

# A Pattern Search Method for Model Selection of Support Vector Regression\*

*Michinari Momma<sup>†</sup>, Kristin P. Bennett<sup>‡</sup>*

## Abstract

We develop a fully-automated pattern search methodology for model selection of support vector machines (SVMs) for regression and classification. Pattern search (PS) is a derivative-free optimization method suitable for low-dimensional optimization problems for which it is difficult or impossible to calculate derivatives. This methodology was motivated by an application in drug design in which regression models are constructed based on a few high-dimensional exemplars. Automatic model selection in such underdetermined problems is essential to avoid overfitting and overestimates of generalization capability caused by selecting parameters based on testing results.

We focus on SVM model selection for regression based on leave-one-out (LOO) and cross-validated estimates of mean squared error, but the search strategy is applicable to any model criterion. Because the resulting error surface produces an extremely noisy map of the model quality with many local minima, the resulting generalization capacity of any single local optimal model illustrates high variance. Thus several locally optimal SVM models are generated and then bagged or averaged to produce the final SVM. This strategy of pattern search combined with model averaging has proven to be very effective on benchmark tests and in high-variance drug design domains with high potential of overfitting.

---

\*This work is supported by the NSF Grants IIS-9979860 and IRI-97092306.

<sup>†</sup>Dept. of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180, [mommam@rpi.edu](mailto:mommam@rpi.edu)

<sup>‡</sup>Dept. of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, [bennetk@rpi.edu](mailto:bennetk@rpi.edu)

# 1 Introduction

Support Vector Machines (SVMs) have been proven to be very effective methods for inference. Robust efficient algorithms exist for optimizing SVM models. In this work we utilize the freeware SVMTorch [3]. To obtain good generalization, it is necessary to choose a sufficiently good model parameter set for the particular learning problem. The choice of model parameters can dramatically affect the quality of the solution. Poor choice of parameters can result in failure of the method. “Cheating”, done by selecting parameters based on testing results also used to estimate final model quality, can result in overestimates of the final model quality. This is a particular problem for smaller datasets where insufficient data is available. Thus model selection represents a stumbling block for SVM users. One would like a fully automatic method for selecting SVM parameters. While this may seem like a mundane goal, it is none the less essential.

Our work is motivated by an application in drug design. We would like to predict the bioactivity of a molecule. Since very few molecules have been assayed and there are a great many possible descriptors for each molecule, the data consists of relatively few high-dimensional data points. While SVM can work quite well in this domain, model selection is a real-issue. To estimate model accuracy, one must perform some sort of cross-validation. One must perform model selection for each out-of-sample trial, because choosing a fixed parameter policy based on the final cross-validated model will result in overestimating the model quality. Use of tuning sets to select model parameters is inherently unreliable, since they are of necessity very small. Hand tuning parameters is undesirable since model selection should be done for each validation trial. The availability of large sets of data helps but does not eliminate the model selection problem.

The topic of how to select model parameters for SVM is often neglected. In the area of classification, there are some notable exceptions; There has been some work on efficient methods for estimating model quality such as leave-one-out (LOO), the radius-margin bound and span bound [1, 2] in SVM classification. Chapelle et al. [2] propose using a gradient descent search to optimize the radius-margin and span-bound criteria. An efficient implementation for the radius-margin bound for large problems is reported in [8]. However, typically the criteria, such as LOO error, are not differentiable, so the gradient oriented methods are not generally applicable in using cross-validation type of criterion.

In other SVM research, just the selection criterion is addressed not the search method. To evaluate the efficiency of model selection method, both the criterion and search method should be presented. Moreover, the little results that are available focus on classification. No practical tools for model selection of SVM regression have been developed yet.

We focus on selecting the three model parameters for classical support vector machine regression (SVR) with the radial basis function kernel (RBF) [15], but this work is completely generalizable to other models for classification and regression. We utilize the LOO or cross-validated mean squared error for regression models. When the squared error is normalized by dividing by the sample variance, the resulting statistic is referred to as  $Q^2$ . Since the method is not dependent on

derivatives, any metric and method for estimating model quality can be used. Our ultimate goal is to base the methodology on less expensive estimates of model quality.

