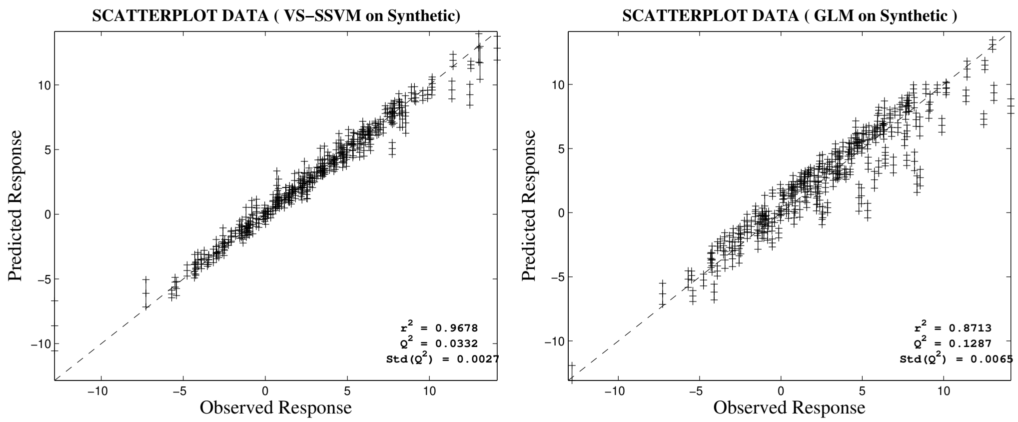

Figure 1: On synthetic data; *left*: VS-SSVM result; *right*: stepwise regression result.

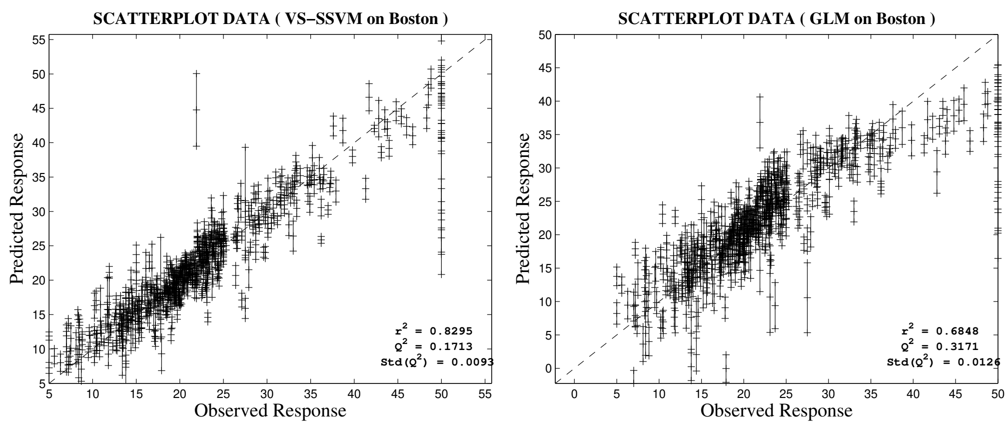

Figure 2: On Boston Housing data; *left*: VS-SSVM result; *right*: stepwise regression result.

5. Model Visualization

Model visualization can be used to interpret the results of the VS-SSVM. Recall that even within a single trial, VS-SSVM produces different linear models using different partitions/bootstraps of data. The final variable subset is the aggregate of these models. Examining the distribution of the weights on each variable in different bootstrap models can yield valuable insights into the relationship between the independent and response variables. Visualization techniques such as starplots, histograms, and stackplots can enhance understanding of the role of the variables (Fayyad et al., 2001).

We constructed a starplot for each variable. The starplot consists of a sequence of equi-angular spokes (radials). Each spoke represents the variable's weight in a different bootstrap model. We did 20 bootstraps, so there are 20 spokes. The spokes are ordered ascendingly by the $||\mathbf{w}||$ corresponding to each bootstrap. The $\mathbf{w}$ for each bootstrap is normalized such that for each component, $|w_j| \in [0, 1]$.

