Multi-class Support Vector Machines Based on Arranged Decision Graphs and Particle Swarm Optimization for Model Selection

Javier Acevedo, Saturnino Maldonado,
Philip Siegmann, Sergio Lafuente, and Pedro Gil

University of Alcala, Teoría de la señal,
Alcala de Henares, Spain
javier.acevedo@uah.es
http://www2.uah.es/teose

Abstract. The use of support vector machines for multi-category problems is still an open field to research. Most of the published works use the one-against-rest strategy, but with a one-against-one approach results can be improved. To avoid testing with all the binary classifiers there are some methods like the Decision Directed Acyclic Graph based on a decision tree. In this work we propose an optimization method to improve the performance of the binary classifiers using Particle Swarm Optimization and an automatic method to build the graph that improves the average number of operations needed in the test phase. Results show a good behavior when both ideas are used.

1 Introduction

Support vector machines [1] (SVM) have been applied with a satisfactory level of success to many different two-class problems [2]. Based on the Statistical Learning Theory (SLT) SVM try to improve the statistical risk rather than the empirical risk. Due to this reason, SVM give a better performance than other learning machines when classifying unseen patterns.

The extension to the multi-category problems, where there are N different classes, does not present an easy solution and is still a field to research. In [3] it was exposed a mathematical formulation to extend the binary case to multi-category problems, but it has to deal with all the support vectors at the same time, resulting a complex classifier that does not provide high performance in many problems. Most of the published works make the extension to the multi-class case building N different classifiers in the so called one-against-rest approach. The usual method is to compare the outputs of the classifiers and to select the one with the highest value. However, in [4] it is remarked that the output of an SVM is not a calibrated value and should be not compared. The way to solve this problem is also proposed in the same paper, adding to the output a estimation of the probability of success.

Another binary based approach is the so called one-against-one approach, where $N(N-1)/2$ classifiers are built, each being classifier trained only on

B. Beliczynski et al. (Eds.): ICANNGA 2007, Part II, LNCS 4432, pp. 238–245, 2007.

two of the N classes. Although this approach can give better results than the one-against-rest case, this scheme has not been widely applied due to the fact that the number of classifiers increase exponentially with the number of classes. However, in most of real applications what it really matters is the time needed to compute the test phase, specially in some real time systems. With this approach, in the training phase, the classifiers obtained can be simpler than in the one-against-rest case. In [5] it was proposed the Max Wins algorithm, obtaining very good results, but it implies that in the test phase all the built classifiers should be used, with a high cost from a computational point of view. In [6] the proposal was to build a graph with the binary classifiers, in such a way that in the test phase it is only necessary to work with N classifiers. This method was called Decision Directed Acyclic Graph based on SVM (DAGSVM). In this work, we propose an automatic method to arrange the graph, resulting in less average time to test the samples.

On the other hand, one of the major problems when using binary classifiers is the choice of the kernels, the parameters associated to these kernels and the value of the regularizing parameter C. The right choice of these parameters, known as model selection, improves the performance of the binary classifiers in a considerable way. In the proposal of the multi-class method of this work, the success of the binary classifiers is basic to ensure the overall performance. In this paper we have applied Particle Swarm Optimization (PSO) [7] to find the optimal value of the parameters.

2 Building the Set of Binary Classifiers

2.1 Model Selection

One of the most difficult points in classification is to tune the parameters associated to the learning machine. In our case, when working with SVM, the choice of the kernel is going to have great influence in the success of the classifier. Most of the classification problems are not linearly separable and a kernel method has to be applied. The most popular kernel for non linear cases is the Radial Basis Function One (1).

$$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma \sum_{i=1}^{n} (x_i - y_i)^2} \quad . \tag{1}$$

When this kernel is used, in addition to the C parameter, the γ parameter has to be tuned. Instead of using common parameters for all the classifiers, as it has been proposed in the previous mentioned work, it seems more logical to find the best combination of parameters for each binary classifier. In Fig.(1) it can be appreciated how the estimation of the error varies with these two parameters.

