A Genetic Algorithm-based Multi-class Support Vector Machine for Mongolian Character Recognition

O. BATSAIKHAN[1]
C.K.HO[2]
Y.P. SINGH[3]

, Faculty of Information Technology
Multimedia University
63100 Cyberjaya- Malaysia
[1]m1160021@mmu.edu.my
[2]ckho@mmu.edu.my
[3]ypsingh@mmu.edu.my

Abstract. This paper proposes a hybrid genetic algorithm and support vector machine (GA-SVM) approach to address the Mongolian character recognition problem. As the character recognition problem can be considered as a multi-class classification problem, we devise a DAG-SVM classifier. DAG-SVM uses the One-Against-One technique to combine multiple binary SVM classifiers. The GA is used to select the multi-class SVM model parameters. Empirical results demonstrate that the GA-SVM approach is able to achieve good accuracy rate.

Keywords: Support vector machines, genetic algorithms, classification.

(Received August 07, 2007 / Accepted December 17, 2007)

1 Introduction

Support vector machines (SVMs) are a set of supervised learning techniques that has gained enormous popularity in the field of pattern recognition and classification. SVMs produce complex decision rules (with respect to classification) by mapping a set of training points to a high-dimensional space in which they are separable. SVMs can potentially provide better classification performance compared to conventional techniques [3, 15]. A difficulty with applying SVM (as with other classifiers) is tuning the model parameters .To this end, existing schemes include the Bayesian evidence framework [8, 13] and the probably approximately correct (PAC) framework [12, 14]. This work deals with Mongolian character recognition, which in essence give rise to a multi-class classification task. As SVMs were originally developed for binary classification, an ensemble of SVMs is required for the multi-class problem. Thus, the problem of model parameter selection becomes much more complex.

In this paper, we introduce a hybrid GA-SVM classifier for Mongolian character recognition. We consider SVMs with Gaussian kernel. A standard GA is used for selecting good values for major SVM parameters, namely the kernel parameter (denoted by γ) and the regularization parameter (denoted by *C*). Such combination of GA and SVM has been proven to be successful in [17]. Since the character recognition problem is a multi-class classification problem, the use

of a single SVM would not suffice since an SVM in its original form is a binary classifier. Hence, multiple binary SVMs are assembled using the One-Against-One paradigm to form the multi-class SVM classifier. This shall be referred to as DAG-SVM as introduced in [10].

The remaining of this paper is organized as follows: In section 2, a brief introduction to binary SVMs and multi-class SVMs is given. Section 3 discusses the SVM model parameter selection problem, and details the proposed GA for multi-class SVM parameter selection. The experimental results and performance analysis is given the section 4. Section 5 concludes this paper.

2 Binary SVMs

In general, a binary classification problem deals with a set of examples $\{x_i, y_i\}$ with $1 \leq i \leq l$ (l being the size of the example set). Each example $x_i \in R^N$, N being the dimension of the input space, belongs to a class labeled by y_i. The goal is to search for a hyperplane that optimally separates the positive from the negative examples. The points x which lie on this hyperplane satisfy the equation $w \times x + b = 0$, where w is normal to the hyperplane. The SVM learning algorithm searches for the maximal separating hyperplane:

$$y_i(x_i \times w + b - 1) \geq 0 \qquad (1)$$

If Eq. 1 is satisfiable, the examples are linearly separable. The optimization problem for obtaining the maximal separating hyperplane can be formulated as:

$$\begin{array}{ll} Minimize & \frac{1}{2} w \times w \\ Subjectto & y_i(x_i \times w + b - 1) \geq 0 \end{array}$$

This is solved using the methods of Lagrange multipliers. Using Lagrange multipliers, it can be shown that the linear support vector training problem is reduced to [15]:

$$\begin{array}{ll} Maximize & L(\alpha) = \Sigma_{i=1}^{l}(\alpha_i) - \\ & \frac{1}{2}\Sigma_{i=1}^{l}(\alpha_i \alpha_j y_i y_j \mathbf{x_i x_j}) \\ Subjectto & \Sigma_{i=1}^{l}(\alpha_i y_i) = 0, \alpha_i \geq 0 \end{array} \qquad (2)$$

with α_i denoting a Langrange multiplier.