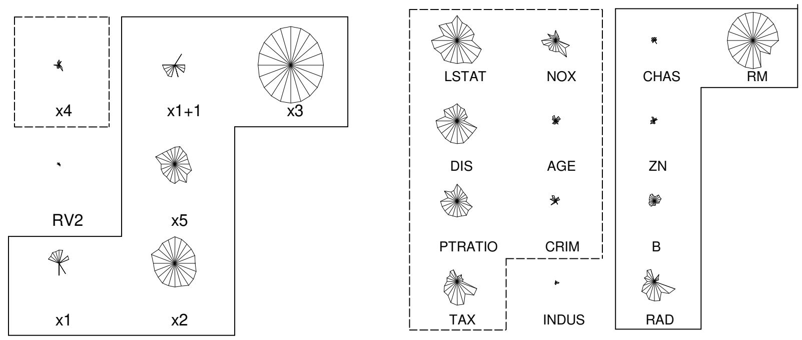

Figure 3: Starplots for *left*: the synthetic data and *right*: the Boston Housing data. Dotted and solid boxes contain non-positive and non-negative weighted variables.

A line connecting the spokes gives the starplot its name. The relative size and shape of starplots allow one to quickly assess relationships beyond simple mean and variance across the bootstraps.

Figure 3(left) is a representative starplot for the synthetic data. The starplots within the dashed box represent those variables with negative weights **w**. The starplots within the solid box represent variables with positive weights. The remaining variables have weights that flip signs. The stars are ordered according to the average weight for each variable on different bootstraps. Variable x_3 was the most positively-linearly related to the response, and x_2 the second, which reflects the truth as $y = x_1 + 2x_2 + 3x_3 + x_4^2 + e^{x_5}$. By only fitting linear functions, VS-SSVM may not always detect nonlinearly-related variables correctly. In our experiments, it did detect the nonlinear variables x_4 and x_5. Note that since x_4^2 is part of the true model, the sign of x_4 is negative, showing that the linear model focuses on correcting the negative values of x_4. The NV_2 has no clear relation with the response since the weights for NV_2 flip signs among different models. The strategy of removing variables with flipping signs can further refine variable selection.

The variables with flipping signs do not always coincide with the variables that are least correlated with the response. On the synthetic data the correlation coefficients r of the variables and the response are:

x_1	x_2	x_3	x_4	x_5	x_1+1	NV_2
0.233	0.498	0.711	-0.077	0.506	0.233	-0.080

Removing all variables with r less than 0.08 in magnitude would also delete x_4.

The size and shape of the starplots provide information about the models. The starplots for variable x_1 and x_1+1 are complementary, which means if a bootstrap model selects x_1, it does not include x_1+1. The starplots help spot such complementary behavior in models. Hence highly correlated variables can be further filtered.

VS-SSVM with model visualization can be valuable even on datasets where variable selection does not eliminate variables. For example, on the Boston Housing data, VS-SSVM does not drop any variables. However, the weights produced by VS-SSVM can help understand the role and the relative importance of the variables in the model. The starplots based on the entire Boston Housing data are given in Figure 3(right). They are drawn in the same way as for the synthetic data. For instance, the RM (average number of rooms per dwelling) is the most positively related to the housing price, which reflects that the number of rooms is important for determining the housing price and the more rooms the house has, the higher the price. The INDUS (proportion of non-retail business acres per town) appears not to affect the housing price significantly in the linear modeling since the corresponding weights flip signs.

6. Generalization Testing on QSAR Datasets

VS-SSVM was also tested on challenging real-life QSAR data. The QSAR data were created in the ongoing NSF-funded Drug Design and Semi-supervised Learning (DDASSL) project (See `http://www.drugmining.com`). Leave-one-out cross-validation was performed for QSAR data. As described in Section 3, variables were selected separately for each left-out point. We used exactly the same group of examples for both experiments with and without variable selection.

Table 2: A summary of data and reduced data.

Dataset	# of Obs.	Original # of Vars.	Preproc. # of Vars.	1st VS # of Vars	2nd VS # of Vars
Aquasol	197	640	525	118	57
Blood/Brain Barrier	62	694	569	64	51
Cancer	46	769	362	73	34
Cholecystokinin	66	626	350	93	69
HIV	64	620	561	53	17
Caco2	27	715	713	79	41

Table 2 summarizes the datasets. Variables with a range greater than 4 standard deviations were removed (a common practice in commercial analytical tools used in chemometrics). This very primitive form of variable filtering rarely hurts and usually improves the results. The resulting numbers of variables are in the 3rd column of Table 2. VS-SSVM greatly reduced the number of attributes as shown in columns 4 and 5 of Table 2 while improving generalization (see Table 3). Column 5 (Table 2) gives the results obtained by iteratively applying VS-SSVM to the original or reduced data until no variables have weights that flip signs.

