SVMTOCP: A Binary Tree Base SVM Approach through Optimal Multi-class Binarization

Diego Arab Cohen[1] and Elmer Andrés Fernández[1,2]

[1] Biomedical Data Mining Group, Facultad de Ingeniería,
Universidad Católica de Córdoba, Argentina
[2] CONICET
{diego.arab,elmerfer}@gmail.com

Abstract. The tree architecture has been employed to solve multi-class problems based on SVM. It is an alternative to the well known OVO/OVA strategies. Most of the tree base SVM classifiers try to split the multi-class space, mostly, by some clustering like algorithms into several binary partitions. One of the main drawbacks of this is that the natural class structure is not taken into account. Also the same SVM parameterization is used for all classifiers. Here a preliminary and promising result of a multi-class space partition method that account for data base class structure and allow node's parameter specific solutions is presented. In each node the space is split into two class problem possibilities and the best SVM solution found. Preliminary results show that accuracy is improved, lesser information is required, each node reaches specific cost values and hard separable classes can easily be identified.

Keywords: multi-class classification, SVM, Binary Tree.

1 Introduction

Over the last few years one of the most used techniques to solve binary classification problems (K=2) has been Support Vector Machines (SVM). It has several advantages compared to other competitors. From a data mining point of view it provides much more information about the problem at hand such as class margin, class separability (if it is a hard or soft problem) and which data from the data base was used to build the solution. Its extension to multi-class problems (K >2) relies on transforming the multi-class problem space into several binary space problems, where each of these can now be solved by a single SVM. Different algorithms have been developed for this purpose such as One-against-All (OvA), One-against-One (OvO)[1], Direct Acyclic Graph SVM (DAGSVM)[2] and Binary Tree SVM (BTSVM)[3] . In OVA each class is faced to the rest of the classes resulting in building as much SVM classifiers as available classes. Classification of a new sample is defined according to the maximum output over all SVM classifiers. In OvO and DAGSVM all the classes are contrasted to each other requiring K(K-1)/2 SVM classifiers and a new sample is labeled according to a voting schema. Binary Trees based SVM multi-class follows a quite different approach. Instead of a contrast in a pairwise or OvA fashion, it builds K-1 SVM classifiers and for each of them recursively splits the multi-class space into two "new classes"

L. Alvarez et al. (Eds.): CIARP 2012, LNCS 7441, pp. 472–478, 2012.

for each of the K-1 SVMs. A new sample is labeled according to the label of the reached leaf of the tree. Different methods were proposed to split the multi-class space. They are mainly based on some kind of clustering and re-assignment technique. One of them is named Multistage SVM (MSVM), introduced by Liu et al [4]. They proposed to split a multi-class space into a two class space through unsupervised Support Vector Clustering (SVC) [5] and then re-label the samples according to the SVC solution. However this approach requires adjusting SVC parameters to find only two clusters each time. In addition it does not guarantee that the input class structure will be preserved (samples for the same class could be re-labeled to a different class) and the resulted tree may not be balanced, requiring, in addition, a balancing step. A second one called Binary Tree SVM (BTS) [3], randomly choses a pair of classes and then builds SVM classifiers for these two and "re-classifies" the remaining samples according to this SVM. Then, re-classified samples feed the child nodes of the previous one, repeating the process until a leaf is reached. Although this approach does not require SVC, it does not solve the drawbacks of the MSVM. The third approach is also a re-assignation procedure, but instead of randomly choosing two classes from the original space to build the SVM classifier, it applies a kind of k-means algorithm with $k=2$, using as a starting point the "gravity centres" of the two most disjointed classes and then reassigning remaining samples to these two classes. However, In order to build "gravity centres" in a feature space it is required to know the mapping function and this is not always the case as in the Gaussian kernel SVMs and it does not take into account class structure as in previously mentioned strategies.

Here we propose a new approach to split a K>2 multi-class problem into two class problem for each node in the tree by looking for the input class combinations that produce the best SVM performance in a specific tree node. This implies to solve for node "i"

$$L_i = \eta \cdot \frac{K_i!}{r!(K_i - r)!} \tag{1}$$

binary problems, where η=1 for K odd and 0.5 for K even and r=[K/2]. Once the best solution if found for node "i" r classes are passed to the child nodes and the process repeated until reaching a leaf. Despite the training phase is time and computationally expensive, the proposed approach always produces a balanced tree and the original class structure is preserved. The last property is very important from a Data Mining point of view because the reached solution allows identifying which of the class combinations provides soft or hard margin solutions (tree nodes could have different kernel parameters) and automatically identifies what are the most difficult input classes to split. These are very important properties for data analysts who need to extract hidden knowledge from a multivariate data base.

2 Support Vector Machines

Here we briefly present the SVM theory. Support vector machines are learning machines with a very nice theoretical background, the Statistical Learning Theory, which provides insights through which can then be used to improve the performance in practical real applications [6]. In the classification context, SVM provides a way to find

Maximal Margin Hyperplane giving a specific configuration of the machine. In a general setting, the learning algorithm finds the optimum weight vector that solves the following problem: Given the set of samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ where $\mathbf{x_i} \in R^p$ is the input pattern for the ith example and $y_i \in \{-1,1\}$ the desired class response, the Hyperplane that separate the data should satisfy

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 + \xi \tag{2}$$

where the weight vector $\mathbf{w} \in R^p$ and the slack variable ξ allows the incorporation of some errors in the solution. The weight vector **w** can be obtained by the minimization of (primal form)

$$\Phi(\mathbf{w}, \xi) = 0.5 \cdot \|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \tag{3}$$

constrained to Eq.1. By the introduction of Lagrange multipliers the problem can be formulated in the dual form as

$$\Phi(\alpha) = \sum_{i=1}^{N} \alpha_i + 0.5\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{4}$$

Subject to the constrains

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{5}$$

$$0 \leq \alpha_i \leq C \text{ for } i=1,2,..,N.$$

Where C , the cost parameter, is a user specified positive value. In practice if all $\alpha_i < C$, the data set can be linearly separated without error (all $\xi_i = 0$) and the problem is said to be a "hard-margin" problem" if not, a "soft-margin" one. The bridge between primal and dual form is given by

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i \tag{6}$$

where SV is the set of input vectors for which $\alpha_i \neq 0$ (the support vectors). The squared norm of the weight vector can be calculated as

$$\|\mathbf{w}\|^2 = \sum_{i=1}^{N} \alpha_i - \frac{\|\alpha\|^2}{C} \tag{7}$$

and the maximum distance between the closest patterns in the hyper-geometrical space R^p is given by

$$2\gamma = \frac{2}{\|\mathbf{w}\|^2} \tag{8}$$

[7] In the case of a "hard margin" solution the second term of Eq. 7 vanishes.

3 SVM Tree Structure

The proposed method uses multiple SVMs arranged in a binary tree structure [3,7] (See Fig.1)). Each node in the tree will constitute the solution of a binary problem which represents a particular partition of the multiclass space. In order to find the best solution of a particular binary problem the best partition of the K classes should be found. At node "*i*" the class space is split in $L_i = c \cdot \frac{K_i!}{r!(K_i - r)!}$ disjoint partitions. Each partition is solved by an optimized SVM and the one with the best performance (i.e less error and less number of support vectors) is chosen to represent this "*i-th*" node in the tree. For instance, if we have four classes (*K=4*) *mc={A,B,C,D}* they are divided in $L_i = 0.5 \cdot \frac{4!}{2!\,2!} = 3$ binary problems, i.e {A,B}vs{C,D}, {A,C}vs{B,D} and {A,D}vs{B,C}. Then $L_i = 3$ SVMs trained in the root node, and from these binary problems (partitions) the one achieving the lowest error rate and the fewest amount of support vectors will represent the root node. Then a *K/2=2* class problem is passed to each of its child nodes. If *K=5*, ten (10) SVMs are required in the root node, then 3 classes will be passed to the left child node and 2 classes to the right child (or vice-versa).

For each of the $\boldsymbol{L}_i$ SVM, the best SVM cost C (eq.3) is found. In order to avoid expensive cross validation procedures, we chose the SVMpath algorithm [8]. This algorithm permits setting the best cost for Linear and Gaussian kernels without cross validation. The proposed algorithm was implemented in R (www.r-project.org) using svmpath and e1071 libraries. The last one is based on the well known LibSVM C++ library [9].

