A Confident Majority Voting Strategy for Parallel and Modular Support Vector Machines

Yi-Min Wen[1,2] and Bao-Liang Lu[1,⋆]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai 200240, China
{wenyimin; bllu}@sjtu.edu.cn
[2] Hunan Industry Polytechnic, Changsha 410208, China

Abstract. Support vector machines (SVMs) have been accepted as a fashionable method in machine learning community, but they cannot be easily scaled to handle large scale problems because their time and space complexities are around quadratic in the number of training samples. To overcome this drawback of conventional SVMs, we propose a new *confident* majority voting (CMV) strategy for SVMs in this paper. We call the SVMs using the CMV strategy CMV-SVMs. In CMV-SVMs, a large-scale problem is divided into many smaller and simpler sub-problems in training phase and some confident component classifiers are chosen to vote for the final outcome in test phase. We compare CMV-SVMs with the standard SVMs and parallel SVMs using majority voting (MV-SVMs) on several benchmark problems. The experiments show that the proposed method can significantly reduce the overall time consumed in both training and test. More importantly, it can produce classification accuracy, which is almost the same as that of standard SVMs and better than that of MV-SVMs.

1 Introduction

In recent years, there are many very large-scale data sets like public-health data, gene expression data, national economics data, and geographic information data. Using these very large data sets, researchers can get higher accuracy, discover infrequent special cases, and avoid over-fitting. However, most of existing machine learning methods are hard to be used to deal with these very large data sets because a very long training time and huge space are required. Therefore, one of the most challenging problems in machine learning community is to develop new learning model to efficiently handle these large data sets.

Today, support vector machine (SVM) [1] has been widely used in the field of pattern recognition for its strong theoretical foundations and good generalization

⋆ To whome correspondence should be addressed. This work was supported in part by the National Natural Science Foundation of China under the grants NSFC 60375022 and NSFC 60473040, and the Microsoft Laboratory for Intelligent Computing and Intelligent Systems of Shanghai Jiao Tong University.

Table 1. The contingency table

	label $y = 0$	label $y = 1$
prediction $h(x) = 0$	Tp	Fp
prediction $h(x) = 1$	Fn	Tn

performance. However, both its training time complexity and space complexity are $O(N^2)$, where N denotes training set size. The reason is that training SVMs is to solve a quadratic programme problem in essence. Many efforts are made to scale SVMs, such as choosing representative samples by preprocessing training data [2] [3] [4] [5], avoiding to solve the quadratic programme problem [6] [7] [8], and using geometric algorithms [9] [10].

The divide-and-conquer principle has been applied to scale SVMs. The SVMs using the divide-and-conquer principle in a serial way include the standard SVMs training method SMO [11], SVM^{light}, and libSVM, as well as using boosting to scale SVMs [12]. The SVMs using the divide-and-conquer principle in a parallel way, which will be named as parallel SVMs later on, include support vector mixtures [13], bayesian committee support vector machine (BC-SVM) [14], min-max modular SVMs (M^3-SVM) [15], and parallel mixture of SVMs [16]. Between sequential and parallel implementation, there are hierarchical and parallel methods [17], [18], [19], which filter non-support vectors in a cascade way.

From the point of view of parallel learning, parallel SVMs have many merits over monolithic SVMs. The first is that parallel SVMs can be benefited from cheap clustering systems by MPI, PVM, and the current grid computing [20]. The second is their reliability that parallel SVMs will still work even though some of their components fail. The third is their speedup, which can bring convenience to parameter selection.

In this paper, a confident majority voting (CMV) strategy is proposed to scale SVMs, which is inspired by an ensemble learning approach [21]. We call the SVMs using the CMV strategy CMV-SVMs. In CMV-SVMs, a large-scale task is divided into many smaller and simpler sub-problems in training phase and some confident component classifiers are chosen to vote for the final outcome in test phase. The experiments show that the proposed method can significantly reduce the overall time consumed in both training and test. More importantly, it produces classification accuracy which is almost the same as that of standard SVMs and better than that of MV-SVMs.