The optimal solutions $\overline{\alpha_i}, (\overline{w}, \overline{b})$ satisfy the following condition [15]:

$$\overline{\alpha_i}[y_i(x_i \times \overline{w} + \overline{b}) - 1] = 0, i = 1, ...l \qquad (3)$$

All $\overline{\alpha_i}$ take the value of zero except those for which the constraints (1) are satisfied with the equality sign. Since most of the $\overline{\alpha_i}$ are zero, the vector w is a linear combination of a relatively small percentage of the points x_i. These points are called support vectors, and they lie on the hyperplane. Having learned the decision boundary, a new example x can be classified using the following decision rule:

$$f(x) = sgn(\overline{w} \times x + \overline{b}) \qquad (4)$$

To allow for cases where the examples are non-separable, positive slack variables ζ_i,$i = 1...l$ and a regularization parameter C are introduced in the constraints. The intention of the slack variable is to allow for a small number of misclassified points. For the situation where $0 < \zeta_i < 1$, the example falls the correct side of the decision surface. For $\zeta_i > 1$, it falls on the wrong side of the decision surface. Thus, the summation of ζ_i gives an upper bound on the training errors. This can then be incorporated into the formulation of the objective function to become:

$$\begin{array}{ll} Minimize & \frac{1}{2} w \times w + C\Sigma_{i=1}^{l}\zeta_i \\ Subjectto & y_i(x_i \times w + b - 1) \geq 1 + \zeta_i \end{array} \qquad (5)$$

For non-linear SVMs, a mapping function $\Phi = R^N \rightarrow H$ is required to map the input features into a high-dimensional space. Then of course the training algorithm would only depend on the data through dot products in H, i.e. on functions of the form $\mathbf{\Phi(X_i)} \times \mathbf{\Phi(X_j)}$. Suppose there is a kernel function $\mathbf{K}$ that does the following:

$\mathbf{K(X_i, X_j)} = \mathbf{\Phi(X_i)} \times \mathbf{\Phi(X_j)}$

In this case, it is sufficient to use $\mathbf{K}$ in the training algorithm, without having to ascertain . Some commonly used kernels include Gaussian, Radial Basis Functions, Polynomials, and Sigmoidal. With the use of a kernel, the training algorithm then maximizes:

$$L(\alpha) = \Sigma_{i=1}^{l}(\alpha_i) - \frac{1}{2}\Sigma_{i=1}^{l}(\alpha_i \alpha_j y_i y_j \mathbf{K(x_i, x_j)}) \qquad (6)$$

and the decision function becomes

$$f(x) = sgn(\Sigma_{i=1}^{l}(\alpha_i y_i \mathbf{K(x_i, x_j)} + \mathbf{b})) \qquad (7)$$

3 Multi-class SVM and its Training

A number of architectures are available to enable SVMs to handle multi-class problems. One way is to process all examples in one optimization formulation. This is called the 'all-in-one' approach [15, 16]. Another techniques is to divide the multi-class problem into sub-problems, each of which can be solved by a binary SVM. The output of all the binary SVMs are then combined to form the final output for the multi-class problem. This technique of dividing the problem into sub-problems and then combining the smaller outputs is called 'divide-and-combine' [11]. The divide-and-combine' approach give rise to the need for local and global parameter tuning. In local tuning, each binary SVM is tuned with respect to the sub-problem it is solving. In global tuning,

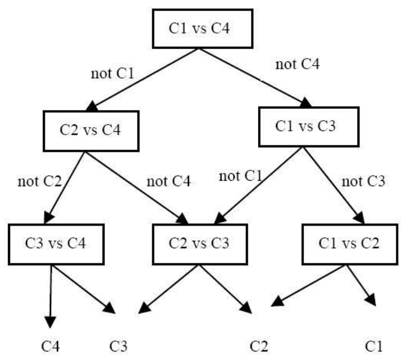

Figure 1: The decision DAG for a four-class problem.

we have two concerns. Firstly, the manner in which the binary SVMs are combined. Secondly, it is necessary to ensure that the parameters selected in local tuning can work reasonably well when the binary SVMs are combined. The fact that each binary SVM may use different kinds of kernels adds to the difficulty of parameter tuning [7].

A popular example of the divide-and-combine approach is DAG-SVM (DAG stands for Directed Acyclic Graph). It was introduced in [10], and has its origins from [4]. Given a classification problem with K classes, the DAG-SVM technique uses K(K-1)/2 binary SVM classifiers. Each of these classifiers is trained on a distinct pair of classes. DAG-SVM can be considered as a 'one-against-one' method. The decision DAG contains K(K-1)/2 SVMs (the internal nodes) and K number on leaf nodes, each representing a class. An example of a decision DAG for a four-class problem is shown in Figure 1.

To classify an example, we begin from the root and travel down the DAG until a leaf node is reached, which gives the final class. At each node, a binary decision function is evaluated. The edges are labeled with the outcome of the binary decision function.

3.1 SVM training

The purpose of training an SVM (as with any other classifiers) is to minimize its generalization error. A scheme called cross-validation is usually used to partition the example set into subsets for use in training and validation. In a *K*-fold cross-validation, the original examples are partitioned into *M* subsets, usually of equal sizes. *M-1* subsets are used for training the SVM, and one for validation. The training is done M number of times. With this arrangement, each subset is used exactly once as the validation set. The *M* results can then be averaged or combined in some other way to produce a single estimation. The leave-one-out cross-validation uses a single example for validation while the rest are used for training. Despite being computationally more expensive, it can give an almost unbiased estimation of the generalization error [2]. Cross-validation is used in this work as it is a well-established technique for supervised learning, and it is easy to implement.

Table 1: The different choice of the regularization parameters on single font and single size Mongolian characters for γ=0.015

Structure	Parameters	Error Learning set	rate Test set
DAG-SVM RBF	C=100 γ= 0.015	0.00%	2.0%
DAG-SVM RBF	C=10 γ= 0.015	0.00%	2.7%
DAG-SVM RBF	C=1 γ= 0.015	0.00%	2.5%
DAG-SVM RBF	C=0.1 γ= 0.015	0.01%	2.3%
DAG-SVM RBF	C=0.01 γ= 0.015	0.08%	1.2%
DAG-SVM RBF	C=0.001 γ= 0.015	0.00%	1.8%

3.2 Sensitivity of Multi-class SVM performance with respect to model parameters

This study employs Gaussian kernels for the SVMs. The parameters to be optimized for SVMs with Gaussian kernels are γ (the kernel parameter) and the regularization parameter *C*.

The experiments to study the sensitivity of the multi-class SVM with respect to *C* and γ consists of two parts. The first part holds γ at a constant value, with γ=0.015. The regularization parameter *C* is varied using the values 0.001, 0.01, 0.1, 1, 10, 100. For the multi-class classification, we have chosen the DAG-SVM classifier. The data set consists of 20 Mongolian characters (Single font and single size) with 100 samples for each character. Each character is represented as a 11×12 grid of binary pixels.

We trained the DAG-SVM classifiers with the Sequential Minimal Optimization (SMO) algorithm on the Arial Mongolian font of size 10. Table 1 presents the error rate of the DAG-SVM classifier with different choices of the regularization parameters *C* and γ=0.015. The DAG-SVM classifier is rather insensitive to different choices of the regularization parameters *C*. For the experimented values in the set 0.001, 0.01, 0.1, 1, 10, 100, the error rate on the test set is about the same (in the area of 1.2% - 2.7%).

The second part of the experiment attempts to discover if the performance of DAG-SVM is more sensitive to the kernel parameter γ. For this purpose, the regularization parameter *C* is set to 100 (constant), and γ takes its values from 0.001, 0.01, 0.1, 0.2, 0.3, 0.5. Table 2 summarizes the classification error rates of DAG-SVM with respect to the different choices of γ on the same data set. We conclude from Table 2 that the DAG-

Table 2: The different choice of γ on single font and single size Mongolian characters for *C* = 100.

Structure	Parameters	Error Learning set	rate Test set
DAG-SVM RBF	C=100 γ=0.001	0.00%	2.7%
DAG-SVM RBF	C=100 γ= 0.01	0.00%	3.3%
DAG-SVM RBF	C=100 γ= 0.1	0.00%	10.0%
DAG-SVM RBF	C=100 γ= 0.2	0.00%	63.7%
DAG-SVM RBF	C=100 γ= 0.3	0.00%	82.4%
DAG-SVM RBF	C=100 γ= 0.5	0.00%	94.2%

SVM classifier is very sensitive to different choices of the γ. For all values in 0.001, 0.01, 0.1, 0.2, 0.3, 0.5, the performance is very different (in the range of 2.7% to 94.2%).

It can be noted from Table 1 and Table 2 that the values of γ have much effect on accuracy rate than the regularization parameter *C* within the selected range. In the following we propose a genetic algorithms for selecting a good value for γ.

We conclude from Table 2 that the DAG-SVM classifier is very sensitive to different choices of the γ. For all values in 0.001, 0.01, 0.1, 0.2, 0.3, 0.5, the performance is very different (in the range of 2.7% to 94.2%).

It can be noted from Table 1 and Table 2 that the values of γ have much effect on accuracy rate than the regularization parameter *C* within the selected range. In the following we propose a genetic algorithms for selecting a good value for γ.

3.3 Genetic Algorithm based model parameter selection for multi-class SVM classifiers

Genetic algorithms (GAs) fall under a category of algorithms called evolutionary algorithms (EAs). Evolutionary algorithms were motivated and designed based on models of organic evolution. GA in the earliest form is developed by Holland, a computer scientist and psychologist at the University of Michigan. Holland's work is the starting point for nearly all known applications and implementations of genetic algorithms [5]. The main idea of GA is to search for a good solution from a population of candidate solutions. It represents the candidate solutions using strings of symbols called chromosomes. The collection of chromosomes makes up the GA population, which is basically a set of sample points from the search space [9].

The GA population is subjected to the following operations: selection, crossover and mutation. Selection produces a group of candidate solutions on which crossover is performed. Crossover aims to produce better solutions from the existing solutions. The next step is to perform mutation, which are small, random perturbations made to the newly produced candidate solutions. Crossover provides search intensification while mutation helps in diversification. Collectively, the crossover and mutation operations are known as the reproduction functions. The process of selection, crossover and mutation is repeated until a stopping criterion is fulfilled. The computational steps of a simple GA can be summarized as follows:

Generate the initial population P_0 of *N* individuals

1. $i \leftarrow 1$
2. $P_i^{'} \leftarrow$ *selection_function*$(P_i - 1)$
3. $P_i \leftarrow$ *reproduction_function*$(P_i^{'})$
4. $Evaluate(P_i)$
5. $i \leftarrow i + 1$
6. Repeat step 3 until termination.

We use a GA for selecting values for γ. Let γ_j represent the model parameter of interest for the jth binary SVM. We employ real-valued chromosomes. A chromosome consists of all the γ_j values, i.e. γ_1,γ_2,,γ_M, where *M* is the required number of binary SVMs. The decoding of the chromosome is then straight forward. We use the overall classification rate on a validation set as the fitness function.

The population consists of 30 chromosomes. The decision to use this population size is based on two factors. The first is to keep the computational cost low since each fitness function evaluation involves executing the GA. Secondly, a trial-and-error effort indicates that using 30 chromosomes can still produce acceptable results. The selection, crossover and mutation operations and their parameters are shown in Table 3. The arithmetic crossover operator is implemented as follows [6]:

$$\begin{aligned} \overline{X'} &= r\overline{X} + (1-r)\overline{Y} \\ \overline{Y'} &= (1-r)\overline{X} + r\overline{Y} \end{aligned} \tag{8}$$

where $\overline{X'}$ and $\overline{Y'}$ are the offspring from $\overline{X}$ and $\overline{Y}$, and *r* is a uniform random number in the range (0,1). The non-uniform mutation is implemented such that for any gene x_i that is selected for mutation, its new value $x_i^{'}$ is determined as follows [6]:

$$x_i^{'} = \begin{cases} \mathrm{x}_i + (b_i - x_i)f(G) & \text{if } \mathrm{r}_1 < 0.5 \\ \mathrm{x}_i + (x_i + a_i)f(G) & \text{if } \mathrm{r}_1 \geq 0.5 \\ \mathrm{x}_i, & \text{Otherwise} \end{cases} \tag{9}$$

where $f(G) = (r_2(1 - \frac{G}{G_{max}}))^b$,

Table 3: GAOT Parameters used for real-valued genetic representation

Operator Name	Operator Parameter
Non-Uniform Mutation	Frequency: 3
Arithmetic Crossover	Frequency: 3
Tournament Selection	Tournament size: 5