Most of the published works fix a C parameter and search to find the best γ obtained. Then, having found the γ parameter, the best possible C parameter is calculated. However, in Fig.(2) it can be appreciated that, fixing the C parameter to low values in this case, lead us to select a wrong value of the γ parameter. So, it is necessary to take into account both parameters at the same time, searching

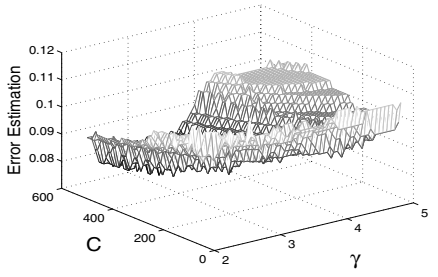

Fig. 1. An example of the Error with Different values for C and γ

for the combination of them that minimizes the estimation of the error. The direct method is to make the search space discrete and test all the possible combinations, but this procedure is very expensive from a computational point of view, specially when the one-against-one method is selected, due to the high number of classifiers to be used. So, the proposal is to use a statistical search method (SSM) to find an optimal value of the kernel parameters.

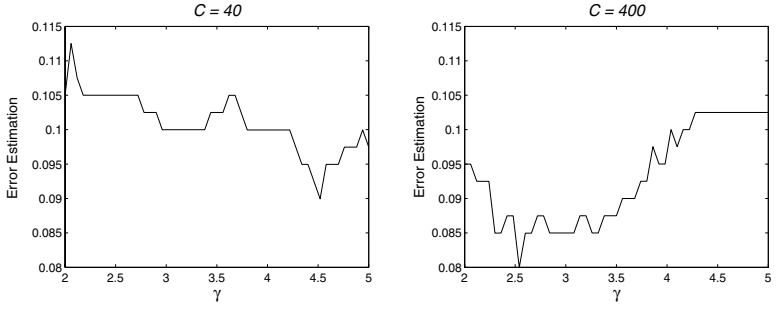

Fig. 2. An example of the Error Estimation with C fixed and γ variable

There are several estimators of the generalization error that can be used as evaluation function. The leave-one-out error is known to be an unbiased estimator, but it requires to train many SVM. The most extended estimator, as it has been used in this work, is the k-fold crossvalidation, that gives good results in a reasonable time.

2.2 PSO for Tuning SVM Parameters

Once that the importance of the parameter has been exposed, the question is how to select the appropriated values. It has to be noted that the functions described to estimate the error are not derivable and as it is shown in Fig.(1), there are multiple local minima. Moreover, there is not a priori information of the

error function until we train the dataset and the error is estimated. With these starting points, a method based on SSM for continuous function minimization seems to be appropriated to solve our problem.

PSO is a recent method for function minimization and it is inspired by the emergent motion of a flock of birds searching for food. Like in other SSM the search for the optimum is an iterative process that is based on random decisions taken by m particles searching the space at the same time. Each particle i has an initial position $\boldsymbol{x_i}$ that is a vector with a possible solution of the problem. In our case, the components of the vector are the C and γ parameters if the kernel is RBF. Each position is evaluated with the objective function and the particles update their position according to (2) where $v_i(t+1)$ is the new velocity of particle i, $\phi(t)$ is the intertia function, *pbest* is the best position achieved by the particle i, *gbest* is the best position achieved by any of the particles, $c_{1,2}$ are coefficients related to the strength of attraction to the *pbest* and *gbest* position respectively and $r_{1,2} \in [0,1]$ are random numbers.

$$\begin{aligned} v_i(t+1) &= \phi(t)\, v_i(t) + c_1 r_1 (pbest - x_i) + c_2 r_2 (gbest - x_i) \\ x_i(t+1) &= x_i(t) + v_i(t+1) \quad . \end{aligned} \tag{2}$$

The search of each particle is done using its past information and the neighborhood one. This fact makes that the particles fly to a minima position but they can scape if it is a local minima.

As it has been mentioned, the evaluation function measures the crossvalidation error. However, it has been observed that in some problems there are several combinations allowing the error to be minimized, usually in a plain region of the crossvalidation error. In such cases, the best choice is to select the solution that also minimizes the number of operations required as it is described in (3).

$$f(C,\gamma) = \widehat{Error}(C,\gamma) + \frac{\bar{N}_s}{l^2} \quad . \tag{3}$$

Where $\bar{N}_s$ is the average number of support vector obtained as a result of the crossvalidation partition and l is the total number of samples available for the training phase.