Table 3 gives a comparison of the results based on all attributes versus those selected by one iteration of VS-SSVM. We compared VS-SSVM to the correlation coefficient ranking method. The ranking method chose the q variables most correlated to the response. The number q was chosen to be the same number of variables selected by VS-SSVM. After the variables were selected, Algorithm 2 was used to construct the final nonlinear model. Table 4 presents the results for VS-SSVM after iteratively removing variables with flipping signs versus the results for the correlation coefficient ranking. The squared correlation coefficient between the actual and predicted responses (1 is the best) and leave-one-out Q^2 (0 is the best) are reported. The standard deviation of Q^2, Std(Q^2), is computed as the standard deviation of the squared errors on the test data scaled by the variance of the actual response. By cross referencing with Table 3, Table 4 shows that the ranking scheme by correlation coefficients failed to improve the generalization performance. The significance of the differences was assessed using a paired t-test. We calculated the mean of the error $\varepsilon = |y - \hat{y}|$ and

Table 3: The experimental results on full datasets and reduced datasets obtained by VS-SSVM. The 1st VS-SSVM means to run VS-SSVM once on the full data.

	Full Data			1st VS-SSVM reduced data		
Dataset	r^2	Q^2	Std(Q^2)	r^2	Q^2	Std(Q^2)
Aquasol	0.918	0.082	0.006	0.929	**0.071**	0.018
Blood/Brain Barrier	0.693	0.310	0.079	0.719	**0.286**	0.074
Cancer	0.507	0.500	0.171	0.779	**0.223**	0.102
Cholecystokinin	0.616	0.404	0.087	0.673	**0.332**	0.068
HIV	0.551	0.458	0.086	0.652	**0.357**	0.066
Caco2	0.693	0.326	0.083	0.736	**0.300**	0.069

Table 4: The comparison of VS-SSVM by iteratively eliminating "flipped" variables and the correlation coefficient ranking.

	2nd VS-SSVM			corr. coef. rank		
Dataset	r^2	Q^2	Std(Q^2)	r^2	Q^2	Std(Q^2)
Aquasol	0.936	**0.065**	0.010	0.908	0.092	0.006
Blood/Brain Barrier	0.718	**0.283**	0.076	0.693	0.300	0.017
Cancer	0.836	**0.164**	0.035	0.820	0.185	0.020
Cholecystokinin	0.643	0.370	0.073	0.650	**0.350**	0.020
HIV	0.612	**0.396**	0.087	0.592	0.415	0.025
Caco2	0.725	**0.293**	0.074	0.672	0.351	0.046

performed the paired t-test on the squared errors ε^2. The results are shown in Table 5. The absolute errors for modeling with full data and with variable reduction are denoted as ε_f and ε_r respectively.

From all these tables we conclude that VS-SSVM is effective at reducing the dimensionality on QSAR problems. The first phase of VS-SSVM significantly reduced the number of variables

Table 5: The experimental results for the paired t-test.

Dataset	mean ε_f	mean ε_r	t-statistic	p-value
Aquasol	0.489	0.458	0.469	0.640
Blood/Brain Barrier	0.331	0.305	1.334	**0.187**
Cancer	0.431	0.292	2.506	**0.016**
Cholecystokinin	0.728	0.698	0.577	0.566
HIV	0.721	0.609	2.446	**0.017**
Caco2	0.643	0.636	0.052	0.959

and further reductions were achieved by iteratively feeding QSAR data to VS-SSVM and removing variables with flipping signs in the linear models. VS-SSVM either produced significantly better (3 of the 6 problems by Table 5) or no worse generalization accuracy using dramatically fewer variables. The most significant improvements were obtained on the Cancer and HIV data. Plots of the actual versus predicted responses for Cancer data in Figure 4 illustrate the improved generalization obtained by VS-SSVM.