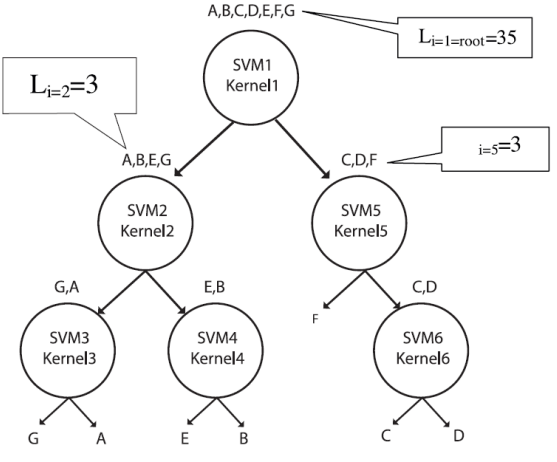

Fig. 1. A seven class problem example

4 Training and Validation

In this work, even though all the SVM parameters could be optimized, the unique tuned SVM parameter was the cost C of eq.3. In this context the used kernel function was radial /Gaussian with $\delta=1$ in all cases. For SVM OVO, the "svm" and "tune.svm" functions were used to set the optimum C cost value (they are available in the e1071 R library). The C value was spanned into the [C_{min} , C_{max}] range. If the best achieved SVM implies a cost equal to C_{min} or C_{max}, the process was repeated expanding the C range until the best SVM satisfied $C_{min} < C < C_{max}$.

For the SVMTree algorithm proposed here, the svmpath function from the "svmpath" R library was used. This algorithm fits the entire path of SVM solutions for every value of the cost parameter, with essentially the same computational cost as fitting one SVM model [8].

In all cases a 5-fold cross validation over 80% of the data base was applied to evaluate the reached solution over different partitions of the data base, and the remaining 20% of the data was used to test the final model.

4.1 Data Sets

Three well known UCI repository data sets were used - Iris, Glass and Yeast, see table 1 for a description of each one.

5 Results

In Table 1, performance of the best 5-fold cross validation models are shown. It is possible to observe that SVMTOCP achieves lowers or almost equal prediction errors but much less support vectors (between 6% and 15% less than in the OVO case). This means a robust solution through the SVMTOCP algorithm. In addition, it is clear that the solution for each of the binary problems does not necessarily imply the same cost. The last columns of Table 1 could range for low to high cost values. The TOCP strategy also achieves higher Hard Margin Solutions (HMS) than OVO strategy.

Table 1. Data Base summary (N: #samples, Cls: #classes and Vars: #variables) and SVM strategy performance (%Err: percentage error, %SV: percentage of SV over the total number of training samples, %HMS: Percentage of Hard Margin Solutions)

Data Base	N	Classes	Vars	SVM multiclass strategy							
				OVO				TOCP			
				%Err	%SV	C	%HMS	%Err	%SV	C [min-max]	%HMS
Iris	150	3	3	6.66	44.2	3	50	0	40	[0.9 -30.83]	50
Glass	216	6	10	41.8	82.5	5	60	39.6	77.7	[0,55 – 39.79]	60
Yeast	1479	10	9	41.9	93.6	5	11	43.6	79.6	[0.24 -90.1]	56

In Fig. 2 the SVMTOCP achieved for Yeast data set is shown. The tree is balanced and no further step is required. The class structure is preserved, meaning that no sample re-allocation is necessary providing a way to identify those classes more difficult

to separate from each other (a nice property from an information discovery point of view). For instance in the Yeast data set, classes NUC and CYT are, despite having several data points, very difficult to separate, remaining together until the corresponding leaf has been reached. This conclusion can be achieved since their solution requires a high C value and produces large classification errors.