This paper is organized as follows. Section 2 introduces the model of CMV-SVMs, the definition of classification confidence, and the training and test algorithms for CMV-SVMs. In section 3, some experiments and analysis are presented for giving evidence of the advantages of CMV-SVMs. In section 4, the bias-variance decomposition strategy is employed to explore the reason why CMV-SVMs generalize better than MV-SVMs do. Finally, Section 5 is conclusions.

2 Confident Majority Voting

2.1 Definition of Classification Confidence

Given a problem with a training data set $S_{tr} = \{(x_1, y_1), ..., (x_N, y_N)\}$, where $x_i \in \mathcal{X} \subseteq \mathcal{R}^n$ is an instance, $y_i \in \{0, 1\}$ is its class label, and N denotes the training data set size. After training, a classifier $h : \mathcal{X} \rightarrow \{0, 1\}$ will be obtained. Given a test sample x, $h(x)$ will output the class label of x. In order to evaluate the performance of $h(x)$, a contingency matrix is defined as in Table 1.

In real-world applications, the class label y of a test sample x is not known. A good classifier should output a class label for x with high classification confidence. Otherwise its output cannot be believed and used to handle real-world problems. For example, in the field of medical diagnostics, the classification confidence is very paramount. In this paper, the classification confidence for a test sample x is used to choose the classifiers which can vote for the final classification. The classification confidence for x that is classified as class ω is defined as follows:

$$T(x) = P(y = \omega | h(x) = \omega) = \frac{P(h(x) = \omega | y = \omega) * P(y = \omega)}{P(h(x) = \omega)}, \quad (1)$$

where $T(x)$ denotes the classification confidence for x.

Many work has been made to compute classification confidence [22]. As in Proposition 1, after setting an appropriate neighbor size for a test sample x in a validation data set, the performance of a classifier in the neighborhood of a test sample x is used to evaluate the classification confidence of x. In addition, it should be noted that the classification confidence has been defined as local class accuracy in the work of Woods [21].

Proposition 1. *Subscribing the size of neighborhood of a test sample x in a validation data set. The performance of a classifier in the neighborhood of x is evaluated according to Table. 1. If x is classified as class 0, then its classification confidence can be computed as: $T(x) = \frac{Tp}{Tp+Fp}$. If x is classified as class 1, then its classification confidence can be computed as: $T(x) = \frac{Tn}{Tn+Fn}$.*

2.2 Training and Test Algorithms

CMV-SVMs can be regarded as a parallel implementation of the divide-and-conquer principle and a mixture of ensemble and modular learning.

The training algorithm for CMV-SVMs can be described as follows:

1. Initiation: constant M, i.e., the number of the repeat of training data set partitioning, the value of partition K, and the appropriate parameters for SVMs.
2. For $n = 1, 2, ...M$
   Partitioning: the training data set S_{tr} is randomly partitioned into K subsets with almost the same size, i.e. $\cup_{j=(n-1)*K+1}^{j=n*K} S_{tr}^j = S_{tr}$ and $\cap_{j=(n-1)*K+1}^{j=n*K} S_{tr}^j = \Phi$, where Φ denotes an empty set. The aim of making equal sizes of subsets is intended to keep load balance, although the training time of SVMs does not only depend on the training data size.

3. Training: $M * K$ support vector machines as component classifiers, $h_j, 1 \leq j \leq M * K$, are trained on the corresponding subsets $S_{tr}^j, 1 \leq j \leq M * K$. Because no communication is required in the training phase among the component classifiers, they can be trained in a parallel way.
4. Validating: Use training data set S_{tr} as a validation set to evaluate each component classifier $h_j, 1 \leq j \leq M * K$ and save the examination results. The validation results are used to evaluate the classification confidence for a test sample.

The test algorithm for CMV-SVMs can be described as follows:

1. Initiation: given a appropriate neighborhood size q, a classification confidence threshold $\varepsilon, 0 \leq \varepsilon \leq 1$, and a test sample x.
2. Classifying: Compute $h_j(x), 1 \leq j \leq M * K$ in parallel. If all the $h_j(x), 1 \leq j \leq M * K$ are the same, any $h_j(x)$ can be used as the final class label of x, then return. If not, goto next step.
3. Calculate all the classification confidence of x, i.e. $T_j(x), 1 \leq j \leq M * K$. Find the largest classification confidence: $imax = argmax_{j=1}^{M*K} T_j(x)$.
4. Find the set $\nabla = \{j | (T_{imax}(x) - T_j(x) \leq \varepsilon, 1 \leq j \leq M * K\}$.
5. Confident combining: If $|\nabla| = 1$, where $|\nabla|$ denotes the size of the set ∇, then use h_{imax} to classify, else choose classifiers $h_j, j \in \nabla$ to vote.

3 Experiments and Results

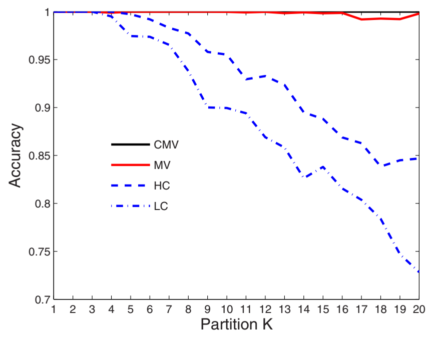
In order to evaluate the performance of the proposed method, some experiments are performed to compare our CMV-SVMs with standard SVMs and MV-SVMs. The experimental platform is PC with 1G RAM and 3G CPU. The training algorithm used is libSVM with cache of 40M and kernel function of RBF. Three data sets are used in the experiments, the first two are artificial data and the last is a real data set. The statistics of all the classification problems and the parameters used for SVMs are shown in Table. 2.

The artificial data sets include two-spirals data and checkboard data. The data of the two-spirals are uniformly chosen from two curves of $\rho = \theta$ and $\rho = -\theta$, where $(\rho\, \theta)$ means polar coordinates. The data of checkboard problem are chosen from a 2D checkboard that divides a 200×200 square into four quadrants in which the points are uniformly distributed [18]. Forest coverType data set comes from UCI [23], and only the samples of its second and sixth classes are chosen, in which one half of the data are used as test data and the rest data are used for training.

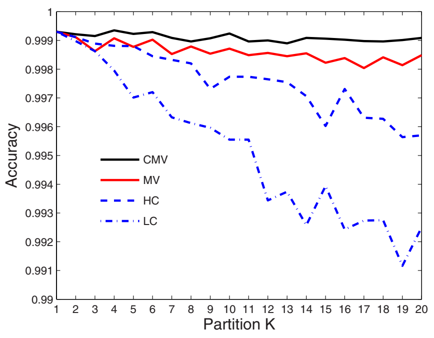
In order to get reliable experimental results, 100 training sets and a common test set are randomly generated for the two-spirals and the checkboard probelms, respectively. For the Forest coverType classification task, 100 training sets are randomly generated and each of them contains two-thirds of the whole training data. As a result, the experiments are performed in 100 times and the average results are presented. In order to systematically evaluate the proposed method,

Table 2. Problem description and the parameters used in SVMs

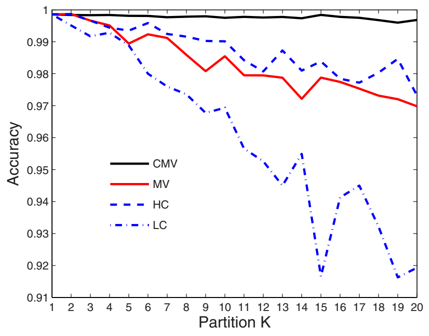
Problems	#attributes	#training data	#test data	c	σ	neighbor size q
Two spirals	2	3000	20000	128	2	5
Checkboard	2	32000	80000	1000	31.62	90
Forest coverType	54	28132	28132	128	0.25	90



(a)



(b)



(c)

Fig. 1. Classification accuracy comparison with $M = 1$. (a) Two-spirals, (b) Checkboard, and (c) Forest coverType. Here HC means the component SVM classifier with the highest classification accuracy, and LC means the component SVM classifier with the lowest classification accuracy

the value of K is set to 2, 3, ..., 20 in the experiments. $K = 1$ means that the classifier is trained by the entire training data, i.e. standard SVM is used.