To optimize the model criteria we utilize a technique called pattern search (PS) [5]. PS is a direct search method. PS is selected because it is a simple effective optimization technique suitable for optimizing a wide range of objective functions. It does not require derivatives, only direct function evaluation – making the method suitable for problems for which it is impossible or hard to obtain information about the derivatives. To exclude inappropriate parameters, a constrained version of the pattern search can be used [11]. The convergence to local minima for constrained problems as well as unconstrained problems has been proven in [11].

In practice PS is robust, and rarely gets trapped in bad local minima. Note that we wish to optimize a very noisy and somewhat expensive estimate of model quality. So we want a simple, fairly inexpensive algorithm for finding good but not necessarily the best solutions. Finding the globally optimal model would probably result in overfitting. Other derivative free search methods are possible. Simulating annealing represents an attractive alternative but we leave this to future research [4]. Genetic Algorithms must be applied cautiously since they require a large number of function evaluations per iteration [4].

Finding an effective strategy to minimize the estimate of model quality is not always sufficient to produce good models. Using PS along with LOO  $Q^2$  is not sufficient in domains such as drug design. Two problems exist. The models that optimize the estimated model quality do not necessarily produce the best generalization; and the search method (PS) may be trapped into a bad local minimum, which leads to a high variance in the selected model. We adopt a bagging strategy to resolve these issues. To automatically produce robust SVM models, we restart PS to produce several locally optimal models, then average these models to produce the final SVM.

As we will see later in Section 4, the error surface typically looks like a valley in the parameter space. All we need to do in this situation is to pick up several local minima from the sink as effective as possible, since calculations of LOO or cross validation are in general expensive. In this sense, a local optimization method such as PS is a better choice than other global optimization that requires more number of function evaluations and iterations to converge.

As the low-cost local optimization method, the PS enables us to utilize bootstrapping sampling and bagging to obtain the final prediction. When the good parameters are known, PS can be used to fine tune the model to boost the predictive capability of SVMs with bootstrapping and bagging. An illustrative example in the case of Boston Housing is shown in this paper.

This paper is organized as follows; In Section 2, the pattern search algorithm is explained. In Section 3, we describe our PS methodology for SVM regression. In Section 4, we successfully apply the methodology in two drug design domains and on the Boston Housing problem.

## 2 Pattern Search Algorithm

We describe the generic PS algorithm adapted from [5]. PS samples points in the search space in a fixed pattern about the current point. This algorithm calculates function values of the pattern and tries to find a minimizer. If it finds a new minimum, then it changes the center of pattern and iterates. If all the values on the pattern fail to produce a decrease, then the search step or pattern size is reduced by half. This search continues until the search step  $\Delta_k$  gets sufficiently small, thus ensuring convergence to a local minimum. Efficiency is gained by reusing pattern values as the pattern center moves

The PS is based on a pattern  $P_k$ . There are many ways to generate  $P_k$ . For the purpose of the three-dimensional search of  $q \in \mathbf{R}^3$  in this paper we used the nearest neighbor sampling pattern shown in Figure 1 and given numerically by

$$P_k = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \end{bmatrix}. \quad (1)$$

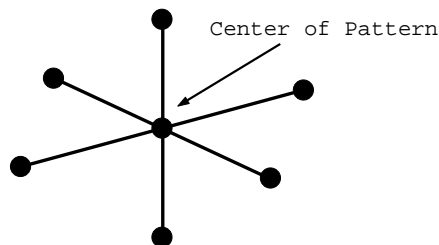


Figure 1. The pattern used in the pattern search

A trial step  $s_k^i$  is defined by  $s_k^i = \Delta_k p_k^i$ , where  $\Delta_k$  is a constant that determines the length of the search step and  $p_k^i$  is the  $i^{\text{th}}$  column in the matrix  $P_k$ . The  $i^{\text{th}}$  trial point in the pattern is given by  $q_k^i = q_k + s_k^i$ , where  $q_k$  is a current best solution and  $q_k^i$  is examined to see if it is a better solution. The PS algorithm is:

### Pattern Search Algorithm

1. Given  $\Delta_k$  and  $\tau$ , randomly pick an initial center of pattern  $q_k$ , then compute the function value  $f(q_k)$ , and set  $min \leftarrow f(q_k)$ .
2. If  $\Delta_k \leq \tau$  then stop.
3. for  $i=1, \dots, p-1$  where  $p$  is number of columns in  $P_k$ .
4.      $q_k^i \leftarrow q_k + s_k^i$
5.     compute  $f(q_k^i)$
6.     if  $f(q_k^i) < min$  then
7.          $q_{k+1} \leftarrow q_k$ ;  $\Delta_{k+1} \leftarrow \Delta_k$ ;  $min \leftarrow f(q_k^i)$ ;  $k \leftarrow k+1$ .
8.  $\Delta_{k+1} \leftarrow \Delta_k/2$ ;  $k \leftarrow k+1$ ; go to 2

The advantages of PS are flexibility, robustness, and simplicity. It does not require differentiability or continuity of the objective function. Furthermore, because the algorithm samples a point even if it is in an increasing direction of the objective function, PS can be more robust to local minima than derivative-based methods. It can be applied to problems where the derivative information is not reliable or even to problems where the function is not differentiable, nor even continuous. We choose a simple but effective variant of PS with few algorithm parameters. The pattern that we use is one of the simplest patterns, which consist of only nearest neighbors. We could have chosen a different pattern such as including next nearest neighbors or more; see [5]. Other direct search or randomized algorithms are certainly possible. We select the PS algorithm because of its proven effectiveness.

### 3 Pattern Search for SVM-Regression

To apply PS to SVM regression, one must define the model and model quality function to be optimized. We used the LOO  $Q^2$  statistic applied to standard SVM regression [15]. Specifically, suppose we are given training examples  $(x_i, y_i)$ , where  $x_i \in \mathbf{R}^n$  and  $y_i \in \mathbf{R}$ . The optimal  $\epsilon$ -insensitive SVM model can be found by solving the following dual quadratic program:

$$\begin{aligned} & \text{minimize } \frac{1}{2}(\alpha^* - \alpha)^T K(\alpha^* - \alpha) - (\alpha^* - \alpha)^T y + \epsilon(\alpha^* + \alpha)^T \mathbf{1} \\ & \text{s.t. } (\alpha - \alpha^*)^T \mathbf{1} = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, l, \end{aligned} \quad (2)$$

where  $K$  is a matrix with  $K_{ij} = k(x_i, x_j)$  and  $\mathbf{1}$  is a vector of ones. The optimal regression function is given by

$$\hat{y} = f(x) = \sum_{i=1}^l (\alpha_i^* - \alpha_i) k(x_i, x) + b. \quad (3)$$

We used the radial-basis function kernel with standard deviation  $\sigma$ :  $k(x_i, x) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right)$ . Once the three model parameters,  $\epsilon$ ,  $C$ , and  $\sigma$ , are fixed, Problem (2) is solved using SVMTool [3]. The approach can be applied to alternative SVM regression formulations such as in [13].

#### SVM Model Selection

To estimate the quality of the model we use the LOO  $Q^2$  statistic which is just the predicted mean squared error divided by the sample variance. Any alternative objective function could be used. For each data point  $x_i$ ,  $i = 1, \dots, n$ , Problem (2) is solved leaving out the  $i^{\text{th}}$  data point and then the predicted value  $\hat{y}_i$  is produced for that point. Then  $Q^2$  is defined as  $Q^2 = \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (\bar{y} - y_i)^2}$ . As in [10, 9] strategies can be developed for computing LOO estimates efficiently. To cut down on the number

of problems solved at each LOO iteration, we solve Problem (2) using all the data. Then for all points  $x_i$  strictly in the tube, i. e. such that the optimal  $\alpha_i = \alpha_i^* = 0$ , the LOO  $\hat{y}_i$  equals  $\hat{y}_i$  computed on the full model since dropping these points would have no effect on the optimal solution.

## 4 Computational Results and Discussion

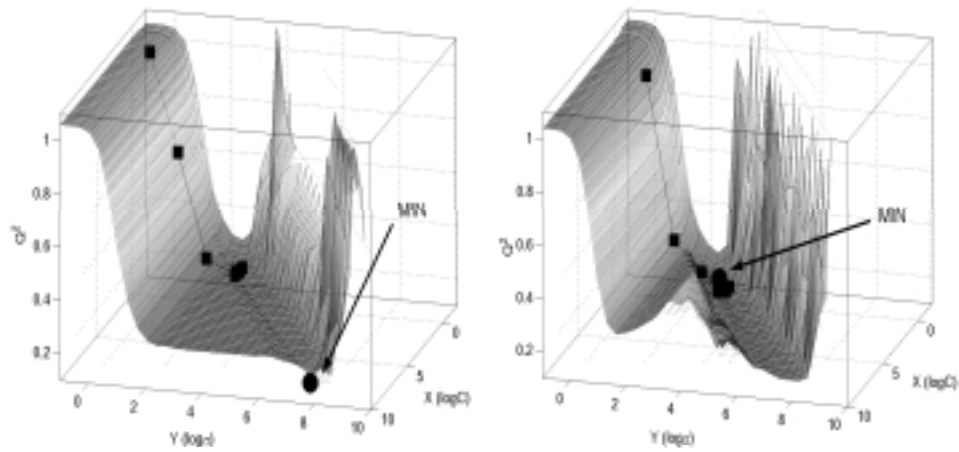
These model selection techniques were explicitly developed for application to drug design problems known as Quantitative Structure-Activity Relationship (QSAR) modeling. The goal of QSAR is to predict biological activity values based on molecular descriptors. Since assessing bioactivity can be very expensive, QSAR datasets typically consist of very few data points (less than 70 in this paper) making it easy to overfit models. We provide sample results from two datasets: the Lombardo blood-brain barrier data set [12] and the HIV reverse transcriptase inhibitors (HIVrt) [7]. The Lombardo data set used consists of sixty-two molecules each with 21 molecular descriptors. The HIVrt data used consists of sixty-four molecules each with 26 molecular descriptors. The descriptor subsets were selected using an SVM feature selection method that is beyond the scope of this paper.

To evaluate the effectiveness of PS, we compared it with a simple grid search (GS). In GS, a uniform grid is defined in the parameter space. Then all the points in the grid are evaluated in order to find a global minimum. The GS will find the global minimum of all the points in the parameter grid. The coarseness of the grid determines the quality of the solution found and the efficiency of the search. Typically such GSs are used either formally or informally for SVM parameter selection.

We first define a finite search region of interest. Usually,  $C$  and  $\sigma$  can take on a large range of values from  $\sim 1$  to  $\sim 1000$  or larger. Therefore, we use a log scale to cover such a large region. For  $\epsilon$ , we use the log scale as well. This transformation is defined as follows:  $X = \log C$ ;  $Y = \log \sigma$ ;  $Z = \log \frac{1}{\epsilon}$ . The  $(X, Y, Z)$  coordinates are used for both PS and GS. For GS, we set the grid size  $\Delta_g = 0.5$  for three-dimensional search and  $\Delta_g = 0.1$  for two-dimensional search. The GS scans the space with the range  $-2 \leq X \leq 10$ ,  $-2 \leq Y \leq 10$  and  $1 \leq Z \leq 6$ , which correspond to  $0.1353 \leq C \leq 22026$ ,  $0.1353 \leq \sigma \leq 22026$ , and  $0.0025 \leq \epsilon \leq 0.3679$ . For the pattern search, we set  $\Delta_1 = 1$  initially and continue the search until  $\Delta_k < 0.04$ .

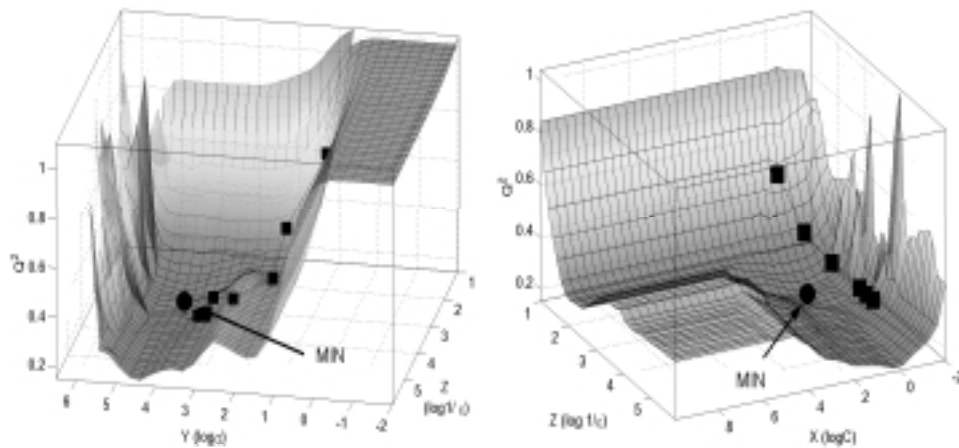
Using the Lombardo dataset, we run GS in two dimensions to see what the error ( $Q^2$ ) surface looks like and examine the performance of PS. The grid size  $\Delta_g$  is set to be 0.1 for all coordinates  $X$ ,  $Y$ , and  $Z$ . In Figure 2 *left*, we show the error surface with  $Z = 2$  ( $\epsilon = 0.1353$ ) fixed. The circle shows the minimum obtained by the GS. We also plot the result of 2D PS. The series of squares shows the path of the PS. In Figure 2 *right*, we show the error surface with  $Z = 4$  ( $\epsilon = 0.0183$ ). The circle and squares are defined as above. The difference, however, is that as  $Z$  increases, or equivalently  $\epsilon$  decreases, the error surface gets more complicated. In Figure 2 *left*, the error surface is smooth and roughly convex. However, in *right*, we can observe there is a large ridge in the search space resulting in multiple local minima. But in both cases PS found a solution near the global minimum.