Table 4: 3-class data set validating rates from multi-class SVM model selection

C	γ					
	0.002	0.004	0.008	0.015	0.03	0.06
100	99.9	99.8	99.9	99.9	100	100
10	100	99.9	100	100	100	99.7
1	99.9	100	100	100	99.8	99.9
0.1	99.8	100	99.9	100	99.9	99.9
0.01	100	99.9	100	100	100	99.9
0.001	100	100	99.9	100	100	99.9

r_1, r_2= a uniform random number in the range (0,1), *G* = the current generation, G_{max} = the maximum number of generations. b = the shape parameter. The GA terminates after 100 generations.

4 Empirical Results

We apply the proposed GA-SVM approach on the Mongolian character recognition problem. We choose the following data sets: 3-class and 6-class. For all the problems, we choose 50 % of training samples as validating samples to select the proper model. We use the SVM Toolbox [1] and GA Toolbox called GAOT [6] for simulation. In the first step, we set the candidate values for γ to be 0.002, 0.004, 0.008, 0.015, 0.03, 0.06. The candidate values for the regularization parameter *C* are 0.001, 0.01, 0.1, 1, 10, 100. We arrived at these candidate values by first estimating the lower bounds using a number of trials. For γ, the lower bound is 0.002. The next candidate value is about twice the lower bound. For *C*, the lower bound is 0.001. The next candidate value is ten times the lower bound. Each subsequent candidate value is determined in the same way. The validation rate of the multi-class SVM is evaluated on the 36 combinations of *C* and γ. Two outcomes are expected from this simple parameter sweep effort.. Firstly, this will identify a very good value for *C* from the candidate values. Secondly, we can estimate a smaller range within which the near-optimal value of γ can be found. The GA is then used to search within this range to obtain the best value for γ.

4.1 Results on the 3-class data set

Table 4 presents the validation rates for the 36 combinations of *C* and γ .

Table 5: Confusion matrix on 3-class problem with GA selected γ.

	Actual Class Label		
	1	2	3
1	99.9%	0.0%	0.1%
2	0.0%	100.0%	0.0%
3	0.0%	0.0%	100.0 %
Average rate	99.97%		

Table 6: 6-class data set validating rates from multi-class SVM model selection

C	γ					
	0.002	0.004	0.008	0.015	0.03	0.06
100	99.3	99.1	99.8	99.6	99.2	99.1
10	99.3	99.7	99.2	99.7	99.7	99.5
1	99.3	99.5	98.9	99.5	99.6	99.8
0.1	99.5	99.6	99.8	99.6	99.7	99.4
0.01	99.3	99.7	99.7	99.9	99.3	99.6
0.001	99.7	99.6	99.6	99.6	99.7	99.1

The best validation rate is 100.0% using the values *C*= 10, 1, 0.1, 0.01, γ= 0.015. We select 99.9% as our selection threshold. Thus, the optimal models are most likely to exist in γ $\in$[0.04, 0.015] with the regularization parameter *C* at values 10, 1, 0.1, 0.01, 0.001. Empirical results show that the values of the regularization parameter *C* have much less effect on accuracy rate than γ within the selected range. Thus, in our GA based search, the regularization parameter *C* is set to 10. GA will then search for the best value for γ within the range [0.04, 0.015].

The overall validating accuracy rate serves as our fitness value. All individual SVMs have the same value for *C*. The best value for γ determined by GA is 0.0089. We then use the selected parameters to train the multi-class SVM. The trained multi-class SVM is then used to perform prediction on the testing samples. The testing confusion matrix is shown in Table 5. It can be observed that the multi-class SVM performed very well, with an average rate of 99.97%.

4.2 Results on the 6-class data set

As before, we first perform a rough search for a good value of the regularization parameter *C* from the values 0.001, 0.01, 0.1, 1, 10, 100. This rough search is also used to identify a small interval in which the best value for γ is located. The initial values for γ are 0.002, 0.004, 0.008, 0.015, 0.03, 0.06. Table 6 presents the validating rates from the above *C* and γ combinations (36 combinations). The best validating rate is 99.9% with the regularization parameter *C* = 0.01. Next, for the GA optimization, we set γ to be within the range [0.015, 0.06]. The regularization parameter *C* is held at 0.01.

