Least Squares Support Vector Machine Classifiers Using PCNNs

Yongsheng Sang
Computational Intelligence Laboratory
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu 610054, P. R. China. and
Southwest University of Science and Technology
Mianyang 621010, P. R. China.
sangys@uestc.edu.cn

Haixian Zhang, Lin Zuo
Computational Intelligence Laboratory
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu 610054, P. R. China.
hxzhang@uestc.edu.cn, linlinzuo@gmail.com

***Abstract*—Least Squares Support Vector Machine (LS-SVM) is a modified version of traditional Support Vector Machine (SVM). LS-SVM considers equality constraints, therefore it solves a set of linear equations instead of quadratic programming problem in SVM. However, the sparseness of LS-SVM is lost due to it's ϵ-sensitive cost function. Sparseness can be obtained by applying a pruning method, which eliminates some vectors with smallest support values and retrains the remaining samples. But iterative retraining is a time-consuming process. Motivated by the fact that boundary samples are more significant for constructing a LS-SVM classifier, this paper proposes a method of using Pulse Coupled Neural Networks (PCNNs) to search boundary samples of original data sets. The original data sets are mapped into some PCNN neurons, and a firing algorithm is designed to determine which samples lie at boundary region. It gives a novel approach to impose sparsity for LS-SVM. Experiments show that the proposed method can effectively detect boundary samples and speed up LS-SVM classifiers.**

I. Introduction

Support vector machine proposed by Vapnik *et al* in [1] has been receiving increasing attention in recent years. SVM is an important machine learning methodology with good generalization ability. It has strong theoretical foundations and excellent empirical successes in many pattern recognition applications, such as pattern classification and function estimation [2] [3]. However, SVM does not scale well with respect to the size of training data. For standard SVM case, it is formulated as a quadratic programming (QP) problem. Given n training instances, the training time complexity of QP is the time $O(n^3)$ and its space complexity is at least $O(n^2)$. Hence, a major stumbling block is in scaling up the method to large data sets, such as those commonly encountered in data mining applications.

To reduce the time and space complexities, some modified methods for SVM are proposed. Decomposition methods are currently one of the major methods for training support vector machines. A popular approach to scale up SVM is by chunking [3]. However, chunking needs to optimize the entire set of non-zero Lagrange multipliers that have been identified, and the kernel matrix may still be too large to fit into memory. A method proposed in [4] suggests that optimizing only a fixed-size working set of the training data each time, while the variables corresponding to the other patterns are frozen. Going to the extreme, the sequential minimal optimization (SMO) algorithm [5] breaks the original QP into a series of smallest possible QPs, each involving only two variables. However, the convergence of SMO algorithm can be slow in large scale data cases.

A more radical approach is to avoid the QP altogether. J.A.K. Suykens *et al* [6] propose a modified SVM called LS-SVM, which has been investigated for classification and function estimation problems. Taking into account equality constraints, LS-SVM only needs to solve a set of linear equations. This method significantly reduces the computation complexity. However, the sparseness of LS-SVM solution is lost because of the choice of ϵ-sensitive cost function. A pruning method, called sparse LS-SVM, is proposed in [7] for imposing sparseness of LS-SVM by Suykens *et al*. Motivated by the fact that the LS-SVM support values are proportional to the errors at the data points, they remove a small amount of points with smallest support values and retrain the LS-SVM iteratively until the user-defined performance index degrades. But the iterative process is still very time-consuming for large size data set. Some other methods are proposed to improve LS-SVM, see [8], [9] etc. A general comparison of various pruning algorithms is given in [10]. It makes a conclusion that pruning based on absolute support values is still most attractive if one takes into account both the computational costs and classification accuracy.

Motivated by the fact that boundary samples are more significant for constructing a LS-SVM classifier, we propose a PCNN-based method to search boundary samples from original data set. PCNNs are developed as a result of studies from the visual cortex of cats and monkeys [13]. The method has been used for image smoothing, image segmentation, feature extraction etc [14]. Similar in spirit to image segmentation algorithms using PCNNs , we map the original data sets into some PCNN neurons, and design a firing algorithm

978–1–4244–1674–5/08/$25.00 © 2008 IEEE

to determine which samples lie at boundary region. Unlike processing pixels in image applications, the proposed method map more than one data samples into a neuron. But only those data points closest to PCNN neurons will be recorded. By this mean, we impose sparsity for our PCNN model and speed up its convergence. This contribution introduces a suitable method to impose sparseness for LS-SVM by selection of significant boundary points. The proposed method can speed up the training speed of LS-SVM classifiers.

This paper is organized as follows. In Section II, the LS-SVM classifiers are reviewed. In Section III, we proposed a modified PCNN model. The algorithm to detecting boundary samples for LS-SVM is given in section IV. The experimental results are reported in Section V. The conclusions are given in Section VI.

II. LS-SVM Classifiers

Given a training set of *n* data points $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where $x_k \in R^p$ is the kth input vector and $y_k \in \{+1, -1\}$ is the corresponding class label. Assuming the data points are linearly separable, there exists a linear classifier in input space such that $y(x) = Sign(w^T x + b)$. For nonlinearly separable cases, we employ the idea of mapping the data points into a high dimensional feature space by means of a nonlinear function $\varphi(.)$, and a separating hyperplane in feature space takes form as $y(x) = Sign(w^T \varphi(x) + b)$. To build an SVM classifier, one needs to find the optimal hyperplane between the two classes of training samples, which has the maximum margin of separation $1/||w||$. Obviously, the maximization of the margin is equivalent to the minimization of the Euclidean norm of w. In order to obtain the optimal hyperplane, LS-SVM needs to build an optimization problem

$$\min_{w,b,e} J_3(w, b, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{N} e_k^2. \tag{1}$$

Unlike traditional SVM, LS-SVM considers equality constraints

$$y_k[w^T \varphi(x_k) + b] = 1 - e_k, k = 1, ..., N. \tag{2}$$

The Lagrangian is constructed as

$$\begin{aligned} &L(w, b, e; \alpha) = J_3(w, b, e) - \\ &\sum_{k=1}^{N} \alpha_k \{y_k[w^T \varphi(x_k) + b] - 1 + e_k\}. \end{aligned} \tag{3}$$

According to Kuhn-Tucker conditions, one of the conditions for optimality is obtained as follows

$$\frac{\partial L}{\partial e_k} = 0 \rightarrow \alpha_k = \gamma e_k, k = 1, ..., N. \tag{4}$$

In Equation (4), α_k are Lagrange multipliers. α_k multipliers reflect the importance of the training samples. Their weights are proportional to the e_k, errors in the training samples. From Equ.(2) we can formulate e_k as

$$e_k = 1 - y_k \left(w^T \varphi(x_k) + b\right). \tag{5}$$

Those data samples closest to and farthest from the separating hyperplane have largest $|e_k|$. Obviously, these samples also have largest $|\alpha_k|$. Sparseness can be obtained for LS-SVM by applying a pruning method [7] , which eliminates some training samples based on the sorted $|\alpha_k|$ spectrum. By eliminating some vectors, represented by the smallest values from the $|\alpha_k|$ spectrum, the number of support vectors can be reduced. The irrelevant points are omitted, by iteratively leaving out the least significant vectors. As a result, the pruning method in [7] shows they keep an amount of data points closest to and farthest from the separating hyperplane. The remaining samples form a subset of the boundary samples. So if we can seek all the boundary samples which include the most significant vectors, then a sparse LS-SVM can be obtained. Motivated by this reason, we propose a PCNN-based method to search boundary samples as candidates of support vectors for LS-SVM. Therefore an sparse LS-SVM classifier on boundary samples can be obtained directly.

III. A Modified PCNN Model

The original Pulse-Coupled Neural Networks (PCNNs) were derived from studies on the cat's eye, which was proposed by Eckhorn et. al [13]. They had studied the cat visual cortex, the part of the brain that processes the information from the eye, and discovered that the midbrain in an oscillating way creates binary images that extract different features from the visual impression. Based on these so-called pulse images, the actual image is created in the cat brain. They proposed a neural network that simulated this behavior, and it was developed by many researchers. PCNNs can be applied in many fields, such as digital image processing, moving object recognition, communication and optimization etc [14], [15].

In a two dimensional PCNN model, the PCNNs consist of multiple nodes forming in a grid [14]. The nodes are coupled together with their neighbors within a radius r_0. A typical neuron has two input compartments, linking and feeding. The feeding compartment, F, receives both an external and a local stimulus, whereas the linking compartment, L, only receives a local stimulus. The feeding and linking are combined in a second order fashion to form the membrane voltage, U. This is then compared to a dynamic threshold Θ for determining the output.

To search boundary samples, we introduce some modifications into standard PCNN model. These modifications are reflected in the neuron architecture as illustrated by the Fig.1. The proposed model can be described as following equations:

$$F_{ij}[n] = \begin{cases} 0, & \text{neurons with data sample.} \\ S, & \text{neurons without data sample.} \end{cases} \tag{6}$$

$$L_{ij}[n] = \sum_{kl} W_{ijkl} Y_{kl}[n-1] \tag{7}$$

$$C_{ij}[n] = \begin{cases} 1, F_{ij}[n] > 0 \\ 0, Otherwise \end{cases} \tag{8}$$

$$U_{ij}[n] = F_{ij}[n] + \beta L_{ij}[n] \tag{9}$$

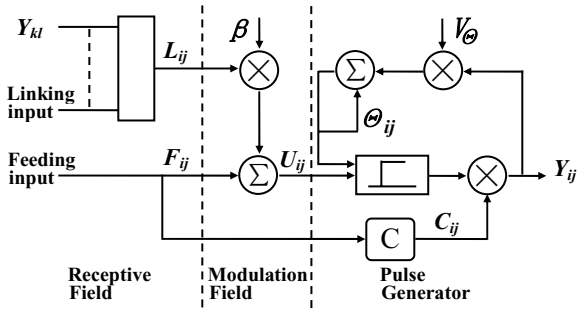

Fig. 1. Architecture of modified PCNN neuron.

$$
\begin{aligned}
Y_{ij}[n] &= Step(U_{ij}[n] - \Theta_{ij}[n])C_{ij}[n] \\
&= \begin{cases} 1, & \text{if } U_{ij}[n] > \Theta_{ij}[n] \text{ and } C_{ij}[n] = 1 \\ 0, & \text{if } U_{ij}[n] < \Theta_{ij}[n] \text{ or } C_{ij}[n] = 0 \end{cases}
\end{aligned}
$$

$$\Theta_{ij}[n] = e^{\alpha_F}\Theta_{ij}[n-1] + V_\Theta Y_{ij}[n] \tag{10}$$

The proposed PCNN neurons consist of three parts: the receptive field, the modulation field, and the pulse generator. In the input receptive field, see Equ.6, the feeding input F_{ij} is simplified to external input. The feeding input can be set to zero or a nonzero constant S according to the neuron having a data sample or not. The linking input L_{ij}, see Equ.7, is the sum of responses of the output from surrounding neurons. The feeding and linking are combined in an additive fashion to form U_{ij}. A control unit C_{ij}, see Equ.8, is introduced in our model, which is used to filter the output of the neurons, see Equ.10. The control unit promises our model stop after two times firing. At first firing time, all the neurons without data samples fire because their feeding inputs are initialized as a adequate big constant S. At the same time, their control unit outputs are 1 which make sure that they can emit pulses. These pulses make those neurons stored boundary samples fire at next time, yet the outputs of these neurons' control units are 0 which make they emit no pulses. The PCNNs firing process thus stop naturally. Data samples stored in the neurons fired at the second time are boundary samples.

IV. Searching Boundary Samples by PCNNs

In the following accounts, we illustrate our method with two dimensional data sets. Given a training set of *n* data points $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where (x_k, y_k) is the kth input vector. To search boundary samples for the data set, a PCNNs with $l \times m$ ($l << n, m << n$) neurons are necessary, which form a $l \times m$ grid. The first row of neurons are used to map those data samples with the smallest y values, and the lth row are used to map samples with the largest y values. The first column of neurons are used to map those data samples with the smallest x values, and the mth column are used to map those samples with the largest x values. Unlike processing pixels by one-to-one correspondence between pixels and neurons in image applications, we map more than one data samples into a PCNN neurons, yet only those data samples closest to the PCNN neurons are stored. By this mean, the sparseness for our PCNN model is obtained. Generally, there are two kinds of neurons in the our model: neurons with data samples and neurons without samples. Fig.2 shows how to map data samples into neurons. The dotted circles are neurons having data samples and the other circles are neurons without data samples. The little dots are data samples. For searching boundary samples effectively, two extra rows are added as the first row and the last row, and two extra columns are added too. The PCNN model thus scales up to $(l+2) \times (m+2)$, see Fig.2.

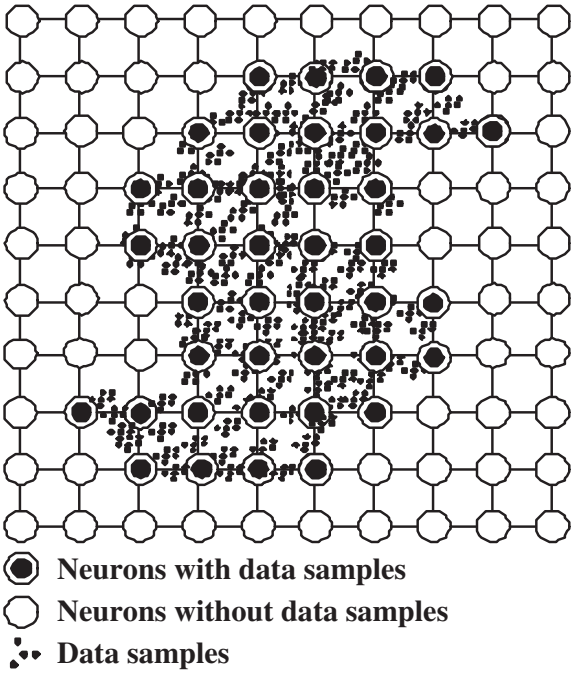

Fig. 2. Mapping data samples into PCNN neurons.

According to the neuron type, The feeding inputs can be arranged as two choices. For neurons without data points, F_{ij} are set to $S(S > \Theta_{ij}[0])$. The feeding inputs of neurons with data points are set to 0. At time t_0, all the linking inputs are 0 and only external sources are added to PCNN neurons. Thus, the neurons without data point will fire at time t_0. The control field C_{ij} is used to filter the neuron output. The control field of the neuron is set to 1 when $F_{ij} > 0$, otherwise set to 0. This control part makes the neurons without data points emit a pulse when they fire, yet the neurons with data point emit no pulses. It thus promise the PCNN Auto-wave stop when it meets neurons having data samples. As a result, to search boundary samples of data set, only two times of firing are needed. All the neurons without data points fire and emit pulses at time t_0. Those neurons with boundary samples will fire at time t_1, because they recept enough linking inputs from their surrounding neurons fired at time t_0. All the data samples stored in the neurons fired at t_1 are boundary samples.

The detail description of our method is given as follows:

1) Normalize the features of input vectors.

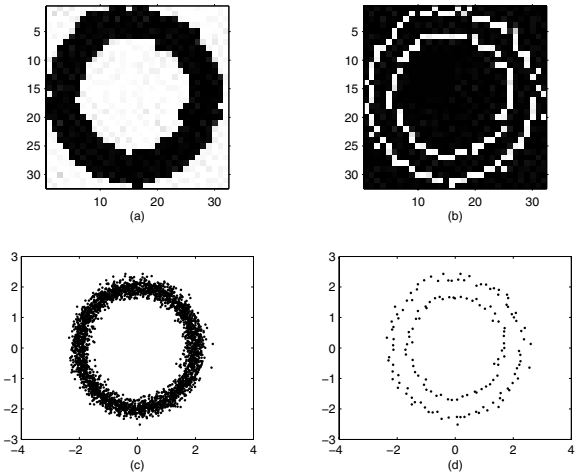

Fig. 3. Searching boundary samples using a 32×32 PCNNs for circle data set. (a). The first time firing map, the highlighted regions are neurons fired at time t_0. (b). The second time firing map, the highlighted regions are neurons fired at time t_1. (c). Original circle data set with 3142 data samples. (d). 129 Boundary samples stored in the neurons fired at time t_1.

2) Design a PCNN. Compute the range of feature values and determine the PCNN size: l and m , and then design a PCNN with $(l+2) \times (m+2)$ neurons.

3) Map data samples into the PCNN neurons, and store those data points closest to the neurons.

4) Initialize F_{ij}, L_{ij}, β and Θ_{ij}.

(i) Set Feeding inputs F_{ij}. The feeding inputs of neurons with data samples are set to 0, otherwise set to 1.

(ii) Set Linking inputs L_{ij} . All the linking inputs of neurons are set to 0 at time t_0.

(iii) Set linking weight β. The linking weight of each neuron is set to W

$$W = \begin{pmatrix} 0.5 & 1 & 0.5 \\ 1 & 1 & 1 \\ 0.5 & 1 & 0.5 \end{pmatrix}$$

(iv) Set threshold Θ_{ij}. All the thresholds are set to $\Theta(< 1)$ at time t_0.

5) The first time firing. At time t_0, the total internal activity U_{ij} of the neurons without data samples are more than their thresholds, hence these neurons will parallelized fire. After firing, their threshold values increase by a large constant V_Θ which prevent them from firing at time t_1.

6) The second time firing. At time t_1, the neurons with boundary samples will fire, because they can recept adequate linking inputs from their neighbors.

7) Find all the data samples stored in the neurons fired at time t_1. These samples are boundary samples.

8) end.

Fig.3 shows the process for searching data boundary of a circle data sets with 3142 data points. To search boundary samples for the data set, a 32×32 PCNN is designed. In (a), the highlighted regions are neurons fired at time t_0. In (b), the highlighted regions are neurons fired at time t_1, in which the boundary samples are stored. (c) is the original circle data set. In (d), the dots are boundary samples stored in the neurons fired at time t_1.

V. Experiments

In this section, some experiments will be carried out to test our method. Part A is a experiment for linearly separable problems, and part B is a nonlinearly separable experiment. Our experiments are carried on an Intel Pentium(R)4 2.66GHz with 512MB of memory and running Windows XP Professional 2002. In the two experiments presented here, LS-SVM is carried out using LS-SVM tools from LS-SVMlab [16] based on Matlab 6.5 system. The proposed searching boundary data samples algorithm is carried out based on matlab 6.5 program.

A. linearly separable problems

In this experiment, we use a linearly separable training set, called Syn1 data set, to test our method in classification application. The data set has two classes (1000 points for each class) in a two dimensional space (Fig.4), which is generated from Gaussian distributions. Linear kernel is employed for the LS-SVM classifier in the experiment. In Fig.4, the classification result is trained on the original 2000 data samples. Fig.5 shows the LS-SVM training result on the 57 boundary samples searched by our method.

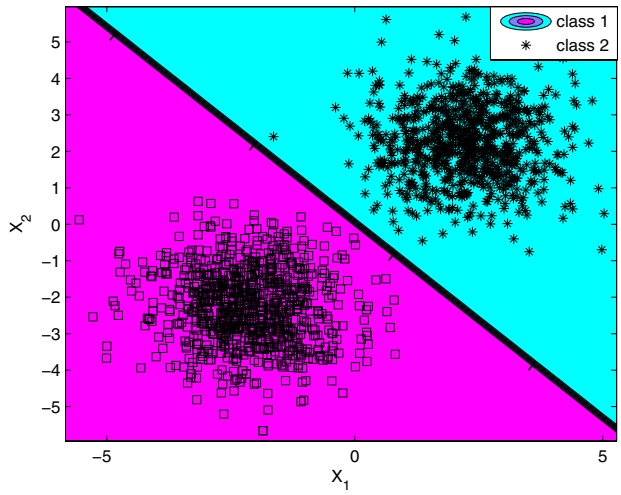

Fig. 4. The LS-SVM training result on the original Syn1 data set(2000 points).