7. Conclusions and Discussion

The key components of our variable selection approach are first, exploiting the inherent selection of variables done by the sparse linear SVM LP(3), second, aggregation of many sparse models to overcome the unreliability of any single model, and third, visualization or analysis of bagged models to discover trends. In this research, we only investigated a sparse SVM regression algorithm, but any sparse modeling process can serve a similar function. We focused on starplots for model visualization, but other visualization methods can also be applied and may be more informative. For instance, we found parallel coordinate plots of variable weights versus bootstraps to be valuable.

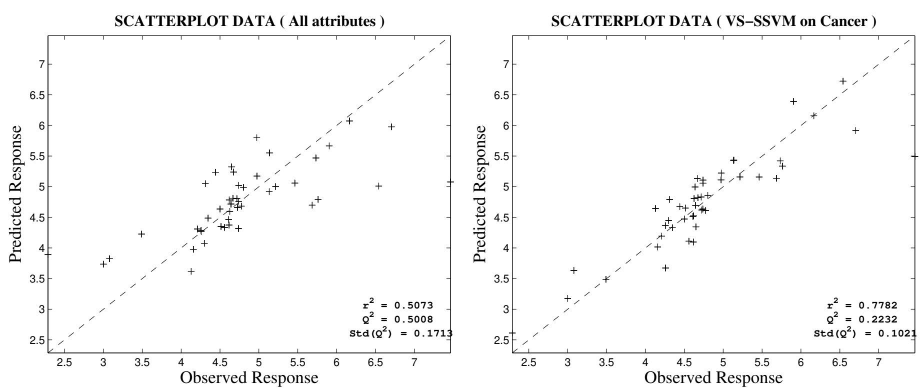

Figure 4: *Left*: Cancer full data; *right*: Cancer with variable selection.

VS-SSVM proved to be very effective on problems in drug design. The number of variables was dramatically reduced while maintaining or even improving the generalization ability. This method outperforms SVMs trained using all the attributes and the attributes selected by correlation ranking. Chemists have found model visualization to be useful both for guiding the modeling process and for interpreting the effects of the descriptors used in the models (Song et al., 2002). Through model visualization, we discovered the simple rule of eliminating variables with weights that flip signs in distinct individual models. Automating this rule proved to be a valuable heuristic for further refining variable selection.

VS-SSVM is not a general methodology suitable for all types of problems. We have demonstrated its effectiveness on very high-dimensional problems with very little data. On problems where linear models cannot adequately capture relationships, the method would fail. Open research areas include a theoretical underpinning of the approach, characterization of the domains on which it is effective, and extension to nonlinear interactions.

Acknowledgements

This work was supported by NSF grant number IIS-9979860.

Appendix

Algorithm 1 The Variable Selection Algorithm.

```
arguments:  Sample data X ∈ ℝ^{M×N}, y ∈ ℝ^M
return:  Variable subset S
function  VS-SSVM(X, y)
  Add in L random variables RVs as new descriptors, X ← (X RVs) ∈ ℝ^{M×(N+L)}
  t ← 0
  repeat
      t ← t + 1
      Randomly partition data into training (X_tr, y_tr) and validation(X_v, y_v) sets
      Perform model selection over parameters C and ν using pattern search
      Solve LP(3) on (X_tr, y_tr), obtain a linear model with the weight vector w^(t)
  until t ≥ T, the maximum number of iterations
  Combine weight vectors to obtain w ← Combine(w^(t), t = 1, 2, ..., T),
  set the threshold γ ← average{w_{N+l}, l = 1, ··· , L}, the weights for RVs.
  return Variable subset S consisting of variables with w greater
        than threshold γ
end
```

Algorithm 2 The Induction Algorithm.

```
arguments:  Sample X ∈ ℝ^{M×N}, y ∈ ℝ^M
return:  SVM regression model f.
function  SSVM(X, y)
  t ← 0
  repeat
      t ← t + 1
      Randomly partition data into training (X_tr, y_tr) and validation (X_v, y_v) sets
      Perform model selection over parameters C, ν, and σ^2
      Solve LP(2) over (X_tr, y_tr) with best C, ν, and σ^2,
         obtain the nonlinear model f^(t)
  until t ≥ T, maximum number of bootstraps
  Bag models f^(t), f ← (1/T) ∑_{t=1}^{T} f^(t)
  return SVM regression model f
end
```

References

K. P. Bennett. Combining support vector and mathematical programming methods for classification. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Machines*, pages 307–326, Cambridge, MA, 1999. MIT Press.

