Extended Evaluation of The UPM Method for Multiclass Problems

Hernán C. Ahumada, Guillermo L. Grinblat, Pablo M. Granitto

CIFASIS, French Argentine International Center for Information and Systems Sciences, UPCAM (France) / UNR–CONICET (Argentina),
Bv. 27 de Febrero 210 Bis, 2000 Rosario, Argentina
{ahumada,grinblat,granitto}@cifasis-conicet.gov.ar

Abstract. Multiclass problems are usually of high technological value, but many classification methods are binary in origin. In the last years, several improved solutions based on the combination of simple classifiers were introduced. An interesting solution is based on creating a hierarchy of sub-problems by clustering prototypes of each one of the classes; therefore the solution is heavily influenced by the label's information. In this work we analyze a new strategy to solve multiclass problems that makes more use of spatial information than other methods. We construct a hierarchy of subproblems, but opposite to previous developments, based only on spatial information and not using a single prototype for each class. We evaluate the use of different clustering methods (either agglomerative or divisive) for this task and also the use two different classifiers (linear SVM and FDA–GenRidge) for each sub-problem (if needed, because in several cases the clustering method directly gives a subset with samples of a single class). We compare the new method with several previous approaches, finding promising results. The good performance of our approach is virtually independent of the classifier coupled to it, which suggest that it success is primarily related to the use of an appropriate clustering strategy.

1 Introduction

Many interesting machine learning tasks are multiclass classification problems, i.e., they can be posed as the problem of assigning a given example to one of a finite set of classes. For example, a handwritten recognition system has to read a series of strokes in a device and to assign them to one of the valid entries to the system. Other classical examples include text and speech categorization [3], object recognition in machine vision [10] or cancer diagnosis based on gene expression data [20]. However, the most effective and influential classifiers available nowadays, Adaboost [9] and the Support Vector Machine (SVM) [6], were designed to handle solely two classes (binary classifiers). Thus, the problem of extending these binary classifiers to multiclass problem efficiently has been the subject of several publications on the last years. Lorena et al. [17] is a valid an complete review of recent works in the field.

In this paper we describe the UPM (Unsupervised data-driven Partitioning of Multiclass problems) method, a novel strategy to deal with multiclass problems, that combines unsupervised and supervised learning. Applying a clustering algorithm, the method recursively splits the dataset in two, until the resulting sub-problems can be easily discriminated. The result of the process is a decision tree, which drives each example to an appropriate classifier at a given leaf. We analyze the performance of the new method using diverse supervised and unsupervised methods over several different multiclass datasets. Our results show that UPM has potential advantages over similar approaches.

The rest of the paper is organized as follows. In the next section we review previous works on multiclass problems, which motivate our development. In Section 3 we give a detailed presentation of the UPM method. Then, in Section 4 we evaluate the combination of UPM with different clustering methods and classifiers on five datasets and finally, in Section 5 we discuss the results and future lines of research.

2 Related Works

As we stated before, the problem of extending binary classifiers to multiclass problem has been the subject of several publications on the last years. In particular, for SVM there are some direct extensions to multiclass problems, but unfortunately they do not produce accurate classifiers in most cases. For example, Weston and Watkins [24] proposed a new formulation of the SVM (WW–SVM from here on) that can solve a multiclass problem as a single optimization task. Crammer and Singer [5], on the other side, introduced a generalized notion of the margin of multiclass problems, with which they cast them as constrained optimization tasks with a quadratic objective function, leading to multiple optimization problems of reduced size. We call that method the CS–SVM.

In general, most published efficient strategies are based on reducing the multiclass problem to a set of binary ones. There are mainly two types of approaches to do this.

The first kind of methods is based on the principle that all classes are equivalent (we call them "flat strategies") and, in order to make a decision, each class is compared to all others in the same way. The most simple method is the "One–vs-All" or OVA method [21], in which a k–class problem is decomposed into k binary sub-problems consisting in separating one of the classes from the rest. Hastie and Tibshirani [12] suggested a different approach in which all $k(k-1)/2$ pairs of classes are compared to each other. This approach is called "One–vs-One" or OVO, and is usually considered to be more effective than the OVA approach [14], in particular for SVM. Flat methods are simple to understand and implement, but they ignore useful information when solving the problem. For example, it is easy to see that some classes in a given problem could be so distant in the feature space that there is no need to train a particular classifier on them. Also, Dietterich and Bakiri [7] presented an ensemble method in which

the classes are partitioned into opposing subsets using error-correcting codes, with a redundant number of classifiers.

The second kind of methods are Hierarchical Strategies (HS), which attempt to use some of the spatial information in the problem at hand. HS methods build a decision tree (or a decision directed acyclic graph [19]) with a binary classifier at each node, and one class at each leaf. HS methods differ in the way in which each binary classifier is created [8,23]. For example, Liu et al. [16] at each step split the classes in two subsets using the k-means clustering method [18] applied to the centroids of each class, and then train an SVM to learn to discriminate both groups. The procedure is iterated until each node contains a single class. The idea behind this method (called Adaptive Hierarchical Multi-class SVM or AHM–SVM) is to look at each step for the split with the biggest separation between both subsets. The number of binary classifiers created by AHM–SVM is low, $k-1$. Benabdeslem and Bennani [4] introduced a very similar procedure (which we will call ALHC–SVM), in which the k-means clustering is replaced with the average linkage hierarchical clustering method [15].

Both HS methods described before create a hierarchy of sub-problems based on prototypes of each one of the k classes. According to this, the solution produced by the clustering stage is heavily influenced by the label's information. Furthermore, they use a single prototype for each class, assuming that all classes are compact structures in the feature space. Unfortunately, real world datasets frequently do not have this property.

3 The Unsupervised Partitioning Method for Multiclass Problems

The UPM method follows the same principle that our previous REPMAC development for imbalanced problems [1]: To divide the multiclass problem intelligently into several sub-problems (clusters) in order to translate a big multiclass problem into a set of simpler and smaller sub-problems. The general strategy is simple, we first use a clustering method in order to produce a hierarchy of sub-problems and then we train classifiers, when needed, to solve the simpler problems at the leaves of the decision tree. In Figure 1 we show a scheme of UPM.

Our strategy to solve multiclass problems makes use of even more spatial information than cited HS methods, because UPM produces a hierarchy of sub-problems, but opposite to previous developments, based only on spatial information and not using class labels at any time neither a single prototype for each class. Our goal is to split the problem into a series of simple and natural sub-problems using a given clustering method (either agglomerative or divisive). Following, we apply a classifier to each sub-problem (if needed, because in several cases the clustering method directly gives a subset with samples of a single class).

UPM

Inputs:
D : The dataset
$Cl()$: A clustering method
$DF()$: A decision function (i.e. a classifier)

Function UPM(D, Cl, DF):
1. Apply Cl to D to create D_1 and D_2
2. For $i = 1$ to 2:
IF Stopping–Criteria(D_i) is met THEN
Build a classifier $DF(D_i)$
ELSE
CALL $UPM(D_i, Cl, DF)$

Fig. 1. The UPM algorithm.

3.1 Hierarchy Construction

In order to create the hierarchy of subproblems we applied in this work two different hierarchical clustering methods.

Divisive clustering: On one side, we used a divisive method, recursively splitting the dataset in two with the well-known k-means clustering method [18] (KM from here on). Divisive clustering methods usually have a stopping criteria for the recursive process. In this case we stop the recursion if:

i) all the datapoints in a cluster belong to the same class or
ii) there are only two classes in the cluster or
iii) there are less than S_p points in the cluster [1].

The first two conditions are easy to understand, the last one stops the splitting if the cluster has only a few datapoints and a further split should leave us with less points than needed to train efficiently a classifier at the given node.

Aglomerative clustering: On the other side, we applied two well-known hierarchical clustering methods [22], which are agglomerative in nature: the Average Linkage (AL) and the Single Linkage (SL) strategies. In this case, the method builds a full hierarchy, starting from individual datapoints and ending with all the dataset in a single cluster. In order to use this hierarchy in the learning process we need to prune the resulting tree with the aim of having problems that can be solved at each leaf. According to this, starting from the root, we go down the tree and check at each node if any of the three stopping criteria explained above is met. When this is the case, we prune the tree at that node and replace the subtree with a leaf containing all the points in it.

[1] We usually use $S_p = 15$ in this work, but we discuss this selection in detail in the next section.

Classifiers: At the final step, an appropriate decision function is used to assign classes to all the points at each leaf. If the cluster has only one class (criteria (i) was met) then we obviously assign that class to the leaf. If criteria (ii) was met (a binary problem) we fit directly a binary classifier to the datapoints in the cluster. Finally, in the few cases in which the leaf was created by criteria (iii) we fit a multiclass classifier to the leaf.

In this work we use two types of classifiers. On one side we utilize the well-known SVM [6] which find the maximum-margin separating hyperplane between datapoints of different classes. Linear SVM has only one free parameter, the constant C that controls the margin. When one of the clusters meets criteria (ii) we train a binary linear SVM with the datapoints in the cluster. If the leaf was created by criteria (iii), we fit an OVO-SVM to the leaf.

We also use Flexible Discriminant Analysis as classifier. FDA was introduced by Hastie et al. [11] as an improved version of classical Fisher's LDA [13]. LDA is a standard tool for classification and dimension reduction. Roughly, it seeks a linear combination of features, which maximizes the ratio of its between-class variance to its within-class variance. After that, classes are typically assigned according to Mahalanobis distances to class centroids in this transformed space. FDA is a regularized version of LDA, more appropriate for noisy, high–dimensional situations. The method is based on recasting the LDA problem as a regularized linear regression one, and then to apply any of the many well-known techniques available for this task. We use here the standard version of FDA called Ridge Regression (GenRidge) [13], which has only one free parameter, the ridge constant λ that penalizes high values of the fitted variables. λ plays a similar role to the C parameter in SVMs, regulating the margin of the solutions [13]. As FDA is a direct multiclass classifier, we fit an standard GenRidge FDA to a given cluster when criteria (ii) or (iii) is met.

Outliers handling: Outliers, which by definition are isolated points of one class, are usually a problem for HS methods. At this first development stage we choose to ignore outliers: If at a given cluster there are less than O_{min} datapoints of a given class ($O_{min} = 4$ in this work) we consider them as outliers and do not count them when evaluating the stopping criteria or training a classifier.

3.2 Classification of New Datapoints

Once we have the full decision tree, a new example is classified following this procedure: At each level of the tree (starting from the root), the example is assigned to one of the branches, according to the rules of the clustering method (for example, for divisive KM, looking for the nearest centroid or looking for the nearest neighbor for SL). The procedure is iterated until a leaf is reached, where the example is classified using the decision function associated to that leaf (a given class for pure nodes or a classifier in other cases).

Table 1. Details of the 5 datasets used in this work. The k column shows the number of classes, p column shows the number of inputs, train and test columns show the corresponding number of datapoints in each set.

Dataset	k	p	train	test
Clouds	5	2	600	400
Pendigits	10	16	7494	3498
Letters	26	16	16000	4000
Satimage	6	36	4435	2000
Yeast	10	8	1113	371

4 Experimental Evaluation

We evaluated the performance of the UPM method first over an artificial dataset and then using 4 different datasets obtained from the UCI repository [2]. In Table 1 we show their main characteristics. All used problems have several classes and different number of samples/features. Three datasets (Clouds, Pendigits and Letters) are well balanced while the other two show classes with a low fraction of samples.

The Clouds artificial datasets is a two dimensional problem with 5 classes. Four classes are simple Gaussian distributions, but class one is a bimodal distribution created by the union of two well-separated Gaussians. Figure 2 show a sample of the dataset.

4.1 Setup

In this work we analyze the performance of UPM using three different clustering methods (KM, SL and AL) coupled with two classifiers: linear SVM and FDA–GenRidge. As we have fixed external test sets for evaluation, for each dataset we produced a single run of each method, training with the corresponding subset and after that applying the classifiers to the test set.

In the initial set of experiments we evaluated the use of SVM classifiers. In this case we selected for comparison several multiclass SVM methods that were already discussed in the Related Works section. First, we applied two direct multiclass SVM methods, the WW-SVM and the CS-SVM. Then we considered a flat method, the OVO strategy. Finally, we included two HS methods, the AHM–SVM and the ALHC–SVM. For the UPM method we used the three clustering strategies discussed before, KM, SL and AL. In all cases we used linear SVMs as classifiers, with the C parameter selected by an internal Cross Validation (CV) of the corresponding training data.

For the second set of experiments we utilized FDA–GenRidge as classifier. In this case we used for comparison the direct multiclass FDA–GenRidge (as it is available) and the OVO strategy. We also evaluated both HS strategies discussed before and the three UPM methods (KM, SL and AL), in all cases replacing the

Table 2. Comparison of the classification error using linear SVM classifiers.

	WW	CS	OVO	ALHM	AHM	KM	SL	AL
Clouds	0.115	0.113	0.110	0.113	0.110	0.023	0.108	0.025
Pendigits	0.078	0.081	0.048	0.135	0.127	0.050	0.029	0.076
Letters	0.283	0.253	0.144	0.285	0.271	0.156	0.130	0.155
Satimage	0.160	0.174	0.138	0.153	0.162	0.110	0.139	0.116
Yeast	0.402	0.512	0.394	0.412	0.461	0.434	0.442	0.439

Table 3. Comparison of the classification error using FDA-GenRidge classifiers.

	FDA	OVO	ALHM	AHM	KM	SL	AL
Clouds	0.133	0.123	0.135	0.123	0.033	0.125	0.040
Pendigits	0.171	0.060	0.181	0.166	0.048	0.029	0.078
Letters	0.305	0.189	0.373	0.307	0.165	0.098	0.155
Satimage	0.178	0.157	0.177	0.198	0.114	0.136	0.121
Yeast	0.383	0.529	0.426	0.407	0.404	0.471	0.420

linear SVM with an FDA–GenRidge classifier. In all cases the ridge constant λ was determined by an internal CV as same as previous experiment.

4.2 Results

In Table 2 we show the corresponding error levels for all methods using SVM classifiers. In four out of five problems one of the versions of UPM gives the best result and in the Yeast dataset only the OVO strategy outperforms our method. It is interesting to note that the traditional OVO method works better than all other previous methods in the five datasets considered in this experiment. Comparing the results of the three different strategies for constructing the hierarchy of subproblems in UPM, the divisive KM is superior in three cases and the agglomerative SL in others two, while AL ends in between of both methods in 4 out of the five cases.

The error levels obtained with FDA–GenRidge classifiers are shown in Table 3. The performance of all methods over the five datasets is slightly lower than the implementation with linear SVM. The key result is that, as with SVM, an UPM version outperforms the other methods in four out of the five datasets. Another interesting result is that OVO is superior to the direct implementation of FDA in all but the last dataset.

In Table 4 we show, for different values of S_p, the test set error level and the number of classifiers (grouped by type) used by the UPM method with AL clustering. It can be seen that in all datasets the influence of S_p in the performance of the method is minimum. As expected, the number of multiclass leaves increases and the total number of leaves decreases with higher S_p values. As a compromise, we choose $S_p = 15$ as the general stopping criteria in this work. With respect to the number of binary classifiers, it is interesting to note that the ALHM and AHM methods use less of them $(k - 1)$ than all UPM

Table 4. