Australian Journal of Basic and Applied Sciences, 6(6): 15-20, 2012
ISSN 1991-8178

Fine Tuning on Support Vector Regression Parameters for Personalized English Word-Error Correction Algorithm

[1]Abu Bakar Hasan, [1]Tiong Sieh Kiong, [1]Johnny Koh Siaw Paw, [1]Emillino Tasrip and [2]Muhammad Sufyian Mohd Azmi

[1]College Of Engineering, Universiti Tenaga Nasional, 43009 Kajang, Malaysia
[2]College of Information Technology, Universiti Tenaga Nasional, 43009 Kajang, Malaysia

Abstract: A better understanding on word classification and regression could lead to a better detection and correction technique. We used different features or attributes to represent a machine-printed English word, and support vector machines is used to evaluate those features into two class types of word: *correct* and *wrong* word. Our proposed support vectors model classified the words by using fewer words during the training process because those training words are to be considered as personalized words. Those *wrong* words could be replaced by *correct* words predicted by the regression process. Our results are very encouraging when compared with Microsoft's spell checker, and further improvement is in sight.

Key words: statistical theory, support vector machines, artificial intelligence, FPGA

INTRODUCTION

Support vector machines (SVM) are a useful technique for data classification (Vapnik, 1998; Chang, *et al.*, 2011) and regression (Vapnik, 1998; Smola, *et al.*, 2003). SVM requires data to be presented as a vector of real numbers and scaling of data before applying SVM (Hsu, *et al.*, 2010). In supervised learning problems, feature selection is important for generalization performance and running time requirements (Weston, *et al.*, 2001; Dasgupta, *et al.*, 2007), whereby a subset of the features available from the data are selected for application of a learning algorithm (Blum, *et al.*, 1997; Weston, *et al.*, 2001). Selecting an optimal set of features is in general difficult, both theoretically and empirically (Kohavi, *et al.*, 1997). In machine learning, satisfactory features selection is typically done by cross-validation, grid search and automatic scripts method (Hsu, *et al.*, 2010; Chang, *et al.*, 2011), that is, in doing so it eliminates the irrelevant features or variables (Alan, 2003). In our previous work (Abu, *et al.*, 2012) on support vector regression, we found that it was not a direct mechanism in getting the *correct* word as the replacement for the *wrong* word. Hence, in this paper, we are proposing to fine tuning one of the eleven support vector parameters to meet the objective, that is, to be able to replace a *wrong* word with the *correct* word at a better rate. We have selected the parameter *typ1*, that is, by setting its value equals to zero for all data will technically meant that the word-error is independent of the first character's type (vowel or consonant) in a word. This paper has been organized in such a way that, in the next section, we provide brief literature survey on materials and methods, experimental results and discussion, and finally the conclusion.

MATERIALS AND METHODS

Support vector machines (SVM) can be generally divided into two: support vector classification (SVC) and regression (SVR). SVM is based on the structural risk minimization principle (SRM) from the statistical learning theory (Vapnik, 1998). SVM is a set of related supervised machine learning methods used for classification, on the other hand, the empirical risk minimization principle, which is used by neural network to minimize the error on the training data, the SRM minimizes a bound on the testing error, thus allowing SVM to generalize better than conventional neural network (Gunn, 1997). Apart from the problem of poor generization and overfitting in neural network, SVM also address the problem of efficiency of training and testing, and the parameter optimization problems frequently encountered in neural network (Cristiani, *et al.*, 2000). Support vectors in SVM can be categorized into two types: the training samples that exactly locate on the margin of the separation hyperplane, and the training samples that locate beyond their corresponding margins, whereby the later is regarded as misclassified samples (Zhan, *et al.*, 2005). At the moment SVM usage generally covers data classification and regression only (Chang, *et al.*, 2011; Harris, *et al.*, 1997). Given training data set represented by [{x_i, y_i}, for i= 1, 2 …., *m*], where $x_i \in R^n$ represents an n-dimensional input vectors, and $y_i \in R$. We construct a linear regression function,

$$g(x) = \boldsymbol{w}.\boldsymbol{x}_i + b \qquad (1)$$

Corresponding Author: Abu Bakar Hasan, College Of Engineering, Universiti Tenaga Nasional, 43009 Kajang, Malaysia.
E-mail: abakarh@uniten.edu.my.

The ***w*** and *b* in equation (1) are obtained by solving an optimization problem:

$$\text{Min } [½\|\mathbf{w}\|^2 + C \textstyle\sum_{i=1}^{m} (\xi_i + \xi_i^*)] \tag{2}$$

The optimization criterion penalizes data points whose *y*-values differ from *g*(*x*) by more than ε. The slack variables, ξ_i and ξ_i^*, correspond to the size of the positive and negative deviation respectively. In most cases the optimization problem (equation 2) can be solved more easily in its dual formulation. Hence, the standard dualization method utilizing Langrange multipliers (Smola, *et al.*, 2003) is used giving the dual optimization problem. Hence, it can shown that

$$w = \textstyle\sum_{i=1}^{m} \quad (\alpha_i - \alpha_i^*)\, x_i \tag{3}$$

Finally the ε-insensitive regression loss function is given by:

$$g(x_j) = \textstyle\sum_{i=1}^{m} \quad (\alpha_i - \alpha_i^*)(\mathbf{w}.\mathbf{x}_i + b) = [\textstyle\sum_{i=1}^{m} \quad (\alpha_i - \alpha_i^*) K(x_i , x_j) + b] \tag{4}$$

whereby $K(x_i , x_j)$ is the Kernel function and parameter x_i is the training or test data while x_j is the input data for prediction.

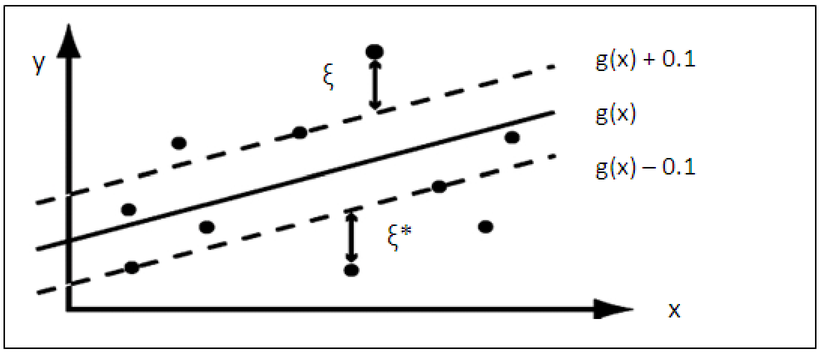

Fig. 1: The ε-insensitive loss function and the role of slack variables ξ and ξ* with ε=0.1 (Basak, *et al.*, 2007)

For our case study, we have chosen the radial basis function (RBF) as the kernel as it has less numerical difficulties (Vapnik, 1998) and has the properties defined by kernel function $K(x_i, x_j)= \exp(-\gamma \mathrm{I}x_i\text{-}x_j\mathrm{I}^2)$, kernel bandwidth parameter γ, and penalty parameter *C* (Hsu, *et al.*, 2010). A recent result (Keerthi, *et al.*, 2003) shows that if RBF is used with model selection, then there is no need to consider the linear kernel. We defined *word* literally as a word that could be correctly read or identified without any ambiguity, and non-word is defined as vice-versa. The scope of our study comes under the isolated-word error category (Kukich, 1992) and specifically on machine printed English word only. In this study, we are proposing a model to represent a word to meet the SVR requirements, with a maximum of eleven attributes or features namely numbers of consonant or non-vowel (*nvow*) and vowel (*vow*) character in them, total letter length (*len*) and their weight (*wgt*), weight of first (*wgt1*) and last letter (*wgtn*), position-vowel factor (*pvf*), sum of position-consonant-vowel factor (*pcf*), position factor (*pf*), real value (*RV*) and lastly, the type for the first letter (*typ1*) in the word. In order to classify these words correctly, hereby we are proposing five rules as stated below:

Rule 1: The number of non-vowel and vowel in a word equals to the length of the word.

Rule 2: The weight of each word is equal to the word own weight. *We define each character has a specific weight: (A, a)=1, (B, b)=2, (C, c)=3, (D, d)=4, until (Z, z)=26, and (')=0*

Rule 3: The type for first character in any word is set to 0.

Rule 4: Each word has a unique real value represented by parameter *RV*; otherwise the regression process is invalid.

Rule 5: If the weight of the first letter in the first word is different from the first letter of the second word, and consecutively, then either word is different.

To illustrate our proposed regression model, we take any English word, for example, the word *final*, then, with the help of Rule 1, 2 and 3, it can be shown that *nvow* = 3, *vow* = 2, *len* = 5, *wgt* = 42, *wgt1* = 6, *wgtn* = 12, *pvf* = 22, *pcf* = 108, *pf* = 130, *typ1* = 0 and *RV* = 0.553132238. To obtain the respective normalized values for each parameter, we use the use the following mathematical equation:

$$Nparameter = [Current\ Parameter - \min(parameter)]/[\max(parameter) - \min(parameter)] \tag{9}$$

For example, again for *final* it can be shown that:

$$NRV = [0.553132238-0.0744822387]/[8.416-0.0744822387] = 0.057381627 \qquad (10)$$

We used LibSVM (Hsu, *et al.*, 2010; Chang, *et al.*, 2011), which is a SVM learning tool package, for the training and testing process of the normalized words. The five rules above will mathematically play an important function in the classification and regression of each word. We are proposing 3-steps procedure for SVR:

Step 1: Train the normalized training data by setting the correct value for parameter C and γ obtained through cross-validation process, whereby RBF function was chosen.

Step 2: Predict the normalized Test data to obtain the mean squared error and squared correlation coefficient

Step 3: Fine tuning the chosen SVR parameter, if *Step 2* needs further improvements.

RESULTS AND DISCUSSION

In the study, we apply SVR on the test data set, after modeling the training data to obtain the relevant parameters. Remember that all our training data consists of words that are considered *correct* words. Meaning, if we wanted to replace any *wrong* word, technically we just need a *correct* word to do so. Thus, we are trying to avoid a situation whereby a *wrong* word is been used to correct a *wrong* word during the correction process. We observed that during the training of 1147 data, there was no support vector and *rho* = 1.0 giving the value of $g(x)$ equals to unity always for all values of parameters C and γ. During the testing process, again we did observe that the squared correlation coefficient (*SCC*) equals to one; indicating that the regression line perfectly fitted the training data (Basak, *et al.*, 2007) with parameter $\varepsilon = 0.1$. In other word, all the training were within the region of $-0.1 < \varepsilon < 0.1$, meaning that all the data were regressed without failed.

To elaborate further our proposed idea on fine tuning, we tested it on the *wrong* word *inal* in which its respective *correct* word was *final* (Table 1). Applying Rule 3, by setting parameter *typ1*=0, the *NRV* values were more consistent with respective to the other words *fina, finl, final* or *fial*. Thus, minimizing the regression error and directly make the regression accuracy higher. This phenomenon was also true for the other words that had been investigated in our work. Here, we defined regression error, δ as the absolute difference in *RV* values between any *wrong* words with its respective *correct* word.

Table 1: Selected Results Before and After Fine Tuning of *typ1*on *NRV*

Label	*Typ1 (before)*	*Typ1 (after)*	*NRV (before)*	*NRV (after)*	word
1	0	0	0.057381627	0.057381627	*final*
0	0	0	0.016201122	0.016201122	*fina*
0	0	0	0.04795683	0.04795683	*finl*
0	1	0	0.172170813	0.052288551	*inal*
0	0	0	0.043404009	0.043404009	*fial*

Note: Label 0 represents *wrong* words, Label 1 otherwise

Our mechanism to correct any *wrong* word with the *correct* word now became more mathematically self explanatory. Looking at Table 1, we observed that the regression errors for the words *inal* and *final* before and after the fine tuning of *typ1* were 0.114789186 and 0.005093076 respectively. Thus, we believed that the latter result will provide a better correction possibility for *inal* to be replaced by *final*. With this finding, we applied the fine tuning process on *typ1* for all our data, and we found that each of the 1147 words has a unique *NRV* value, thus fulfilling Rule 4. Subsequently, our proposed regression model led into an isolated-word-error correction algorithm (Table 2) that could correct a *wrong* word based on its normalized real value, *NRV*; regardless the first character's type in that word, *typ1*.

Our results involving the word-error classification and regression were put together with the final part of this work, that is, the correction of *wrong* words. Figure 3 shows the block diagram of the whole program that was embedded into an FPGA development board (Figure 2) for verification and testing purposes.

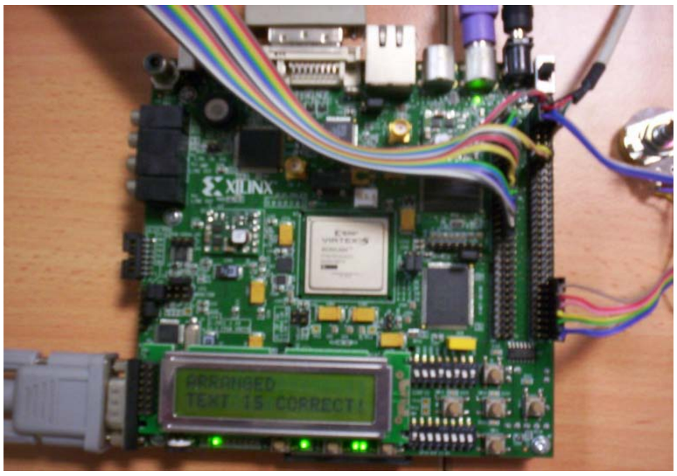

Fig. 2: Hardware Implementation of the Proposed Algorithm

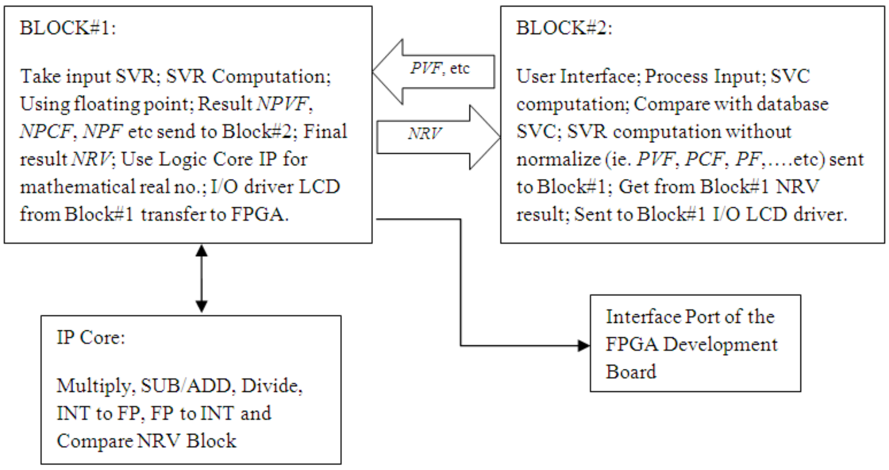
BLOCK#1:
Take input SVR; SVR Computation; Using floating point; Result NPVF, NPCF, NPF etc send to Block#2; Final result NRV; Use Logic Core IP for mathematical real no.; I/O driver LCD from Block#1 transfer to FPGA.
PVF, etc
NRV
BLOCK#2:
User Interface; Process Input; SVC computation; Compare with database SVC; SVR computation without normalize (ie. PVF, PCF, PF,….etc) sent to Block#1; Get from Block#1 NRV result; Sent to Block#1 I/O LCD driver.
IP Core:
Multiply, SUB/ADD, Divide, INT to FP, FP to INT and Compare NRV Block
Interface Port of the FPGA Development Board

Fig. 3: Block Diagram of the Hardware Development Language

We found that our results from the hardware implementation of the proposed correction algorithm meet our expectations, that is, the results are in line as the ones that come from the LIBSVM simulation. To do the *wrong* word correction, we are proposing a novel correction algorithm based on SVM principle. The next paragraph will discuss the said subject and the observed results in great details.