This behavior can be understood better if we take a look at Figure 3. Figure 3



**Figure 2.** The error surface with left:  $Z = 2$  ( $\epsilon = 0.1353$ ) and right:  $Z = 4$  ( $\epsilon = 0.0813$ ) fixed. Circle: minimizer obtained from GS. Squares: path of PS.

left shows the error surface with  $X = 3$  ( $C = 20.0855$ ) fixed and right with  $Y = 3$  ( $\sigma = 20.085$ ) fixed. In both figures, as the value of  $Z$  increases, or equivalently as  $\epsilon$  decreases, the error surface becomes more complicated.

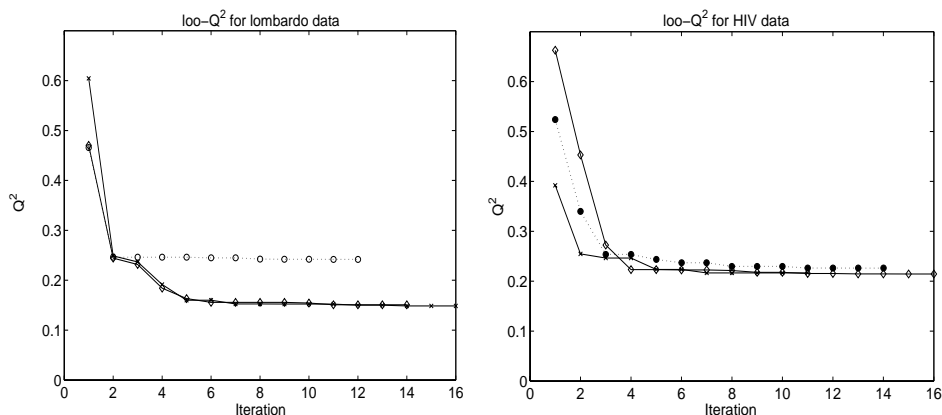


**Figure 3.** The error surface with left:  $X = 3$  ( $C = 20.0855$ ) and right:  $Y = 3$  ( $\sigma = 20.0855$ ) fixed. Circle: the minimizer obtained from GS. Squares: path of PS.

The shape of the error surfaces suggests that there exists a region that has

small LOO  $Q^2$ . In other words, it suffices to pick up a point in this region for our purpose of finding a good local minimum. This task is not so difficult; not too many steps for PS are needed to find a local minimum in the region. This becomes clear in the three-dimensional case.

Next, we show results of GS and PS in the three-dimensional space. In GS, we set  $\Delta_g = 0.5$  and  $1.0$ . Since in three dimensions it is very time consuming to reduce the  $\Delta_g$  more, we do not go further. The results are LOO  $Q^2 = 0.213792$  for  $\Delta_g = 0.5$  and  $0.217684$  for  $\Delta_g = 1$ . In Figure 4 *left*, we show the LOO  $Q^2$  with three different initial centers of the pattern. The minimum value is  $0.1486$ . Two of the three lines behave very similarly. However, one of them is trapped in a local minimum. We note that the number of iterations is different among experiments, thus the ends of the lines in Figure 4 are not the same.



**Figure 4.** LOO  $Q^2$  for the *left*: Lombardo and *right*: HIV datasets. There are three different lines corresponding to different initial starting points.

