Support Vector Regression Machines

Harris Drucker* **Chris J.C. Burges**** **Linda Kaufman****
Alex Smola** **Vladimir Vapnik**+

*Bell Labs and Monmouth University
Department of Electronic Engineering
West Long Branch, NJ 07764
**Bell Labs +AT&T Labs

Abstract

A new regression technique based on Vapnik's concept of support vectors is introduced. We compare support vector regression (SVR) with a committee regression technique (bagging) based on regression trees and ridge regression done in feature space. On the basis of these experiments, it is expected that SVR will have advantages in high dimensionality space because SVR optimization does not depend on the dimension&y of the input space.

This is a longer version of the paper to appear in *Advances in Neural Information Processing Systems 9* (proceedings of the 1996 conference)

1. Introduction

In the following, lower case bold characters represent vectors and upper case bold characters represent matrices. Superscript "t" represents the transpose of a vector. y represents either a vector (in bold) or a single observance of the dependent variable in the presence of noise. $y^{(p)}$ indicates a predicted value due to the input vector $\boldsymbol{x}^{(p)}$ not seen in the training set.

Suppose we have an unknown function $G(\boldsymbol{x})$ (the "truth") which is a function of a vector $\boldsymbol{x}$ (termed *input space). The vector* $\boldsymbol{x}^t = [x_1, x_2, \ldots, x_d]$ has d components where d is termed the dimensionality of the input space. $F(\boldsymbol{x}, \boldsymbol{w})$ is a family of functions parameterized by w. $\hat{\boldsymbol{w}}$ is that value of $\boldsymbol{w}$ that minimizes a measure of error between $G(\boldsymbol{x})$ and $F(\boldsymbol{x}, \hat{\boldsymbol{w}})$.

Our objective is to estimate $\boldsymbol{w}$ with $\hat{\boldsymbol{w}}$ by observing the N training instances $\boldsymbol{v}_j$, $j=1, \cdots, N$. We will develop two approximations for the truth $G(\boldsymbol{x})$. The first one is $F_1(\boldsymbol{x}, \hat{\boldsymbol{w}})$ which we term a feature space representation. One (of many) such feature vectors is:

"Support Vector Regression Machines", *Neural Information Processing Systems 9*, with C.J. Burges, L. Kauffman, A. Smola, and V. Vapnik, eds. M.C. Mozer, J.I. Jordan, T. Petsche, pp. 155-161, MIT Press. 1997

$$z^t=[x_1^2,\cdots,x_d^2,x_1x_2,\cdots,x_ix_j,\cdots,x_{d-1}x_d,x_1,\cdots,x_d,1]$$

which is a quadratic function of the input space components. Using the feature space representation, then $F_1(x,\hat{w})=z^t\hat{w}$, that is, $F_1(x,\hat{w})$ is linear in *feature* space although it is quadratic in input space. In general, for a p'th order polynomial and d'th dimensional input space, the feature dimensionalityfof $\hat{w}$ is

$$f = \sum_{i=d-1}^{p+d-1} C_{d-1}^{i}$$

where $C_k^n = \frac{n!}{k!(n-k)!}$. From now on, when we write $F_1(x,\hat{w})$, *we mean* the feature space representation and we must determine the f components of $\hat{w}$ from the N training vectors.

The second representation is a support vector regression (SVR) representation that was developed by Vladimir Vapnik (1995):

$$F_2(x,\hat{w})=\sum_{i=1}^{N}(\alpha_i^*-\alpha_i)(v_i^tx+1)^p + b$$

F_2 is an expansion explicitly using the training examples. The rationale for calling it a support vector representation will be clear later as will the necessity for having both an a* and an a rather than just one multiplicative constant. In this case we must choose the 2N + 1 values of a; α_i^* and b. If we expand the term raised to the p'th power, we find f coefficients that multiply the various powers and cross product terms of the components of x. So, in this sense F_1 looks very similar to F_2 in that they have the same number of terms. However F_1 has f free coefficients while F_2 has 2N+1 coefficients that must be determined from the N training vectors. For instance, suppose we have a 100 dimensional input vector and use a third order polynomial. The dimensionality of feature space for both F_1 and F_2 exceeds 176,000, but for the feature space representation, we have to determine over 176,000 coefficients while for the SVR representation we have to determine 2N+1 coefficients. Thus, it may take a lot more data to estimate the coefficients in $\hat{w}$ in the feature space representation.