Table 2: Proposed SVM-based Word-error Correction Algorithm

Step1	Enter the *wrong* word
Step2	Calculate LEN_w , RV_w and NRV_w of the *wrong* word
Step 3	Calculate the respective correction error, δ_w between NRV_w with those NRV_c of *correct* words in the database, and $0.0001 \le \delta_w \le 0.05$ that meet $\lvert LEN_w- LEN_c\rvert \le \pm1$
Step 4	List down the *correct* word with first or last character the same as in the *wrong* word. Replace the *wrong* word with the chosen *correct* word

The worst-case scenario in an error-free communication system is for 1-bit error acceptance level, meaning that for any *wrong* word that used 8-bit ASCII code our hypothesis for the maximum number of character to be misspelled or missing is one character per word only. Thus in Step 3 (Table 2), we clearly state the condition: $\lvert LEN_w- LEN_c\rvert < \pm1$ must be observed during the calculation of δ_w. For example, for *fina* (LEN_w= 4 characters), Step 3 will check for LEN_c = 3, 4 and 5 characters. To test the assumption mentioned above, we apply it to our 1147 *correct* words in the data file, and Table 3 shows the values of δ_w for selected *wrong* words of *fial*, *inal*, *fimal* and *finalk*. Table 4 shows the full lists of *correct* word for *wrong* word *fina* with LEN_c = 5 only.

After Step 4 (Table 2) was carried out, Table 5 shows comparison between our proposed word-error correction algorithm with the one from Microsoft's spell checker that is presently available at our laboratory. The said table indicates that our proposed algorithm has a better performance for *inal* and *fimal*.

Table 3: Selected *correct* words of *fina, inal, fial* and *finl* After Step 3

len$_c$	*correct* word	*fina*	*inal*	*fimal*	*finalk*
5	*final*	0.046319836	0.005728692	0.000300328	0.021922943
4	*year*	0.193327	0.152735856	0.147307492	0.125084221
6	*project*	0.205922032	0.165330888	0.159902524	0.137679253
7	*progress*	0.25683272	0.216241576	0.210813212	*0.188589941*
6	*report*	0.203047182	0.162456038	0.157027674	*0.134804403*
6	*period*	0.08972734	0.049136195	0.043707832	*0.021484561*
5	*title*	0.084064544	0.060831572	0.055403208	*0.033179937*
7	*nuclear*	0.164565451	0.123974307	0.118545943	*0.096322672*
6	*beyond*	0.040741907	0.00150763	0.005277601	*0.027500872*

Table 4: *Correct* words with 5 characters for *fina After Step 3*

Correct words	*Correct* words	*Correct* words
final	*build*	*Bangi*
being	*Datuk*	*Ahmad*
crude	*Korea*	*Hairi*
above	*Balui*	*Chong*
drive	*cable*	*email*
hence	*Baram*	*kuasa*
cubic	*Baleh*	*fifth*
fired	*basic*	*excel*
among	*going*	*graph*
cycle	*flora*	*bound*
board	*fauna*	*cause*
could	*force*	*Alang*
based		

Table 5: Comparison Results

wrong word	Possible *correct* words by our proposed correction algorithm	Possible *correct* words by Miscrosoft's spell checker
fina	*final, flora, fauna, force, fifth*	*fine, final, fin, finer, fiona*
inal	*final*	*anal, final, inlay, pinal, inhale*
fimal	*final*	*final, female, formal*
finalk	*final, faces, fuels, force*	*final, finals, finale, finally*