From Fig. 1, we can see firstly that the generalization ability of CMV-SVMs is almost the same as standard SVMs in case of different partitions and sometimes better than standard SVMs. Secondly the generalization accuracy of CMV-SVMs is higher than all its component SVM classifiers. This demonstrates that the confident combining can efficiently make the component SVM classifiers work cooperatively. In addition, considering Table. 3, it seems that CMV-SVMs can

Table 3. Comparing the accuracy of CMV-SVMs with the accuracy of Knn

Problems	Knn		CMV-SVMs		
	neighbor size	accuracy	$K = 1$	$K = 2, 3, 4, ..., 20$	
				mean	variance
Twospirals	5	1.000	1.000	1.000	0.00007
Checkboard	90	0.995	0.999	0.999	0.00012
Forest coverType	90	0.989	0.999	0.998	0.00061

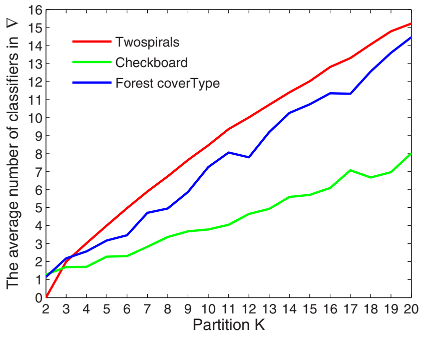


Fig. 2. The average number of confident SVMs for one test instance

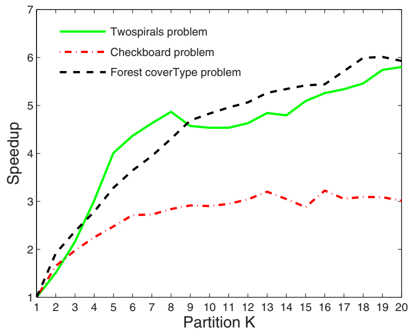


Fig. 3. The speedup, here the CPU time includes both training and test time

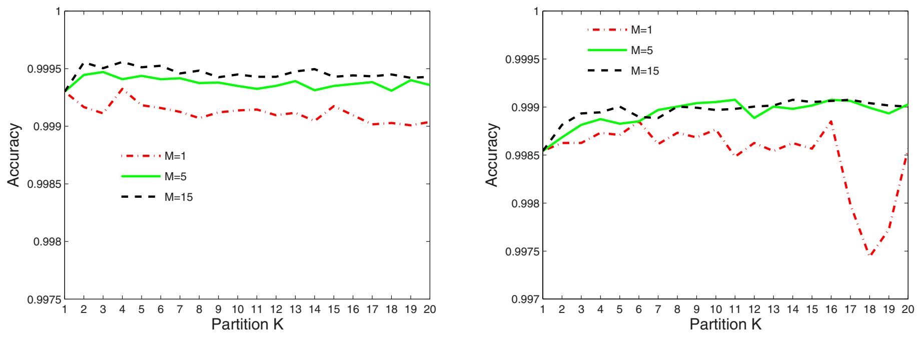


Fig. 4. Large values of M can improve classification accuracy. The left is on checkboard data set, and the right is on Forest coverType data set.

get higher accuracy than k-NN does. Therefore, k-NN cannot substitute CMV-SVMs even CMV-SVMs use the information of the nearest neighbor of a test instance. Thirdly the generalization accuracy of CMV-SVMs is higher than that of MV-SVMs.