We let α represent the 2N values of α_i and α_i^*. The optimum values for the components of $\hat{w}$ or α depend on our definition of the loss function and the objective function. Here the primal objective function is:

$$U\sum_{j=1}^{N}L[y_j-F(v_j,\hat{w})]+||\hat{w}||^2$$

where L is a general loss function (to be defined later) and F could be F_1 or F_2, y_j is the observation of $G(x)$ in the presence of noise, and the last term is a regularizer. The regularization constant is U which in typical developments multiplies the regularizer but is placed in front of the first term for reasons discussed later.

If the loss function is quadratic, i.e., we $L[\cdot]=[\cdot]^2$, and we let $F=F_1$, i.e., the feature space representation, the objective function may be minimized by using linear algebra techniques since the feature space representation is linear in that space. This is termed ridge regression (Miller, 1990). In particular let V be a matrix whose i'th row is the i'th training vector represented in feature space (including the constant term "1" which represents a bias). V is a matrix where the number of rows is the number of examples (N) and the number of columns is the dimensionality of feature space f. Let E be the fxf

diagonal matrix whose elements are 1/U. y is the Nx1 column vector of observations of the dependent variable. We then solve the following matrix formulation for $\hat{w}$ using a linear technique (Strang, 1986) with a linear algebra package (e.g., MATLAB):

$$V^t y = [V^t V + \mathrm{E}]\ \hat{w}$$

The rationale for the regularization term is to trade off mean square error (the first term) in the objective function against the sire of the $\hat{w}$ vector. If U is large, then essentially we are minimizing the mean square error on the training set which may give poor generalization to a test set. We find a good value of U by varying U to find the best performance on a validation set and then applying that U to the test set. U is very useful if the dimensionality of the feature set is larger than the number of examples.

Let us now define a different type of loss function termed an e-insensitive loss (Vapnik, 1995):

$$L = \begin{cases} 0 & \text{if } |\, y_i - F_2(x_i, \hat{w})\,| < \varepsilon \\ |\, y_i - F_2(x_i, \hat{w})\,| - \varepsilon & \textit{otherwise} \end{cases}$$

This defines an ε tube (Figure 1) so that if the predicted value is within the tube the loss is zero, while if the predicted point is outside the tube, the loss is the magnitude of the difference between the predicted value and the radius ε of the tube.

Specifically, we minimize:

$$U(\sum_{i=1}^{N} \xi_i^{*} + \sum_{i=1}^{N} \xi_i) + \frac{1}{2}(w^t w)$$

where ξ_i or ξ_i^{*} is zero if the sample point is inside the tube. If the observed point is "above" the tube, ξ_i is the positive difference between the observed value and ε and α_i will be nonzero. Similary, ξ_i^{*} will be nonzero if the observed point is below the tube and in this case α_i^{*} will be nonzero. Since an observed point can not be simultaneously on both sides of the tube, either a; or α_i^{*} will be nonzero, unless the point is within the tube, in which case, both constants will be zero.

If U is large, more emphasis is placed on the error while if U is small, more emphasis is placed on the norm of the weights leading to (hopefully) a better generalization. The constraints are: (for all i, i=1,N)

$$\begin{aligned} y_i - (w^t v_i) - b &\le \varepsilon + \xi_i \\ (w^t v_i) + b - y_i &\le \varepsilon + \xi_i^{*} \\ \xi_i^{*} &\ge 0 \\ \xi_i &\ge 0 \end{aligned}$$

The corresponding Lagrangian is:

$$L = \frac{1}{2}(w^t w) + U(\sum_{i=1}^{N} \xi^{*}{}_i + \sum_{i=1}^{N} \xi_i) - \sum_{i=1}^{N} \alpha_i^{*}[y_i - (w^t v_i) - b + \varepsilon + \xi_i^{*}]$$
$$- \sum_{i=1}^{N} \alpha_i[(w^t v_i) + b - y_i + \varepsilon + \xi_i] - \sum_{i=1}^{N} (\gamma_i^{*} \xi_i^{*} + \gamma_i \xi_i)$$

where the γ_i and α_i are Lagrange multipliers

We find a saddle point of L (Vapnik, 1995) by differentiating with respect to w_i, b, and ξ

which results in the equivalent maximization of the (dual space) objective function:

$$W(\alpha,\alpha^*) = -\varepsilon\sum_{i=1}^{N}(\alpha_i^*+\alpha_i)+\sum_{i=1}^{N}y_i(\alpha_i^*-\alpha_i) - \frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i^*-\alpha_i)(\alpha_j^*-\alpha_j)(\boldsymbol{v}_i^t\boldsymbol{v}_j+1)^p$$

with the constraints:

$$0\le\alpha_i\le U \qquad 0\le\alpha_i^*\le U \qquad i=1,\ldots,N$$
$$\sum_{i=1}^{N}\alpha_i^* = \sum_{i=1}^{N}\alpha_i$$