K. P. Bennett and E. J. Bredensteiner. Geometry in learning. In C. Gorini, E. Hart, W. Meyer, and T. Phillips, editors, *Geometry at Work*, Washington, D.C., 1997. Mathematical Association of America. www.rpi.edu/∼bennek/geometry2.ps.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.

P. Bradley, O. Mangasarian, and J. Rosen. Parsimonious least norm approximation. *Computational Optimization and Applications*, 11(1):5–21, 1998.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

C. Breneman, K. Bennett, M. Embrechts, S. Cramer, M. Song, and J. Bi. Descriptor generation, selection and model building in quantitative structure-property analysis. In *Experimental Design for Combinatorial and High Throughput Materials Development*, pages 203–238, 2002.

C. Campbell and K. P. Bennett. A linear programming approach to novelty detection. In *Neural Information Processing Systems*, volume 13, pages 395–401, 2000.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, May 1995.

J. Dennis and V. Torczon. Derivative-free pattern search methods for multidisciplinary design problems. In *The Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 922–932, American Institute of Aeronautics and Astronautics, Reston, VA, 1994.

M. J. Embrechts, F. A. Arciniegas, M. Ozdemir, C. M. Breneman, and K. P. Bennett. Bagging neural network sensitivity analysis for feature reduction in QSAR problems. In *Proceedings of 2001 INNS - IEEE International Joint Conference on Neural Networks*, volume 4, pages 2478–2482, Washington D. C., 2001. IEEE Press.

U. Fayyad, G. Grinstein, and A. Wierse. *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann, 2001.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environ. Economics and Management*, 5:81–102, 1978.

ILOG. *ILOG CPLEX 6.5 Reference Manual*. ILOG CPLEX Division, Incline Village, NV, 1999.

I. Inza, M. Merino, P. Larranaga, J. Quiroga, B. Sierra, and M. Girala. Feature subset selection by population-based incremental learning. Technical Report no. EHU-KZAA-IK-1/99, University of the Basque Country, Spain, 1999.

J. Kittler. Feature selection and extraction. In T. Y. Young and K.-S. Fu, editors, *Handbook of Pattern Recognition and Image Processing*. Academic Press, New York, 1986.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 1997(1 - 2): 273 – 323, 1997.

M. Kudo, P. Somol, P. Pudil, M. Shimbo, and J. Sklansky. Comparison of classifier-specific feature selection algorithms. In *SSPR/SPR*, pages 677–686, 2000.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1983.

A. J. Miller. Subset selection in regression. In *Monographs on Statistics and Applied Probability 40*. London: Chapman and Hall, 1990.

M. Momma and K. P. Bennett. A pattern search method for model selection of support vector regression. In *Proceedings of the SIAM International Conference on Data Mining*, Philadelphia, Pennsylvania, 2002. SIAM.

B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207 – 1245, 2000.

A. J. Smola. *Learning with Kernels*. Ph.D. thesis, Technische Universität Berlin, 1998.

A.J. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proceedings ICANN'99, Int. Conf. on Artificial Neural Networks*, Berlin, 1999. Springer.

M. Song, C. Breneman, J. Bi, N. Sukumar, K. Bennett, S. Cramer, and N. Tugcu. Prediction of protein retention times in anion-exchange chromatography systems using support vector machines. *Journal of Chemical Information and Computer Science*, 42(6):1347–1357, 2002.

H. Stoppiglia and G. Dreyfus. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research, Special Issue on Variable/Feature Selection*, 2003. In this issue.

R. Tibshirani. Regression selection and shrinkage via the lasso. *Journal of the Royal Statistical Society Series B*, 58(1):267–288, 1996.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-Plus*. Springer, New York, 1994.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Neural Information Processing Systems*, volume 13, pages 668–674, 2000.

J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In J. Koza et al., editor, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, page 380, Stanford University, CA, USA, 1997. Morgan Kaufmann.