The GA converges very fast, usually around 20 gen-

Table 7: Confusion matrix on the 6-class problem with GA selected

	Actual Class Label					
	1	2	3	4	5	6
1	99.9%	0.0%	0.0%	0.0%	0.0%	0.1%
2	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%
3	0.0%	0.0%	99.8%	0.0%	0.0%	0.2%
4	0.0%	0.0%	0.0%	100.0%	0.0%	0.0%
5	0.0%	0.0%	0.0%	0.0 %	100.0%	0.0%
6	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%
	Average rate 99.95%					

erations. The best testing rate we obtained is for γ=0.0513. The corresponding test results are shown in Table 7. An average accuracy rate of 99.95 is obtained, which can be considered as very good. The results on the 3-class and 6-class data set demonstrate the effectiveness of the multi-class SVM model parameter selection technique, which is a combination of a simple exhaustive search for the regularization parameter, and a GA search for the kernel parameter γ.

5 Conclusions

We have described a hybrid GA-SVM multi-class classifier with application to the Mongolian character recognition problem. Emphasis is placed on the selection of two important SVM parameters. The regularization parameter is selected using an exhaustive search procedure based on a set of candidate values. We have demonstrated that the classification performance of the multi-class SVM on the Mongolian character recognition problem is less sensitive to changes in the regularization parameter. This hybrid system harnesses the capability of a standard GA to search for the best value for the SVM Gaussian kernel parameter γ The multi-class SVM, using the optimized parameters, have shown very good classification results on the 3-class and 6-class data sets, with average accuracy of at least 99.9%.

References

[1] Cawley, C. G. MATLAB SVM toolbox (v0.54β). *University of East Anglia, School of Information Systems, Norwich, Norfolk, United Kingdom*, 2000.

[2] Chapelle, O. and Vapnik, V. Model selection for support vector machines,Advances Neural Information Processing Systems. *MIT press*, 12, 2000.

[3] Cortes, C. and Vapnik, V. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[4] Friedman, J. Another Approach to Polychotomous Classification. *Dept. Statist., Stanford University, Stanford,CA,http://www-stat.stanford.edu/reports/friedman/poly.ps.Z.*

[5] Holland, J. H. Adaptation in natural and artificial systems. *The University of Michigan Press*, 1975.

[6] Houck, C. R., Joines, J. A., and Kay, M. G. A Genetic Algorithm for Function Optimization: A Matlab Implementation. *North Carolina State University*, 1996.

[7] Hsu, C. W. and Lin, C. J. A Comparison of methods for multi-class support vector machines. *IEEE Transaction on Neural Networks*, 13:415–425, 2001.

[8] Kwok, J. Integrating the evidence framework and the support vector machine. *Proceedings European Symposium on Artificial Neural Networks, Brussels*, 1999.

[9] Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. *Springer Verlag, New York*, 1992.

[10] Platt, J. C., Cristianini, N., and Shawe-Taylor, J. Large margin DAGs for multi-class classification, Advances in Neural Information Processing Systems, Cambridge. *MA: MIT Press*, 12:547–553, 2000.

[11] Scholkopf, B., Burges, C., and Vapnik, V. Extracting support data for a given task. *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, 1995.

[12] Scholkopf, B., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. Kernel dependent support vector error bounds. *Proceedings of the International Conference on Artificial Neural Networks,IEEE press, New York, 1999.*, 1, 1999.

[13] Tipping, M. The relevance vector machine. *Advances in Neural Information Processing Systems,MIT press, Cambridge, MA*, 12:633–699, 2001.

[14] Tsuda, K. Optimal hyperplane classifier based on entropy number bound. *Proceedings of the International Conference on Artificial Neural Networks, IEEE press, New York,*, 1, 1999.

[15] Vapnik, V. Statistical Learning Theory. *New York:Wiley*, 1998.

[16] Weston, J. and Watkins, C. Support vector machines for multi-class pattern recognition. *Proceedings 7th European Symposium on Artificial Neural Networks*, 1999.

[17] Xu, P. and Chan, A. An efficient algorithm on multi-class support vector machine model selection. *Proceedings of the International Joint Conference on Neural Networks*, 4:3229–3232, 2003.