We must find N Largrange multiplier pairs (a,, α_i^*). We can also prove that the product of α_i and α_i^* is zero which means that at least one of these two terms is zero. A $\boldsymbol{v}_i$ corresponding to a non-zero α_i or α_i^* is termed a support vector. There can be at most N support vectors. Suppose now, we have a new vector $\boldsymbol{x}^{(p)}$, then the corresponding prediction of $y^{(p)}$ is:

$$y^{(p)} = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)(\boldsymbol{v}_i^t\boldsymbol{x}^{(p)} + 1)^p + b$$

Maximizing W is a quadratic programming problem but the above expression for W is not in standard form for use in quadratic programming packages (which usually does minimization). If we let

$$\beta_i = \alpha_i^* \qquad \beta_{i+N} = \alpha_i \qquad i=1,\ldots,N$$

then we minimize:

$$f(\beta)=\frac{1}{2}\boldsymbol{\beta}^t\boldsymbol{Q}\boldsymbol{\beta}+\boldsymbol{c}^t\boldsymbol{\beta}$$

subject to the constraints

$$\sum_{i=1}^{N}\beta_i = \sum_{N+1}^{2N}\beta_i \text{ and } \quad 0\le\beta_i\le U \quad i=1, \quad ,2N$$

where

$$\boldsymbol{c}^t=[\varepsilon-y_1,\varepsilon-y_2,\cdots,\varepsilon-y_N,\varepsilon+y_1,\varepsilon+y_2,\cdots,\varepsilon+y_N]$$
$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{D} & -\boldsymbol{D} \\ -\boldsymbol{D} & \boldsymbol{D} \end{bmatrix}$$
$$d_{ij} = (\boldsymbol{v}_i^t\boldsymbol{v}_j + 1)^p \qquad i,j = 1,\cdots,N$$

Note that no matter what the power p, this remains a quadratic optimization program. The rationale for the notation of V as the regularization constant is that it now appears as an upper bound on the β and a vectors. These quadratic programming problems can be very cpu and memory intensive. Fortunately, we can devise programs that make use of the fact that for problems with few support vectors (in comparison to the sample size), storage space is proportional to the number of support vectors. We use an active set method (Bunch and Kaufman, 1980) to solve this quadratic programming problem.

Although we may find $\hat{\boldsymbol{w}}$ in the SVR representation it is not necessary. That is, the SVR representation is not explicitly a, function of $\hat{\boldsymbol{w}}$. Similarly, if we use radial basis functions, the expansion is not an explicit function of $\hat{\boldsymbol{w}}$. That is, we can express the predicted values as:

$$y^{(p)} = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) \exp\left[-\gamma || \nu_i - x^{(p)} ||^2 \right] + b$$

In this case, the elements of the Q matrix above are

$$d_{ij} = \exp\left[-\gamma || \nu_i - \nu_j ||^2 \right]$$

and $\hat{w}$ cannot be explicitly obtained,

2. **Nonlinear Experiments**

We tried three artificial functions from (Friedman, 1991) and a problem (Boston Housing) from the UCI database. Because the first three problems are artificial, we know both the observed values and the truths.

Friedman #1 is a nonlinear prediction problem which has 10 independent variables that are uniform in [0,1]:

$$y=10\sin(\pi x_1 x_2)+20(x_3-.5)^2+10x_4+5x_5+n$$

where n is N(0,1). Therefore, only five predictor variables are really needed, but the predictor is faced with the problem of trying to distinguish the variables that have no prediction ability (x_6 to x_{10}) from those that have predictive ability (x_1 to x_5).