B. nonlinearly separable problems

In this experiment, we use another nonlinearly separable training set, called fourclass data set [17], [18]. The data set has 862 data samples and two classes. RBF kernel function with $\gamma = 7$, $\sigma^2 = 0.3$ is employed for the LS-SVM classifier. In the Fig.6, the classification result is trained on the original data set. There are 144 boundary samples searched by our method. The training result on the boundary samples is shown in Fig.7.

From Fig.5 and Fig.7, one can see that the training results on the boundary samples are comparable to the results on original

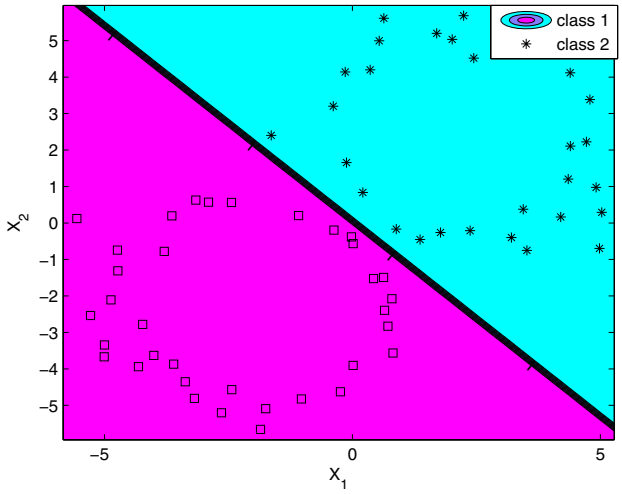

Fig. 5. The LS-SVM training result on 57 boundary samples of Syn1 data set.

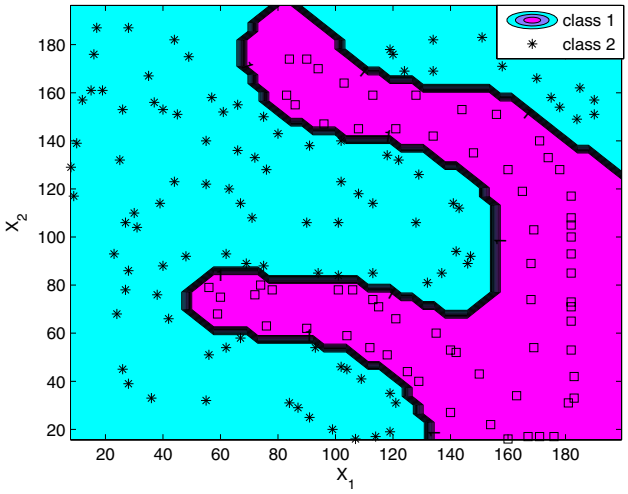

Fig. 7. The LS-SVM training result on 144 Boundary samples of the fourclass data set.

data sets. A simple comparison is given in Table I, which shows the proposed method can effectively reduce the number of support vectors and speed up LS-SVM. In the table, the training time of our method is the Sum of searching boundary samples time and training a sparse LS-SVM time. Obviously, our algorithm is computationally attractive. At the same time, the tests show that there is no accuracy loss by using our method.

TABLE I
COMPARISON OF REGULAR LS-SVM AND PROPOSED METHOD.

Data Sets	# of SVs		Training time (s)		Error rate (%)	
	LS-SVM	Proposed Method	LS-SVM	Proposed Method	LS-SVM	Proposed Method
Syn1	2000	57	0.946	0.078	0.15	0.1
fourclass	862	144	8.54	0.326	0	0

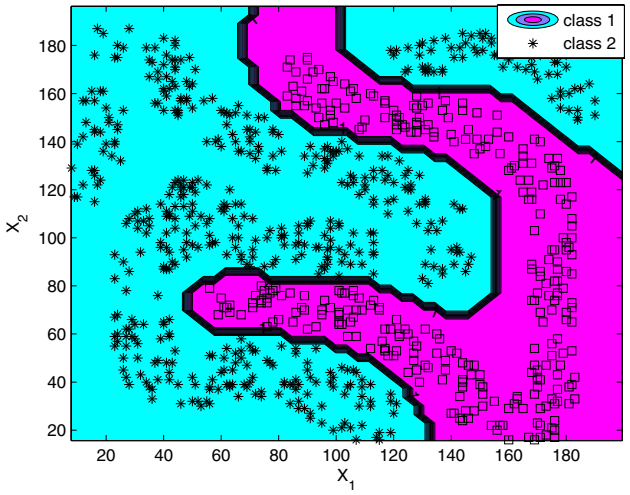

Fig. 6. The LS-SVM training result on the original fourclass data set(862 points).