For the HIV data, the LOO  $Q^2$  obtained by GS are as follows:  $0.266734$  for  $\Delta_g = 1$  and  $0.233563$  for  $\Delta_g = 0.5$ . The results using PS are shown in Figure 4 *right*. The average minimum value is  $0.2145$ . In this case again, one of the initial points leads to a local minimum that is slightly larger than the best solution.

In both data sets, PS takes less than 20 iterations to converge. As seen in Figure 4, it is possible to stop the search when the objective stops changing. Considering the structure of the pattern, the number of LOO function evaluations is bounded by  $7 \times k$ , where  $k$  is the number of iterations to get a local minimum. In the above results,  $k \leq 20$  so that total number of function evaluations is bounded by 140. In GS, with the grid size  $\Delta_g = 1.0$ , it takes 1183 function evaluations and for  $\Delta_g = 0.5$ , 8125 function evaluations. Obviously PS is much more efficient compared with the 3D GS for the same degree of accuracy.

## Generalization

We now examine the generalization ability of SVM with parameters obtained by PS. We use 10-fold cross validation to estimate the generalization error. We compare results using PS with those using grid search. In the first experiment, the two pharmaceutical datasets are used for comparison. The results are given in Table 1. Since the computational cost is high for the 3D grid search, the results reported are calculated by using only one experiment of the 10-fold cross validation. The grid search with  $\Delta_g = 0.5$  gives  $Q^2 = 0.2843$  and mean squared error (MSE) = 0.0165 for the HIV dataset, and  $Q^2 = 0.1669$  and MSE = 0.0103 for the Lombardo dataset. For PS,  $Q^2 = 0.2697$  and MSE = 0.01566 for the HIV dataset, and  $Q^2 = 0.17908$  and MSE = 0.01102 for the Lombardo dataset. We note that the results of PS are averaged over 9 runs with different initial starting points. The total number of function evaluations (LOO calculations), in one trial of 10-fold cross validation, is about 80000 for the grid search and 420 to 430 for each trial of PS.

To increase the prediction accuracy and robustness, it is better to use bagging for several PS runs. In this case, we bag or average the models produced by the 9 trials of PS and the resulting values are:  $Q^2 = 0.2526$  and MSE = 0.01467 for HIV, and  $Q^2 = 0.1753$  and MSE = 0.0108 for Lombardo. By averaging or bagging the models, we can exploit the fact that PS may find several different local minima and it will reduce the chance of overfitting the training examples. This comparison between grid search and PS shows PS is much more effective than grid search in terms of generalization and computational cost.

	HIV			Lombardo		
	GS	PS	Bag	GS	PS	Bag
$Q^2$	0.2843	0.2697	0.2526	0.1669	0.1791	0.1753
MSE	0.0165	0.01566	0.01467	0.0103	0.01102	0.0108
N.F.E.	81250	424	3816	81250	421	3789

**Table 1.** Results of the HIV and Lombardo datasets of 10-fold CV. N.F.E.: number of function evaluations. Numbers for PS is averaged over 9 trials from different starting point. Bagging makes use of all the 9 trials to obtain one prediction for each fold of cross validation.

In order to obtain more reliable statistics, in the second experiment, we analyze PS further by restarting the experiments 10 times with different random splits of the 10-fold cross validation. The results are shown in Table 2. To compare the results with those obtained by a fixed policy, we adopt the values reported for the Boston Housing data in [13]:  $C = 10 \times (y_{max} - y_{min})$ , where  $y_{max}$  is the maximum of the response variable and  $y_{min}$  the minimum of the response variable, and  $\sigma = \sqrt{0.15N}$  where  $N$  is the number of attributes. For  $\epsilon$ , we pick  $2/(y_{max} - y_{min})$  since it gives the best result in [13]. The fixed policy performs very well on the Boston Housing data since it was tuned based on testing set information. But the fixed policy performs poorly on the two QSAR datasets as shown in the Table 2. PS

	HIV			Lombardo		
	Fixed	PS	Bag	Fixed	PS	Bag
$Q^2$	0.425	0.300	0.278	0.465	0.180	0.170
STD	0.071	0.032	0.021	0.101	0.029	0.010
MSE	0.025	0.017	0.016	0.029	0.020	0.011
STD	0.004	0.002	0.001	0.006	0.029	6E-4
ITE	—	11.74	11.93	—	12.07	12.4
F.E.	1	42.4	42.8	1	41.9	43.3

**Table 2.** Results of HIV and Lombardo datasets average of 10 runs of 10-fold CV. STD: standard deviation. ITE: average number of iterations per one run of PS, and F.E.: average number of function evaluations per one run of PS.

gives better results than the fixed policy and bagging using different starting points further improves the results. Most significantly bagging reduces the variance in the error estimate.