Friedman #2,#3 have four independent variables and are respectively:

$$\#2 \qquad y=(x_1^2+(x_2x_3-(1/(x_2x_4)))^2)^{1/2}+n$$

$$\#3 \qquad y=\tan^{-1}\left\{ \frac{x_2x_2-(1/x_2x_4)}{x_1} \right]+n$$

where the noise. is adjusted to give 3:1 ratio of signal power to noise power and the variables are uniformly distributed in the following ranges:

$$0 \le x_1 \le 100$$

$$20 \le (x_2/2\pi) \le 280$$

$$0 \le x_3 \le 1$$

$$1 \le x_4 \le 11$$

Boston Housing: This has 506 cases with the dependent variable being the median price of housing in the Boston area. There are twelve continuous predictor variables. This data was obtaining from the UCI database (anonymous ftp at ftp.ics.uci.edu in directory /pub/machine-learning-databases) In this case, we have no “truth”, only the observations.

In addition to the input space representation and the SVR representation, we also tried bagging. Bagging is a technique that combines regressors, in this case regression trees (Breiman, 1994). We used this technique because we had a local version available. Our implementation of regression trees is different than Breiman's but we obtained slightly better results. Breiman uses cross validation to pick the best tree and prune while we use a separate validation set. In the case of regression trees, the validation set was used to prune the trees.

Suppose we have test points with input vectors $x_i^{(p)}$ $i=1,M$ and make a prediction $y_i^{(p)}$

using any procedure discussed here. Suppose y_i is the actually observed value, which is the truth $G(\boldsymbol{x})$ plus noise. We define the prediction error (PE) and the modeling error (ME):

$$ME=\frac{1}{M}\sum_{i=1}^{M}(y_i^{(p)}-G(\boldsymbol{x}_i))^2$$

$$PE=\frac{1}{M}\sum_{i=1}^{M}(y_i^{(p)}-y_i)^2$$

For the three Friedman functions we calculated both the prediction error and modeling error. For Boston Housing, since the "truth" was not known, we calculated the prediction error only. For the three Friedman functions, we generated (for each experiment) 200 training set examples and 40 validation set examples. The validation set examples were used to find the optimum regularization constant in the feature space representation. The following procedure was followed. Train on the 200 members of the training set with a choice of regularization constant and obtain the prediction error on the validation set. Now repeat with a different regularization constant until a minimum of prediction error occurs on the validation set. Now, use that regularizer constant that minimizes the validation set prediction error and test on a 1000 example test set. This experiment was repeated for 100 different training sets of size 200 and validation sets of size 40 but one test set of size 1000. Different size polynomials were tried (maximum power 3). Second order polynomials fared best. For Friedman function #1, the dimensionality of feature space is 66 while for the last two problems, the dimensionality of feature space was 15 (for d=2). Thus the size of the feature space is smaller than that of the number of examples and we would expect that a feature space representation should do well.

A similar procedure was followed for the SVR representation except the regularizer constant U, ε and power p were varied to find the minimum validation prediction error. In the majority of cases p=2 was the optimum choice of power.

For the Boston Housing data, we picked randomly from the 506 cases using a training set of size 401, a validation set of size 80 and a test set of size 25. This was repeated 100 times. The optimum power as picked by the validations set varied between p=4 and p=5.

3. Results of experiments

The first experiments we tried were bagging regression trees versus support regression (Table I).

Table I. Modeling error and prediction error on the three Friedman problems (100 trials).

	bagging ME	SVR ME	bagging PE	SVR PE	# trials better
#1	2.26	.67	3.36	1.75	100
#2	10,185	4,944	66,077	60,424	92
#3	.0302	.0261	.0677	.0692	46

Rather than report the standard error, we did a comparison for each training set. That is, for the first experiment we tried both SVR and bagging on the same training, validation, and test set. If SVR had a better modeling error on the test set, it counted as a win. Thus

for Friedman #1, SVR was always better than bagging on the 100 trials. There is no clear winner for Friedman function #3. In Table II below we normalized the modeling error by the variance of the truth while the prediction error was normalized by the variance of the observed data on the test set. This shows that the worst modeling error is on Friedman #1 while the worst prediction error is on function #3.

Table II. Modeling error and prediction error normalized.

	bagging ME	SVR ME	bagging PE	SVR PE	# trials better
#1	.1850	.0542	.251	.130	100
#2	.0797	.0387	.376	.343	92
#3	.0473	.0409	.486	.497	46

Subsequent to our comparison of bagging to SVR, we attempted working directly in feature space. That is, we used F_1 as our approximating function with square loss and a second degree polynomial. The results of this ridge regression (Table III) are better than SVR. In retrospect, this is not surprising since the dimensionality of feature space is small (f=66 for Friedman #1 and f=15 for the two remaining functions) in relation to the number of training examples (200). 'Ibis was due to the fact that the best approximating polynomial is second order. The other advantages of the feature space representation in this particular case are that both PE and ME are mean squared error and the loss function is mean squared error also.