Fig. 2 illustrates the average number of classifiers for one test sample when all $h_j(x), 1 \leq j \leq M*K$ are not the same. From Fig. 2, we can see that classification

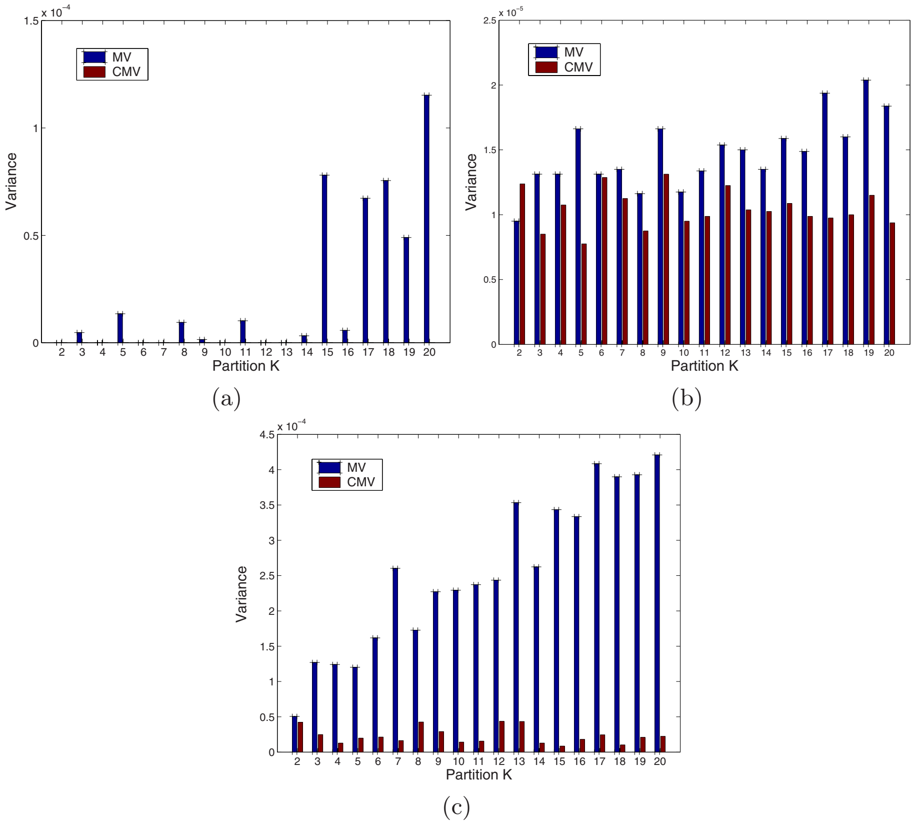


Fig. 5. Comparison of the variance of MV and CMV in case of different partitions: (a) Two-spirals, (b) Checkboard, and (c) Forest coverType

confidence requirement filters some component SVMs. It is like that for a given question only experts with richer experience can be selected to take part in decision-making. Therefore, the confident combining strategy can improve the generalization accuracy. The deeper reason is explored again by bias-variance decomposition [24] strategy in the next section.

In Fig. 3, the CPU time considered includes both training and test time. It can be seen that CMV-SVMs can significantly reduce the overall time. Fig. 4 shows that the larger the value of M the higher the accuracy. It seems because the larger M will lead to more diverse SVM classifiers and so more confident classifiers can be found to combine for classifying a test sample.

4 Bias-Variance Decomposition

Zhou *et al*. proposed an approach GASEN, which selects some neural networks based on the evolved weights to make up the ensemble, to show many could be better than all in neural networks ensembling [25]. By the bias-variance decomposition, it was explored that GASEN can significantly reduce both the variance and the bias simultaneously. Liking GASEN, CMV-SVMs selects some confident classifiers to make up the ensemble. In order to further explore the reason why CMV-SVMs generalize better than MV-SVMs do, the bias-variance decomposition is also employed in this paper.