VI. CONCLUSIONS

An effective sparse method for LS-SVM is proposed by using a modified PCNN model to search boundary samples of original data set. The boundary samples can be regarded as candidates of support vectors which include the most significant vectors for LS-SVM classifiers. Experiments for linearly separable and nonlinearly separable cases are given. These experiments show support vectors can be reduced obviously by proposed method without loss of performance. The proposed model can be developed for three dimensional data sets. For high dimensional problems, we can use PCA (Primary Component Analysis) for dimension reduction, and then use our PCNN model to search boundary samples based on their primary components.

ACKNOWLEDGMENT

This work was supported by Chinese 863 High-Tech Program under Grant 2007AA01Z321.

REFERENCES

[1] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag, 1995.
[2] B. Schölkopf, C. Burges, and A. J. Smola, *Advances in Kernel Methods: Support Vector Learning*, Cambridge, MA: MIT Press, 1999.
[3] V. Vapnik, *The support vector method of function estimation, in J.A.K. Suykens and J. Vandewalle (Eds) Nonlinear Modeling: Advanced Black-Box Techniques*. Boston, USA: Kluwer Academic Publishers, pp. 55-85, 1998.
[4] E. Osuna, R. Freund, and F. Girosi. "An improved training algorithm for support vector machines", in *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*,Amelia Island, FL, USA. pp. 276-285, 1997.
[5] J. C. Platt. "Fast training of support vector machines using sequential minimal optimization", in *B. Scholkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods C Support Vector Learning*, pp. 185-208. MIT Press, Cambridge, MA, 1999.

[6] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers", *Neural Process. Lett.*, Vol.9, No.3, pp. 293-300, June 1999.

[7] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse Least Squares Support Vector Machine Classifiers", in *Proc. of he European Symposium on Artificial Neural Networks (ESANN2000)*, Bruges, Belgium, pp. 37-42, 2000.

[8] B. J. de Kruif and T. J. de Vries, "Pruning error minimization in least squares support vector machines", *IEEE Trans. Neural Netw*., vol. 14, no. 3, pp. 696-702, May 2003.

[9] X. Y. Zeng and X. W. Chen, "SMO-based pruning methods for sparse least squares support vector machines, *IEEE Trans. Neural Netw*., vol.16, no.6, pp. 1541-1546, Nov. 2005.

[10] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor, "A comparison of pruning algorithms for sparse least squares support vector machines", in *Proc. 11th Int. Conf. ICONIP*, Calcutta, India, Nov. 22-25, 2004.

[11] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines", in *Proceeding of the First SIAM International Conference on Data Mining*, 2001.

[12] K. Lin and C. Lin, "A study on reduced support vector machines", in *IEEE Trans.Neural Networks*., vol. 14, pp. 1449-1459, 2003.

[13] R. Eckhorn, H. J. Reitboeck, M. Arndt *et al*. "Feature Linking via Synchronisation among Distributed Assemblies: Simulations of Results from Cat Cortex", Neural Comput., vol.2, pp. 293-307, 1990.

[14] T. Lindblad, J. Kinser. "Image Processing using Pulse-Coupled Neural Network", Springer-Verlag, 1998.

[15] John L. Johnson, Mary Lou PADGETT. "PCNN Models and Applications",*IEEE Trans. Neural Netw*., vol. 10, no. 3, pp. 480-498, 1999.

[16] LS-SVMlab, http://www.esat.kuleuven.ac.be/sista/lssvmlab/.

[17] FOURCLASS data set,http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/.

[18] Tin Kam Ho and Eugene M. Kleinberg, "Building projectable classifiers of arbitrary complexity", in *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, Austria, pp. 880-885, 1996.