	Fixed		PS		Bag	
	Training	Test	Training	Test	Training	Test
$Q^2$	0.07087	0.1245	0.06155	0.1440	0.04697	0.1081
STD	0.003127	0.1028	0.0239	0.1129	0.006184	0.08326
MSE	5.978	8.824	5.189	10.56	3.960	7.887
STD	0.2549	6.779	1.990	8.410	0.5007	6.059
ITE	—		8.94		181.3	
F.E.	1		50.4		1020	

**Table 3.** Results of Boston Housing dataset. STD: standard deviation. ITE: average number of iterations per final prediction, and F.E.: average number of function evaluations per final prediction. PS starts from the fixed policy. We take 20 bootstrapping samples to obtain one final prediction.

### Fine Tuning from the Fixed Policy

We now show how PS can be combined with the fixed policy and bagging to further improve results. The same experimental format was applied to the Boston Housing data set as in [6], [13] and [14]. Out of the total of 506 examples, the number of training examples is set to be 401, validation examples to 80 and test examples to 25. For the fixed policy, we randomly divide the whole examples into training (481) and test (25) examples, then produce a SVM prediction using the training examples and evaluate the predictive ability with the test examples. This process is repeated 100 times. The results are shown in Table 3. The values are averaged over the 100 trials. We obtain a similar value (MSE = 8.8) to [13] (MSE = 8.7).

For comparison, the same 100 divisions of the examples are repeatedly used in the subsequent experiments.

Our first approach was to apply PS using the fixed policy for the starting point. Only one validation set was used as in [6] and [14]. The fixed policy provides a good start for PS in terms of optimizing error criteria in the validation set. In this case, however, there is an issue of overfitting. Since we utilize one validation set instead of LOO or cross-validated error measurements for PS, we need to avoid overfitting the training data. We stop the search at the 10<sup>th</sup> iteration, which is a little bit earlier than the average convergence in Table 2. The idea behind this is that we would like to avoid overfitting too much of the training and validation examples. The experiments are done with the same 100 divisions as used for the fixed policy. However, as seen in Table 3, results for PS show overtraining of the training examples; test results get worse while training results get better. This is due to the fact that the fixed policy is tuned by looking at the validation set result, which causes PS to overtrain easily, and we might as well stop even earlier. It would be possible to use some heuristic as a stopping criteria, however, there is no theoretical justification that this would obtain better results. Therefore, we bootstrap the training examples to create multiple predictions, and then bag them to get the final prediction.

The bagging result is fascinating. The same number of validation and training examples are used for bootstrapping, that is, out of 401 training examples, 80 validation examples are randomly chosen and used for model validating in the PS algorithm. This operation is repeated 20 times to obtain 20 different predictions. The final result is simply the average value over these 20 predictions then by using test examples, the final bagged model is evaluated. This whole process is repeated 100 times by using the same division of the examples as in the fixed policy. This method reduces the MSE from 8.824 by fixed policy and 10.56 by single PS, to 7.887. The standard deviation is also reduced from 6.779 for the fixed policy and 8.41 for PS, to 6.059. Although the MSE is still larger than the value 7.2 reported in [6] and 7.6 in [14], the reduction from 8.824 to 7.887 is significant. Of course, as in many machine learning textbooks, it is possible to improve results by bagging. However, without an automatic way to tune parameters, it is impractical to do many bootstrapping samplings to produce a final bagged SVM model for each of the 100 trials. A coarse GS is often used for these sorts of calculations; PS provides a more efficient and fine way of tuning parameters.

## 5 Conclusion

In this paper, we developed a pattern search methodology for SVM model selection. Robust self-tuning SVMs are essential in underdetermined problem domains, such as drug design, where over-fitting is a real threat. For two drug design problems and the Boston Housing data set, we found PS was as effective as and more efficient than an exhaustive grid search. We proposed a bagging methodology to handle local minima. This bagging approach is robust to the danger that the global minimum of the model quality estimate may have suboptimal generalization due to small

training set sizes. When applied to Boston Housing, we saw that PS with bagging could be used to enhance predictive ability. The fixed policy gained more power when being incorporated by PS with bagging.