4.1 Bias and Variance

Bias-variance analysis provides a powerful tool to study learning algorithms. By it, one can get insight into the error production of an algorithm and find the ways to improve the algorithms. According to Dietterich [26], the statistics bias of a learning algorithm is the persistent or systematic error that the learning algorithm is expected to make when trained on training sets of size N. Given a set S of training examples, algorithm A outputs a hypothesis $A(S) = \hat{h}_S$. It is convenient to define $\hat{p}_S(x)$ to be the probability that $\hat{h}_S$ misclassifies test point x. This probability is 1 if $\hat{h}_S$ misclassifies x, and 0 otherwise.

$$\hat{p}_S(x) = \begin{cases} 1, & \text{if } \hat{h}_S(x) \neq y, \\ 0, & \text{if } \hat{h}_S(x) = y. \end{cases} \tag{2}$$

Based on the above definition, given a sequence of training sets $S_1, S_2, ..., S_l$, each of size N, and a common test set S_{ts}, applying learning algorithm A to construct hypotheses $\hat{h}_{S_1}, \hat{h}_{S_2}, ..., \hat{h}_{S_l}$, the averaged probability of error can be defined to be the average of these $\hat{p}_S$'s, where the average is taken over all possible training sets:

$$\bar{\hat{p}}(A, N, x) = \lim_{l \to \infty} \frac{1}{l} \sum_{i=1}^{l} \hat{p}_{S_i}(x). \tag{3}$$

The expect error rate of A for a test point x is

$$E(A, N, x) = \bar{\hat{p}}(A, N, x). \tag{4}$$

The definition of Bias and Variance is like below:

$$B(A, N, x) = \begin{cases} 0, & \text{if } \bar{\hat{p}}(A, N, x) \leq 0.5, \\ 1, & \text{if } \bar{\hat{p}}(A, N, x) > 0.5. \end{cases} \tag{5}$$

$$V(A, N, x) = \begin{cases} \bar{\hat{p}}(A, N, x), & \text{if } \bar{\hat{p}}(A, N, x) \leq 0.5, \\ \bar{\hat{p}}(A, N, x) - 1, & \text{if } \bar{\hat{p}}(A, N, x) > 0.5. \end{cases} \tag{6}$$

So, the variance is the increase in the error rate at x relative to the bias.

4.2 Result Analysis

With the experimental methodology illustrated in Section 3, the bias and variance of CMV and MV are computed according to Dietterich's method. Fig. 5 shows that the variances of CMV are smaller than the variances of MV in case of different partitions in all the classification tasks, the only exception lies in the case of Checkboard data classification when $K = 2$. The biases of CMV and MV are zero in all cases and are not displayed. These evidences can explain why CMV-SVMs can generalize better than MV-SVMs do.

From Fig. 5, we can see that the variance of CMV keeps stable while the variance of MV gets bigger with increasing the number of partitions. These evidences can explain why MV-SVMs generalize worse and worse with increasing the number of partitions, while CMV-SVMs maintain their generalization accuracy.

5 Conclusion

In this paper, we have proposed a novel support vector machine called CMV-SVM to scale SVMs. Comparison with other parallel SVMs, CMV-SVMs are more easily to be implemented. Several experimental results indicate that the proposed confident majority voting strategy can get higher accuracy than majority voting does and the proposed CMV-SVMs can not only significantly reduce the overall time consumed in training and test, but also produces classification accuracy that is almost the same as standard SVMs do.

The limitation of the proposed CMV-SVMs lies in the necessity of storing all the training samples to evaluate the classification confidence for novel inputs. However, choosing the confident components can ensure better performance for modular learning system. The future work includes to modify the method of computing classification confidence and compare CMV-SVMs with other parallel SVMs on large-scale problems systematically.

References

1. Vapnik, V.N.: Statistical Learning Theory. Wiley Interscience (1998)
2. Boley, D., Cao, D.W.: Training Support Vector Machine using Adaptive Clustering. In: Proceedings of SDM'04. Lake Buena Vista, USA (2004)
3. Evgeniou, T., Pontil, M.: Support Vector Machines with Clustering for Training with Very Large Datasets. In: Lectures Notes in Artificial Intelligence **2308** (2002) 346-354
4. Yu, H., Yang, J., Han, J.W.: Classifying Large Data Sets using SVM with Hierarchical Cluster. In: SIGKDD'03. Washington, DC, USA (2003)
5. Pavlov, D., Chudova, D., Smyth, P.: Towards Scable Support Vector Machines using Squashing. In: Proceedings of the sixth ACM SIGDD conference on knowledge discovery and data mining (2000)
6. Fung, G., Mangasarian, O.L.: Proximal Support Vector Machine Classifiers. In: Proceedings of the seventh ACM SIGDD conference on knowledge discovery and data mining. San Francisco, CA (2001)

