Available online at www.sciencedirect.com

SCIENCE DIRECT®

NEUROCOMPUTING

Neurocomputing 69 (2006) 1749–1753

www.elsevier.com/locate/neucom

Letters

Estimation of software project effort with support vector regression

Adriano L.I. Oliveira*

Department of Computing Systems, Polytechnic School of Engineering, Pernambuco State University, Rua Benfica, 455, Madalena, 50.750-410 Recife – PE, Brazil

Received 28 October 2005; received in revised form 18 November 2005; accepted 19 December 2005
Available online 20 February 2006
Communicated by R.W. Newcomb

Abstract

This paper provides a comparative study on support vector regression (SVR), radial basis functions neural networks (RBFNs) and linear regression for estimation of software project effort. We have considered SVR with linear as well as RBF kernels. The experiments were carried out using a dataset of software projects from NASA and the results have shown that SVR significantly outperforms RBFNs and linear regression in this task.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Support vector regression; RBF network; Software effort estimation; Regression

1. Introduction

Software is ubiquitous and represents an important fraction of the budget of businesses and government [2]. In spite of the efforts towards improving software development, the failure rate of software projects is still high [2]. It is estimated that these failures have cost between $25 billion and $75 billion to the US economy over the last five years [2]. One of the important causes of software project failures is the inaccurate estimates of needed resources [2].

In the problem of software effort estimation, one builds a regression model using as input variables a number of features related to software development, such as the software size and the methodology [1,4,6,7]. The output is the total effort in man-months considering both programming and management activities. The regression model is trained with a number of past projects and is subsequently employed to predict the effort of novel projects. The software effort estimation problem has been tackled using both traditional methods such as the COCOMO [4] and, more recently, radial basis function (RBF) neural networks [7].

Support vector regression (SVR) is a kernel method for regression based on the principle of structural risk minimization [8,3]. Kernel methods have outperformed more traditional techniques in a number of problems, including classification and regression [8,3]. For example, it has been shown that SVR outperforms other techniques, including RBF and MLP neural networks, in forecasting of chaotic time series [5].

In this paper, we propose the use of SVR for the estimation of software project effort. In our simulations, we have used the well-known NASA software project dataset [1,7] and have considered SVR with both linear and RBF kernels. Our results are compared to linear regression as well as to RBFN results for this dataset reported in the literature [7].

2. The regression methods

In a regression problem we are given a training dataset $\{(x_1, y_1), \ldots, (x_l, y_l)\} \subset \chi \times \mathbb{R}$, where χ denotes the space of the input patterns, e.g., $\mathbb{R}^d$. The goal of regression is to find the function $f(x)$ that best models the training data. In our case, we are interested in building a regression model based on the training data to use it subsequently to predict the total effort in man-months of future software projects.

*Tel.: +55 81 99764841; fax: +55 81 34137749.
E-mail address: adriano@dsc.upe.br.

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.12.119

In linear regression, this is done by finding the line that minimizes the sum of squares error on the training set.

2.1. Support vector regression

In this work we propose to use ε-SVR, which defines the ε-insensitive loss function. This type of loss function defines a band around the true outputs sometimes referred to as a tube, as shown in Fig. 1. The idea is that errors smaller than a certain threshold $\varepsilon > 0$ are ignored. That is, errors inside the band are considered to be zero. On the other hand, errors caused by points outside the band are measured by variables ξ and ξ^*, as shown in Fig. 1.