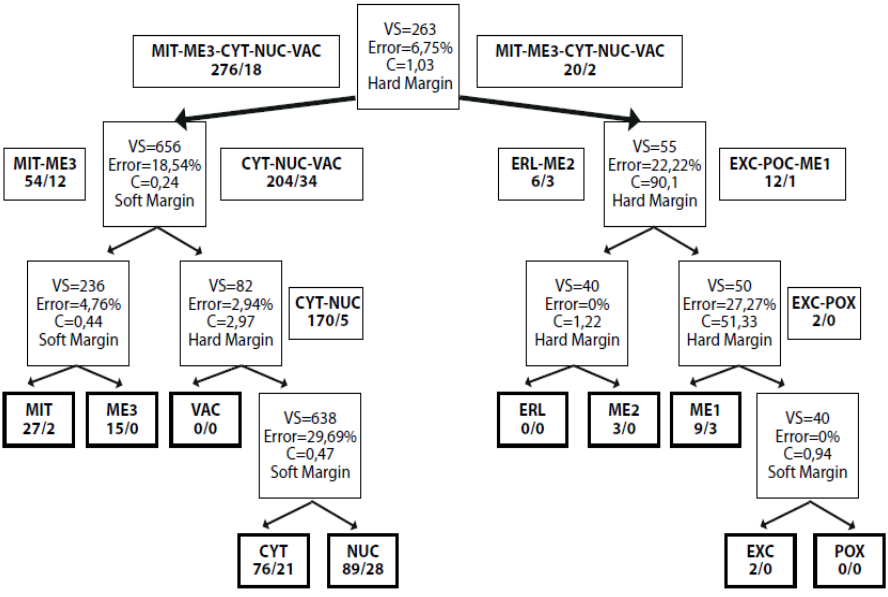

Fig. 2. SVMTOCP for Yeast data set. Square box contains class decision and an amount of correct/bad classification over 20% of testing data. It is also possible to see which nodes achieve a hard margin solution.

6 Conclusion and Future Work

The proposed method is computationally expensive; however it can be easily parallelized (work in progress) by speeding up the best SVM selection for each node. The proposed splitting method always produces a balanced tree and at the end of the training process *K-1* SVM classifiers are obtained and at most $[\log_2 K]$ SVMs are required to consult to classify a new sample. The achieved solution requires a lesser number of support vectors favorably impacting in the margin width and in solution robustness. It has been shown that the SVM for each node does not necessarily require the same cost parameters as in the previously published SVM tree based methods as well as in OVO, OVA and DAGSVM strategies.

From a Data Mining point of view, this method has several advantages. For instance, any kind of kernel could be used in each node and then, each one treated as an independent problem over the rest. By exploring nodes, it is possible to query if a particular class partition reaches a hard ($0 < \alpha_i < C$) or soft ($0 < \alpha_i \leq C$, i.e at least some $\alpha_i = C$) margin solution. In addition, the classes that remain together deep in the tree are

those most difficult to classify or separate. For instance for the Yeast Data Set a higher number of hard margin solutions were achieved (i.e: linear separable cases) compared to OVO. This information is very important when analyzing large data bases in high dimensional problems, providing useful information for the problem at hand.

Further work is required to fully evaluate error performance, since a particular solution (particular C value) could also imply choosing a particular kernel parameter (the gamma parameter in case of Gaussian/Radial kernel). Also optimizing the other kernel parameters could improve the reached solution as in the other strategies such as OVO and OVA. This will also be evaluated.

References

1. Rifkin, R., Klautau, A.: Defence of OneVs.-All Classification. Journal of Machine Learning 5, 101–141 (2004)
2. Platt, C.N., Shawe-Taylor, J.: Large margin DAGSVMs for multiclass classification. In: Advances in Neural Information Processing System, vol. 12, pp. 547–553 (2000)
3. Fei, B., Liu, J.: Binary Tree of SVM: A New fast Multiclass Training and Classification algorithm. IEEE Trans. on Neural Networks 17, 3 (2006)
4. Madzarov, G., Gjorgjevikj, D., Chorbev, I.: A multi-class SVM Classifier Utilizing Binary Decision Tree. Informatica 33, 233–241 (2009)
5. Liu, X.P., Xing, H., Wang, X.: A multistage support vector machine. In: Proc. 2nd Conf. on Mach. Learning and Cybernetics, Xi'an (2003)
6. Ben-Hur, A., Horn, D.: Support vector Clustering. The Journal of Machine Learning Research Archive 2(3/1) (2002)
7. Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge Univ. Press (2000)
8. Gjorgji, M., Dejan, G., Ivan, C.: A Multiclass SVM Classifier Utilizing Binary Decision Tree. Informatica 33, 233–241 (2009)
9. Hastie, T.: `http://www-stat.stanford.edu/~hastie/Papers/svmpath.pdf`
10. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011)