Table III. Modeling error for SVR and feature space polynomial approximation.

	SVR	feature space
#1	.67	.61
#2	4,4944	3051
#3	.0261	.0176

We now ask the question whether U and ε are important in SVR by comparing the results in Table I with the results obtaining by setting ε to zero and U to 100,000 making the regularizer insignificant (Table IV). On Friedman #2 (and less so on Friedman #3), the proper choice of ε and U are important.

Table IV. Comparing the results above with those obtained by setting ε to zero and U to 100,000 (labeled suboptimum).

	optimum ME	suboptimum ME
#1	.67	.70
#2	4,944	34,506
#3	.0261	.0395

For the case of Boston Housing, the prediction error using bagging was 12.4 while for SVR we obtained 7.2 and SVR was better than bagging on 71 out of 100 trials. The optimum power seems to be about five. We never were able to get the feature

representation to work well because the number of coefficients to be determined (6885) was much larger than the number of training examples (401).

One important characterization of a learning method is bias (sometimes termed bias-squared) and variance, Let $\bar{y}_i^{(p)}$ be the average prediction on the test set for test point i. $y_{ji}^{(p)}$ is the prediction for experiment j, test point i. Recall that the test sets are identical for the 100 experiments. but 100 different training and validation sets were used. When

$$\bar{y}_i^{(p)} = \frac{1}{100} \sum_{j=1}^{100} [y_{ji}^{(p)}]$$

then the bias is:

$$bias = \frac{1}{1000} \sum_{i=1}^{1000} [\ \bar{y}_i^{(p)} - G(\boldsymbol{x}_i^{(p)})]^2$$

The last term in the brackets is the truth so essentially the bias is mean square difference between the truth and the average prediction.

The variance is:

$$variance = \frac{1}{100} \frac{1}{1000} \sum_{i=1}^{100} \sum_{j=1}^{1000} [\ \bar{y}_j^{(p)} - y_{ji}^{(p)}]^2$$

The variance is the mean square difference between the prediction for each experiment at each test point and the average prediction for that test point (Table V). Thus, in comparing bagging to SVR. the largest effect is the reduction in bias.

Table V: the bias and variance for bagging and SVR.

	bias bagging	bias SVR	variance bagging	variance SVR
#1	1.74	.0548	.52	.61
#2	3111	90.28	7074	5327
#3	.1440	.0144	.008	.011

4. Linear Experiments

We investigated the behavior of support vector regression on two linear problems.

The first linear problem was as follows:

$$y = 2x_1 + x_2 + x_3$$

In addition there are 24 other components of the vector x, but y is not a (direct) function of these other 24 components. The x's are picked from a N(0,1) distribution and the x's are correlated with $E[x_1 x_j] = .8^j$ for $j = 2, \ldots, 30$. We used normal, uniform and Laplacian noise scaled to give different signal to noise ratios. Thus if there were no noise, our predictor would only have three variables. However, every variable has some predictive power and may help in the presence of noise. We use 60 training examples and 5000 test examples per run with 100 runs. This problem is similar in spirit to that of Breiman

(1994). We compared ordinary least squares (ols), SVR, and forward subset selection (fss) In fss, we first find the best (of thirty variables) in doing an ols prediction of the observed data. We fix that one variable and then find the next variable that used in combination with the first (fixed) variable, does the best ols prediction. This is continued until all thirty variables are used. Because the training error will continuously decrease as variables are added, we use a separate validation set of size 20% of the training set size. Every time we find an additional independent variable to add, we test on the validation set. A plot of validation set performance (mean square error) versus number of variables will show a minimum of the mean squared error and that is the optimum number of variables to be used. We then find the performance on a test set of size 5000 using the optimum set of variables predicted by the validation set.

We report the prediction error for the three types of noise and three prediction procedures (Table VI):

Table VI. Prediction error for normal, uniform, and Laplacian noise using ordinary least squares (ols), support vector regression (SVR), and forward subset selection (fss) for different signal-to-noise ratios (SNR).

	normal			**Laplacian**			**uniform**		
SNR	**ols**	SVR	fss	**ols**	SVR	fss	**ols**	SVR	fss
.8	45.8	29.3	28.0	40.8	25.4	24.5	39.7	28.1	24.1
1.2	20.0	14.9	12.3	18.1	12.5	10.9	17.6	12.8	11.7
2.5	4.61	3.97	3.12	4.17	3.26	2.48	4.06	3.60	2.76
5.0	1.15	1.33	.768	1.04	.516	.599	1.02	1.08	.617

Although SVR is better than ols, fss and SVR gives similar performance. At signal to noise ratios larger than 5, forward subset selection is better than ols but similar to SVR. The above experiment was for a training set of size 60. For training sets of size 200, forward subset selection gives the best results.

The above linear problem has a special structure, namely that there there are really only three predictor variables and there are a total of thirty to choose from, so it might be expected that forward subset selection would perform well.

We therefore tried the linear prediction problem where the output is a function of all the variables:

$$y=\sum_{i=1}^{30} x_i+n$$

In this case, both ols and fss give similar results. SVR is never worse and sometimes slightly better at low SNR's. However, at high SNR's, SVR is worse than ols or forward subset selection.

5 Conclusions

Support vector regression was compared to bagging and a feature space representation on four nonlinear problems. On three of these problems a feature space representation was

best, bagging was worst, and SVR came in second. On the fourth problem, Boston Housing, SVR was best and we were unable to construct a feature space representation because of the high dimensionality required of the feature space. On linear problems, forward subset selection seems to be the method of choice for the two linear problems we tried at varying signal to noise ratios.

In retrospect, the problems we decided to test on were too simple. SVR probably has greatest use when the dimensionality of the input space and the order of the approximation creates a dimensionality of a feature space representation much larger than that of the number of examples. This was not the case for the problems we considered. We thus need real life examples that fulfill these requirements.

6. **Acknowledgements**

This project was supported by ARPA contract number N00014-94-C-1086.

7. **References**

Leo Breiman, "Bagging Predictors", Technical Report 421, September 1994, Department of Statistics, University of California Berkeley, CA Also at anonymous ftp site: ftp.stat.berkeley.edu/pub/tech-reports/421.ps.Z.

Jame R. Bunch and Linda C. Kaufman, " A Computational Method of the Indefinite Quadratic Programming Problem", *Linear Algebra and Its Applications,* Elsevier-North Holland, 1980.

Jerry Friedman, "Multivariate Adaptive Regression Splines", *Annal of Statistics,* vol 19, No. 1, pp. l-141

Alan J. Miller, *Subset Selection in Regression,* Chapman and Hall, 1990.

Gilbert Strang, *Introduction to Applied Mathematics,* Wellesley Cambridge Press, 1986.

Vladimir N. Vapnik, The *Nature of Statistical Learning Theory,* Springer, 1995.

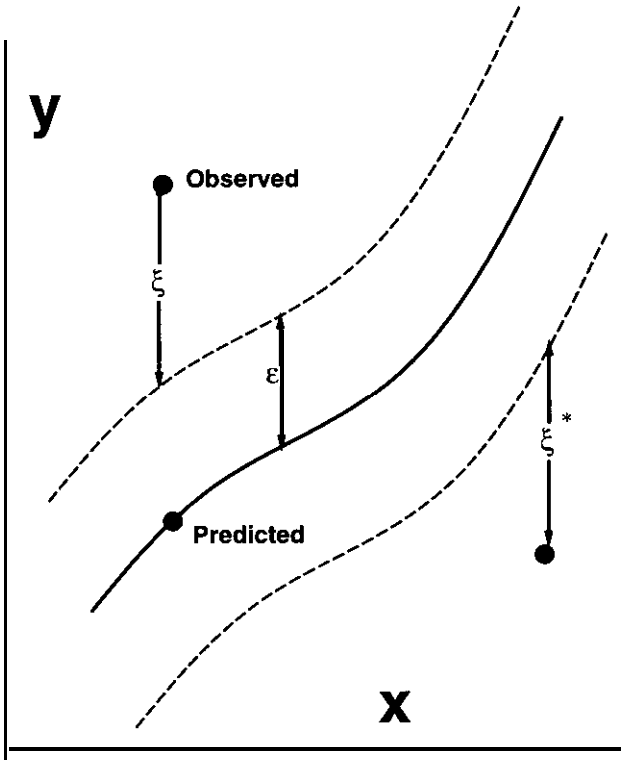

Figure 1: The parameters for the support vector regression.