In the case of SVR for linear regression, $f(x)$ is given by $f(x) = \langle w, x\rangle + b$, with $w \in \chi, b \in \mathbb{R}$. $\langle \cdot, \cdot \rangle$ denotes the dot product. For the case of nonlinear regression, $f(x) = \langle w, \phi(x)\rangle + b$, where ϕ is some nonlinear function which maps the input space to a higher (maybe infinite) dimensional feature space. In ε-SVR, the weight vector w and the threshold b are chosen to optimize the following problem [8]:

$$\text{minimize}_{w,b,\xi,\xi^*} \quad \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{l}(\xi_i + \xi_i^*),$$
$$\text{subject to} \quad \begin{cases} (\langle w, \phi(x_i)\rangle + b) - y_i \leqslant \varepsilon + \xi_i, \\ y_i - (\langle w, \phi(x_i)\rangle + b) \leqslant \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geqslant 0. \end{cases} \quad (1)$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. ξ and ξ^* are called *slack variables* and measure the cost of the errors on the training points. ξ measures deviations exceeding the target value by more than ε and ξ^* measures deviations which are more than ε below the target value, as shown in Fig. 1. The idea of SVR is to minimize an objective function which considers both the norm of the weight vector w and the losses measured by the slack variables (see Eq. (1)). The minimization of the norm of w is one of the ways to ensure the flatness of f [8].

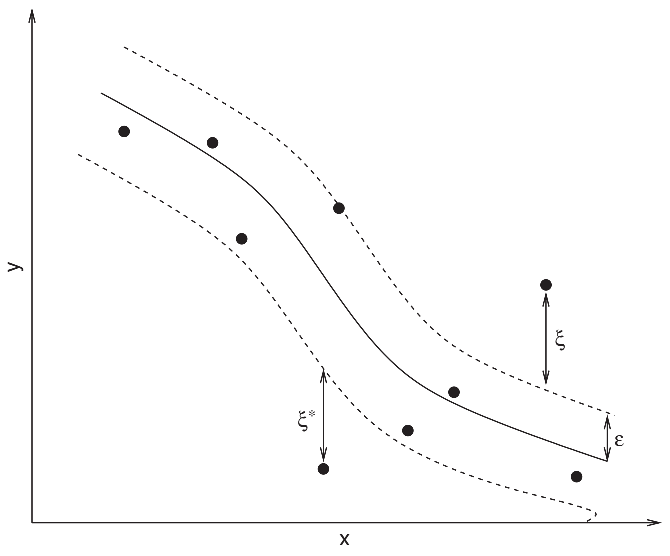

Fig. 1. Regression using ε-SVR.

The SVR algorithm involves the use of Lagrangian multipliers, which rely solely on dot products of $\phi(x)$. This can be accomplished via *kernel functions*, defined as $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle$. Thus, the method avoids computing the transformation $\phi(x)$ explicitly. The details of the solution can be found in [8].

In this work, we consider SVR with both linear and RBF kernels. The linear kernel is computed as $K(x_i, x_j) = \langle x_i, x_j\rangle$, whereas the RBF kernel is computed as $K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2), \gamma > 0$. In most problems, the parameters C and ε significantly influence ε-SVR generalization performance. For RBF kernels, the parameter γ must also be carefully selected.

2.2. Radial basis functions networks

This subsection briefly reviews the SG algorithm for training RBF neural networks [7]. This algorithm was used for software project effort estimation by Shin and Goel [7] and we intend to compare it to SVR in this task. The SG algorithm is used to train standard RBFNs with Gaussian units for function approximation [7]. The RBFN model is completely defined by the parameters m—the number of basis functions, σ—the common width of the basis functions, $\boldsymbol{\mu}$—the centers of the basis functions, and $\mathbf{w}$—the weights from the m basis functions to the output nodes.

The SG has an important parameter, δ, which is a measure of the closeness between the interpolation matrix and its approximation by m vector terms [7]. The first step of the SG algorithm consists in, given the common σ and δ, determining the minimum number of basis functions m that provide the desired degree of approximation [7]. Subsequently, the algorithm computes the centers $\boldsymbol{\mu}$ of the m basis functions using QR factorization. Finally, the weights are computed using the pseudoinverse [7]. The SG algorithm has two critical parameters which must be carefully selected for boosting performance, namely, δ and the common σ.

3. Experiments

The regression methods considered in this paper were compared using the well-known NASA software project dataset, reproduced in Table 1 [1,7]. This dataset consists of two independent variables—Developed Lines (DL) and Methodology (ME)—and one dependent variable—Effort (Y). DL is in KLOC (thousands of lines of codes) and effort is given in man-months [1,7].