Finally, comparing our work with the present word correction technique presently available in the market, our proposed model is a personalized spelling correction word. Meaning that the total number of correct words in our data file is very much less than the ones in a typical English dictionary (Table 6), thus, the system is better in term of memory usage and cost saving. In real implementation, for the personalized word-error-correction system to start operation, the users need to enter in their data base, the minimum of 3000 to 5000 different *correct* words they normally used in life. We strongly believed that our proposed idea requirement was again better with respect to the claims that inferred male and female speak daily a total of 6073 and 8805 words respectively (Wikipedia#1, 2012), and the Wintertree software's American and British English spell checker contains 100,000 words (Wikipedia#3, 2012). We foresee that such personalized system could be used in the mobile environment where available memory is limited.

Table 6: How many words are there in a dictionary? (Wikipedia#2, 2012)

Type	No. of different words
Oxford English dictionary	220,000
Oxford "shorter" dictionary	163,000
Oxford Reference dictionary	115,000
Oxford English Mini-dictionary	40,000

Conclusion:

In this paper, we have demonstrated the SVR's parameter fine tuning on the word errors (covering the isolated-word error and non-word error only) that occurred in the English word. The use of SVM on the correction of the *wrong* word that could one day complement or improve the present English word spelling checker was also shown. Successful implementation on FPGA Development Board involving support vector classification, regression and correction has proven that the original idea was feasible.

REFERENCES

Alain, R., 2003. Variable Selection Using SVM-based Criteria, J. of Machine Learning Research, 3: 1357-1370.

Basak, D., S. Pal and C.P. Dipak, 2007. Support Vector Regression, Neural Information Processing-Letters and Reviews, 11(10): 203-224.

Blum, A.L. and P. Langley, 1997. Selection of relevant features and examples in machine learning, Artificial Intelligence, 97: 245-271.

Chang, C.C. and C.J. Lin, 2011. LIBSVM - A library for support vector machines. ACM Transaction on Intelligent System and Technology, 2(3): 1-27.

Cristiani, N. and S. TaylorJ, 2000. An Introduction to SVM and Other Kernel-based Learning Methods, Cambridge University Press, UK.

Dasgupta, A., P. Drineas, B. Harb, V. Josifovski and M.W. Mahoney, 2007. Feature selection methods for text classification, ACM, pp: 230-239.

Gunn, S., 1997. SVM for Classification and Regression, Technical Report, Speech and Intelligent Systems Research Group, University of Southampton, USA.

Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola and Vladimir Vapnik, 1997. Support Vector Regression Machines, Advances in Neural Information Processing Systems 9, NIPS 1996, MIT Press, pp: 155-161.

Hsu, C.W., C.C. Chang and C.J. Lin, 2010. A practical guide to support vector classification, Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, April 15.

Keerthi, S.S. and C.J. Lin, 2003. Asymptotic behavior of SVM with Gaussian kernel, Neural Computation, 15: 1667-1689.

Kohavi, R. and G.H. John, 1997. Wrappers for feature subset selection, Artificial Intelligence, 97: 273-324.

Kukich, K., 1992. Techniques of automatically correcting words in text, ACM Computing Surveys,

Smola, A.J. and B. Scholkopf, 2003. A Tutorial on Support Vector Regression. NeuroCOLT Technical Report TR-98-030, pp: 1-19.

Vapnik, V. N., 1998. Statistical Learning Theory, Wiley, New York.

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, T. Puggio and V. Vapnik, 2001. Feature Selection for SVMs. Advances in Neural Information Processing Systems, Vol. 13 (Cambridge, MA), MIT Press

Wikipedia#1 at www.funtrivia.com/askft/Question8771.html on 10 Jun 2012

Wikipedia#2 at http://itre.cis.upenn.edu/~myl/languagelog/archives/003420.html on 15 Jun 2012

Wikipedia#3 at www.wintertree-software.com/spell-check/dictionary-size.html

Zhan, Y. and D. Shen, 2005. Design efficient support vector machine for fast classification. J. Pattern Recognition Society, Pergamon, 38: 157-161.