3 Graph Order

Once the classifiers have been trained, each of them with its optimal parameters, we can go back to the DAGSVM algorithm. In Fig.(3) it is shown the proposed order for a 4-classes problem in the DAGSVM method. After testing many datasets it can be said that the order of the graph is not relevant in the total accuracy of the classifier, but it plays a crucial role in the average operations needed in the test phase.

Let us assume that the classifier separating class 1 and class 4 is the one with a high number of operations needed to be evaluated in the test phase. The proposed graph of Fig.(3) makes that all the test samples have to be evaluated through this classifier, but in many cases this classifier is not relevant for the problem.

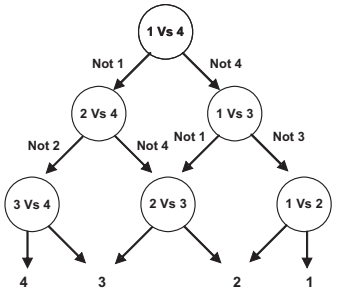

Fig. 3. Graph proposed for 4 classes in the DAGSVM method

If the classifiers have been trained as proposed in the described method to optimize the performance, it is clear that some of the classifiers are best candidates to be placed in the first nodes. Keeping in mind this idea the proposal is to design an automatic procedure to build the graph in any problem.

Given a problem with a set of training vectors $\boldsymbol{x_a} \in \mathrm{R}^n, a = 1, ..., l$ and a vector of labels $\boldsymbol{y} \in \mathrm{R}^l, y_i \in \{i = 1, 2, ...N\}$, and a set of classifiers $A_j, j \in \{1, 2, ...N(N-1)/2\}$ The basic algorithm proposed is summarized in the following steps:

1. Estimate the probabilities of each class C_i as:

$$P(C_i) = \frac{\sum_{h=1}^{l} u(y_h = i)}{l} \quad . \tag{4}$$

   Where $u(.)$ is the step function.
2. Calculate the number of operations associated to the class C_i as:

$$N_{op_i} = \sum_{j=1}^{N-1} \mathrm{adds}(A_{j,i}) + k_1 \mathrm{mult}(A_{j,i}) + k_2 \exp(A_{j,i}) \quad . \tag{5}$$

   Where k_1 and k_2 are two constants calculated as the time needed to compute a multiplication and a exponential function, taking as reference the time needed to calculate an addition.
3. Calculate the list L as following until all the classes are included:

$$L(i) = \arg\min_i (N_{op_i} P(C_i)) \quad . \tag{6}$$

4. Build the graph as shown in Fig.(4). The first classifier is the one that discriminates between $L(1)$ and $L(2)$. Then, the next layer is composed by two classifiers, separating the next element of L, in the example case class 1, and the previous classes. The process is repeated until the end of the list, building in each layer as many classifiers as the number of classes have that been added in the previous layers.

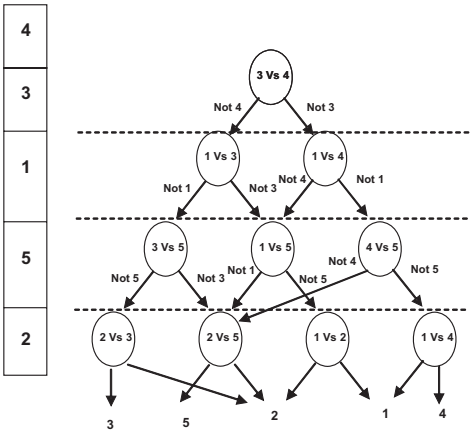

Fig. 4. Graph proposed for 4 classes with an ordered list

4 Results and Discussion

The proposed method, Graph Ordered SVM (GOSVM) was evaluated on different datasets obtained from the UCI [8] repository. The dataset named Cover Type was randomly reduced. We have compared these datasets using also the DAGSVM method, as described in [6] and the One-Against-Rest method coupling a probabilistic estimator (OARPSVM) as described in [4]. In order to see the performance of the optimization, the parameters of each binary classifier, the C and γ parameter were tuned using PSO only in the GOSVM method. DAGSVM and OARPSVM methods were trained with common C and γ parameters. The procedure to select these parameters was done in the classical way, that is , fixing in first place the C parameter and searching for the optimal γ and then searching the C parameter. In all cases, the adjustment of the parameters was done using the training set itself with a 5-crossvalidation error function. Selecting these parameters with an external test set could lead us to not generalize the problem. The results obtained are shown in Table(1). It can be appreciated how the proposed method, combining the PSO optimization for each classifier and the order of the graph give a slightly better accuracy in all cases and an important reduction in the number of operations needed in the test phase. For the Cover Type dataset the error obtained is very high, but this dataset is known to be a hard classification problem and worse results were obtained using other learning methods like neural networks. It is specially meaningful the reduction in the number of operations needed in this last case comparing to the DAGSVM and OARPSVM methods, while the accuracy achieved does not present an important improvement. This behavior can be explained due to the function (3) used in the optimization problem.