DL is a measurement of the size of the program which considers both new lines of source code and code re-used from other projects. This gives a more satisfying measurement for the size of a software, according to Bailey and Basili [1]. Thus, DL is given by the number of developed lines of source code with comments (new lines with comments) plus 20% of re-used lines [1]. The ME variable was computed considering the development methodologies used in each software project. Examples of factors involved in computing ME were (a) the adoption of formal test plans, (b) formal documentation and (c) formal training [1].

In this work we are interested in estimating the effort of future projects, where the effort is given in man-months. The project effort includes both programming and management time. It was defined to be measured from the beginning of the design phase through acceptance testing and to include programming, management and support hours [1]. Note that the effort directly influences the cost of the software.

The simulations were carried out using the Weka tool [9]. In Weka, SVR is implemented using the sequential minimal optimization (SMO) algorithm [3].

Table 1
NASA software project data [1,7]

Project no.	Independent variables		Dependent variable
	DL	ME	Effort (Y)
1	90.2	30	115.8
2	46.2	20	96.0
3	46.5	19	79.0
4	54.5	20	90.8
5	31.1	35	39.6
6	67.5	29	98.4
7	12.8	26	18.9
8	10.5	34	10.3
9	21.5	31	28.5
10	3.1	26	7.0
11	4.2	19	9.0
12	7.8	31	7.3
13	2.1	28	5.0
14	5.0	29	8.4
15	78.6	35	98.7
16	9.7	27	15.6
17	12.5	27	23.9
18	100.8	34	138.3

We have used leave-one-out cross-validation (LOOCV) to estimate the generalization error, as was done by Shin and Goel [7]. In LOOCV, the dataset is partitioned into k subsets where k is the number of samples in the datasets. Training is carried out using $k-1$ of the subsets and the kth is used for prediction. This process is repeated k times, each time employing a different project for prediction. The performance measures considered in our work are the same used by Shin and Goel [7], namely, the mean magnitude of relative error (MMRE), and PRED(25). The MMRE is given by the following equation:

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{Y_i}. \tag{2}$$

PRED(25) is defined as the percentage of predictions falling within 25% of the actual known value, PRED(25). Y_i and $\hat{Y}_i$ are the ith output value and its estimate, respectively, and n is the number of data points.

We have carried out simulations considering two kinds of problems (as in [7]): (a) estimating the effort (Y) based only on DL and (b) estimating the effort (Y) using both independent variables (DL and ME).

In all SVR simulations, we have considered the following values of parameters C and ε: $C = \{10, 100, 1000\}$ and $\varepsilon = \{10^{-3}, 10^{-4}, 10^{-5}\}$. This means that, for linear kernels, we have carried out nine simulations using different pairs of (C, ε). For RBF kernels we have another important parameter, γ, for which we considered the values $\gamma = \{1, 10^{-1}, 10^{-2}\}$. Therefore, for SVR with RBF kernels we have carried out 27 simulations considering different combinations of parameters C, ε and γ.

Table 2 compares the methods considered in this paper with previous results obtained using RBF neural networks [7]. The RBF results were obtained using RBF networks trained with the SG algorithm, which we refer to as RBF–SG [7]. These simulations were carried out considering the estimation of effort (Y) based only on DL. For each method, both training and generalization (LOOCV) results

Table 2
Comparison of models for DL-Y data

Method/parameters	MMRE		PRED(25)	
	Tr.	Gener.	Tr. (%)	Gener. (%)
RBF-SG ($\delta = 1\%$, $\sigma = 0.55$, $m = 3$)	0.1579	0.1870	88.89	72.22
RBF-SG ($\delta = 2\%$, $\sigma = 0.55$, $m = 3$)	0.1579	0.1870	88.89	72.22
RBF-SG ($\delta = 0.1\%$, $\sigma = 0.25$, $m = 6$)	0.1450	0.1881	77.78	72.22
Linear regression	0.2110	0.2330	77.78	72.22
SVR linear ($C = 100$, $\varepsilon = 10^{-4}$)	0.1650	0.1790	83.33	83.33
SVR linear ($C = 10$, $\varepsilon = 10^{-4}$)	0.1650	0.1810	83.33	83.33
SVR linear ($C = 100$, $\varepsilon = 10^{-3}$)	0.1670	0.1810	83.33	83.33
SVR RBF ($C = 100$, $\varepsilon = 10^{-3}$, $\gamma = 10^{-2}$)	0.1650	0.1890	83.33	83.33
SVR RBF ($C = 100$, $\varepsilon = 10^{-4}$, $\gamma = 10^{-2}$)	0.1650	0.1890	83.33	83.33
SVR RBF ($C = 10$, $\varepsilon = 10^{-4}$, $\gamma = 10^{-1}$)	0.1680	0.1900	83.33	83.33

RBF results from [7]. For each regression method, the table presents both the training and generalization (LOOCV) results.

are shown. The results of this table shows that SVR outperforms both linear regression and RBF networks in this dataset. The best results were obtained by an SVR with linear kernel. Note that SVR performance does not depend significantly on the parameters. It is interesting to observe also that RBFN performance on the *training set* is better than that of SVR, which may indicate that RBFN models are overfitting the training data.

Table 3 provides a detailed comparison of the best SVR with the best RBFN in terms of the estimation of the effort for each software project. This table presents the estimated effort for each project (using LOOCV) along with the PRED(25) indicator which indicates, for each project, whether the estimation $\hat{Y}$ was or not within 25% of the known value Y. Tables 2 and 3 show that the best SVR obtained LOOCV MMRE = 0.1790 and PRED(25) = 83.33%, whereas the best RBF results were MMRE = 0.1870 and PRED(25) = 72.22% [7]. This means that the best SVR was able to predict 15 out of 18 projects within 25% of the actual known effort, whereas the best RBF predicted only 13 projects within 25% of the known effort, as shown in Table 3.

We have also carried out simulations considering the problem of estimating the effort (Y) using both input variables, namely, DL and ME. The results of our simulations for this case are shown in Table 4 along with results for RBFNs reported in the literature [7]. In this case, SVR has also outperformed both linear

Table 3
Comparison of actual and estimated efforts for the best RBFN and SVR models. RBF results from [7]

Project number	Actual	Predicted (RBF-SG)		Predicted (SVR)	
	Y	$\hat{Y}$	PRED(25) ind.	$\hat{Y}$	PRED(25) ind.
1	115.8000	124.9378	1	123.997	1
2	96.0000	68.9353	0	64.5460	0
3	79.0000	73.8473	1	64.9510	1
4	90.8000	87.2084	1	75.7560	1
5	39.6000	49.5555	0	44.141	1
6	98.4000	107.3230	1	93.313	1
7	18.9000	18.7930	1	19.7620	1
8	10.3000	16.0970	0	16.6640	0
9	28.5000	32.4551	1	31.1690	1
10	7.0000	5.9942	1	6.4520	1
11	9.0000	7.2109	1	7.8230	1
12	7.3000	12.4829	0	13.028	0
13	5.0000	5.0021	1	5.3510	1
14	8.4000	8.4081	1	9.2560	1
15	98.7000	120.5762	1	108.3310	1
16	15.6000	14.3626	1	15.2510	1
17	23.9000	17.8597	0	19.0950	1
18	138.3000	104.5607	1	128.9450	1
MMRE [PRED(25)]		0.187 (72.22%)		0.179 (83.33%)	