This approach eliminates the parameter selection stumbling block experienced by users. Using this autotuning methodology, the only SVM parameter choices that a user must make are the type of kernel and granularity of search. Individual components of the strategy can be replaced. The methodology can be readily adapted to other model selection problems with a small number of parameters (3 or 4 are ideal) and any model selection criteria that can be computed with reasonable efficiency. Other optimization algorithms such as simulated annealing could also be used. We selected PS strategy because of its simplicity, ease of implementation, and robustness. But the results for grid search indicate that there is no point in performing an extensive search of the parameter space.

When combined with bagging, any strategy that generates a good solution would be effective. The PS method would not be applicable to high dimensional parameter space due to excessive computational costs. For these situations, other optimization method may be preferable. The simplicity of the overall approach should not be used to underestimate its importance. Practical application of SVM and fair comparison of different learning methodologies dictates the development of automated model tuning strategies. This critical issue is too often brushed aside in SVM papers. In fact, no other practical methodology for model selection of SVM regression is available now.

# Bibliography

- [1] O. CHAPELLE and V. VAPNIK, *Model Selection for support vector machines*. In Advances in Neural Information Processing Systems 12, S. Solla, T. Leen, and K. Muller, eds., MIT PRESS, (1999).
- [2] O. CHAPELLE, V. VAPNIK, O. BOUSQUET, and S. MUKHERJEE, *Choosing kernel parameters for support vector machines*, Machine Learning, (2001) to appear.
- [3] R. COLLOBERT and S. BENGIO, *Support vector machines for large-scale regression problems*, IDIAP-RR-00-17, (2000).
- [4] L. DAVIS, *Genetic Algorithms and Simulated Annealing*. Pitman, London, 1987.
- [5] J. DENNIS and V. TORCZON, *Derivative-free pattern search methods for multidisciplinary design problems*, paper AIAA-94-4349 in Proceedings of the 5th AIAA/ USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL, Sept. 7-9, (1994), pp. 922-932.
- [6] H. DRUCKER, C. BURGESS, L. KAUFMAN, A. SMOLA, and V. VAPNIK, *Support vector regression machines*. In Advances in Neural Information Processing Systems 9, M. Mozer, M. Jordan, and T. Petsche, eds, (1996), pp. 155-162.
- [7] R. GARG, S.P. GUPTA, H. GAO, M.S. BABU, A.K. DEBNATH and C. HANSCH, *Comparative QSAR studies on anti-HIV drugs*, Chem. Rev., 99, (1999), pp. 3525-3601.
- [8] S. KEERTHI. *Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms*, IEEE Transactions on Neural Networks, (2001), to appear.
- [9] S. KEERTHI, C. ONG, and M. Lee, *Two efficient methods for computing leave-one-out error in SVM algorithms*, Technical Report CD-00-10, Dept of Mechanical Engineering, National University of Singapore, (2000).
- [10] J. LEE, and C. LIN. *Automatic model selection for support vector machines*, <http://www.csie.ntu.edu.tw/~cjlin/papers/modelselect.ps.gz>, (2000).
- [11] R. LEWIS and V. TORCZON, *Pattern search algorithms for bound constrained minimization*, SIAM J. Optimization, 9, (1996), pp. 1082-1099.

- [12] F. LOMBARDO, J. BLAKE, and W. CURATOLO, *Computation of brain-blood partitioning of organic solutes via free energy calculations*, J. Med. Chem., 30, (1996), pp. 4750-4755.
- [13] B. SCHÖLKOPF, P. BARTLETT, A. SMOLA, and R. WILLIAMSON, *Shrinking the tube: a new support vector regression algorithm*. In Advances in Neural Information Processing Systems 11, M. Kearns, S. Solla, and D. Cohn, Eds., (2000), pp. 330-336.
- [14] M. STITSON, A. GAMMERMAN, V. VAPNIK, V. VOVK, C. WATKINS and J. WESTON, *Support vector regression with ANOVA decomposition kernels*. In Advances in Kernel Methods — Support Vector Learning, B. Schölkopf, C. Burges, A. Smola, eds., MIT Press, Cambridge, MA, (1999), pp. 285-291.
- [15] V. VAPNIK, *The nature of Statistical Learning Theory*, New York, Wiley, (1996).