Once that both improvements have been tested, to adjust the parameters of each classifier and to order the graph, we have compared the GOSVM method

Table 1. Results for different datasets comparing the proposed method with the DAGSVM and one-against-all with probabilistic output

DATASET	N. of Classes	N. of Feat.	Samp. Train	Samp. Test	GOSVM		DAGSVM		OARPSVM	
					Error (%)	N^o Ops (10^9)	Error (%)	N^o Ops (10^9)	Error (%)	N^o Ops (10^9)
SEG. IMAGE	7	18	10000	36200	0.11	6.77	0.44	16.30	0.34	59.51
WAV	3	21	6000	94000	5.04	222.93	6.10	361.0	6.06	462.34
SATELLITE	6	36	40000	88700	0	373.64	0.674	498.86	1.008	1942.5
COVER TYPE	7	54	1120	50000	39.08	1.04	39.51	139.54	40.13	371.31

with the DAGSVM, but in this case the parameters of the binary classifiers have also been optimized for each one. Results are shown in Table(2). It can be appreciated that the reduced number of operations is caused not only by the order of the graph but also by the minimization in (3). However, the order of the graph, as it has been proposed can achieve better results in the operations needed, except in the Cover Type case. This behavior has a clear explanation since the arrangement is based on the estimation of the probability of each class. In the training set of this problem, all the classes have the same probability whereas in the test set some classes have much less probability than others. It can be said that the proposal method to order the graph does not have success when the probabilities of the training set are quite different than in the test set.

Table 2. Results for different datasets with the parameters adjusted for each binary classifier

DATASET	GOSVM		DAGSVM	
	Error (%)	N^o Ops (10^9)	Error (%)	N^o Ops (10^9)
SEG. IMAGE	0.11	6.77	0.12	12.54
WAV	5.04	222.93	5.02	271.87
LETTER	2.65	9.76	2.70	11.14
SATELLITE	0	373.64	0	425.32
COVER TYPE	39.08	1.04	38.97	0.96

5 Conclusion

In this work we have proposed two new ideas to optimize the behavior of a multi-class SVM in a one-against-one approach. Adjusting the value of the parameters each binary classifier improves the success of classification and in this work we have exposed a method to make this tuning without searching the whole space. The order of the nodes in the graph has not great influence in the success rate of classification, but it has an important effect on the average number of operations needed in the test phase.

Model selection is still an open field to research, and in future works some other functions optimizers will be tested searching also for other kernel types. There is also an open research line to test the method here exposed to some research areas, where the number of classes is very high.

Acknowledgments. This work was supported by Comunidad of Madrid project CAM-UAH 2005/031.

References

1. Vapnik, N.V.: The Natureof Statistical Learning Theory. Springer-Verlag, Berlin. (2000) 1ed: 1998.
2. Wang, L.: Support Vector Machines: Theory and Applications. Springer-Verlag, Berlin. (2005)
3. J. Weston and C.Watkins: Multi-class support vector machines. Technical report, Royal Holloway University of London (1998)
4. Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In Smola, A., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers. MIT Press (1999) 61–74
5. Kreßel, U.: Pairwise classification and support vector machines. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods – Support Vector Learning. MIT Press (1998) 225–268
6. Platt, J.: Large margin dags for multiclass classification. In Solla, S., Keen, T., Müller, K., eds.: Advances in Neural Information Processing Systems. MIT Press (2000) 547–553
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. IEEE Press, Piscataway, NJ (1995) 1942–1948
8. D.J. Newman, S. Hettich, C.B., Merz, C.: UCI repository of machine learning databases (1998)