7. Mangasarian, O.L., Musicant, D.R.: Active Set Support Vector Machine Classification. In: Advances in Neural Information Processing Systems. MIT Press **13** (2001) 577-583
8. Mangasarian, O.L., Musicant, D.R.: Lagrangian Support Vector Machines. Journal of Machine Learning Research **1** (2001) 161-177
9. Tsing, I.W., Kwok, J.T., Cheung, P.M.: Core Vector Machines: Fast Svm Training on Very Large Data Sets. Journal of Machine Learning Rescarch **6** (2005) 363-392
10. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. IEEE Trans. Neural Networks **11** (2000) 124-136
11. Platt, J.: Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical Report MSR- TR-98-14, Microsoft Research (1998)
12. Pavlov, D., Mao, J.C., Dom, B.: Scaling-up Support Vector Machines using Boosting Algorithm. In: Proceedings of Internatinal Conference on Pattern Recognition (2000)
13. Kwok, J.T.: Support Vector Mixture for Classification and Regression Problems. In: Proceedings of the International Conference on Pattern Recognition. Brisbane, Queensland, Australia (1998) 255-258
14. Schwaighofer, A., Tresp, V.: The Bayesian Committee Support Vector Machine. In: Lectures notes in computer science **2130** (2001)
15. Lu, B.L., Wang, K.A., Utiyama, M., Isahara, H.: A Part-versus-part Method for Massively Parallel Training of Support Vector Machines. In: Proceedings of IJCNN'04. Budapest, Hungary (2004) 735-740
16. Collobert, R., Bengio, Y., Bengio, S.: A Parallel Mixture of Svms for very Large Scale Problems. In: Neural Information Processing Systems. MIT Press **17** (2004)
17. Graf, H.P., Cosatto, E., Bottou, L., Durdanovic, I., Vapnik, V.: Parallel Support Vector Machines: The Cascade Svm. In: Neural Information Processing Systems. MIT Press **17** (2005)
18. Wen, Y.M., Lu, B.L.: A Cascade Method for Reducing Training Time and the Number of Support Vectors. In: Lecture Notes in Computer Science **3173** (2004) 480-486
19. Wen, Y.M., Lu, B.L.: A Hierarchical and Parallel Method for Training Support Vector Machines. In: Lecture Notes in Computer Science **3496** (2005) 881-886
20. Foster, I.: The Grid: A New Infrasturcture for 21st Century Science. Physics Today **55** (2002) 42-47
21. Woods, K., Kegelmeyer, W.P.J, Bowyer, K.: Combination of Multiple Classifiers using Local Accuracy Estimates. IEEE Trans. Pattern Analysis and Machine Intelligence **19** (1997) 405-410
22. Zaragoza, H., Alché-Buc, F.: Confidence Measures for Neural Network Classifiers. In: IPMU'98 (1998)
23. Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases. Univ. California, Dept. Inform. Comput. Sci., Irvine, CA. [online] (1998) Available: *ftp://ftp.ics.uci.edu/pub/machine-learning-databases*
24. German, S., Bienenstock, E., Doursat, R.: Neural Networks and the Bias/Variance Dilemma. Neural Computation **4** (1992) 1-58
25. Zhou, Z.H., Wu, J.X., Tang,W.: Ensembling Neural Networks: Many Could be Better than All. Artificial Intelligence **137** (2002) 239-263
26. Dietterich, T.G. Kong, E.B.: Machine Learning Bias, Statistical Bias, and Statictical Variance of Decision Tree Algorithms. Technical report, Department of Computer Science, Oregon State University (1995)