Table 4
Comparison of models for DL-ME-Y data

Method/parameters	MMRE		PRED(25)	
	Tr.	Gener.	Tr. (%)	Gener. (%)
RBF-SG ($\delta = 1\%$, $\sigma = 0.85$, $m = 4$)	0.1448	0.2584	88.89	72.22
RBF-SG ($\delta = 2\%$, $\sigma = 0.75$, $m = 4$)	0.1470	0.2474	88.89	72.22
RBF-SG ($\delta = 0.1\%$, $\sigma = 0.85$, $m = 7$)	0.0870	0.1907	77.78	72.22
Linear regression	0.2110	0.2330	77.78	72.22
SVR linear ($C = 10$, $\varepsilon = 10^{-4}$)	0.1480	0.1650	94.44	88.89
SVR linear ($C = 100$, $\varepsilon = 10^{-3}$)	0.1490	0.1740	88.89	88.89
SVR linear ($C = 100$, $\varepsilon = 10^{-4}$)	0.1480	0.1740	94.44	83.33
SVR RBF ($C = 100$, $\varepsilon = 10^{-3}$, $\gamma = 10^{-2}$)	0.1380	0.1780	88.89	83.33
SVR RBF ($C = 100$, $\varepsilon = 10^{-4}$, $\gamma = 10^{-2}$)	0.1370	0.1790	88.89	83.33
SVR RBF ($C = 10$, $\varepsilon = 10^{-4}$, $\gamma = 10^{-1}$)	0.1200	0.1800	83.33	83.33

RBF results from [7]. For each regression method, the table presents both the training and generalization (LOOCV) results.

regression and RBFN. The best SVR has achieved PRED(25) = 88.89%, whereas the best RBFN obtained 72.22% [7]. Note also that the use of another input variable (ME) enabled the improvement of both MMRE and PRED(25). Using both DL and ME, the best SVR achieved MMRE = 0.165 and PRED(25) = 88.89% (Table 4). On the other hand, considering only DL as input variable, the best SVR obtained MMRE = 0.179 and PRED(25) = 83.33% (Table 2).

4. Conclusions

This paper has investigated the use of support vector regression for estimation of software project effort. We have carried out simulations using the well-known dataset of software projects from NASA. The results were compared to both linear regression and RBFNs [7]. We have considered the estimation of effort based solely on the number of developed lines (DL) as well as based on both DL and the methodology (ME). In both cases, our simulations have shown that SVR outperforms both linear regression and RBFNs in this dataset.

Acknowledgment

The author would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] J.W. Bailey, V.R. Basili, A meta model for software development resource expenditure, in: Proceedings of the Fifth International Conference on Software Engineering, San Diego, California, USA, 1981, pp. 107–116.

[2] R.N. Charette, Why software fails, IEEE Spectrum 42 (9) (2005) 42–49.

[3] G.W. Flake, S. Lawrence, Efficient SVM regression training with SMO, Mach. Learn. 46 (1–3) (2002) 271–290.

[4] C.F. Kemerer, An empirical validation of software cost estimation models, Commun. ACM 30 (5) (1987) 416–429.

[5] S. Mukherjee, E. Osuna, F. Girosi, Nonlinear prediction of chaotic time series using support vector machines, in: Proceedings of IEEE Workshop on Neural Networks for Signal Processing (NNSP'97), Amelia Island, FL, USA, 1997, pp. 511–520.

[6] M. Shepperd, C. Schofield, Estimating software project effort using analogies, IEEE Trans. Software Eng. 23 (11) (1997) 736–743.

[7] M. Shin, A.L. Goel, Empirical data modeling in software engineering using radial basis functions, IEEE Trans. Software Eng. 26 (6) (2000) 567–576.

[8] A.J. Smola, B. Scholkopf, A tutorial on support vector regression, Stat. Comput. 14 (3) (2004) 199–222.

[9] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufmann, San Francisco, 2005.

Adriano L. I. Oliveira obtained his B.Sc. degree in Electrical Engineering and M.Sc. and Ph.D. degrees in Computer Science from the Federal University of Pernambuco, Brazil in 1993, 1997, and 2004, respectively. In 2002 he joined the Department of Computing Systems of Pernambuco State University as an Assistant Professor. He is also a Systems Analyst of Pernambuco Court of Accounts since 1995, where he works as an information systems auditor.

His current research interests include machine learning, pattern recognition, data mining, and applications of these techniques to information systems, software engineering, and biomedicine.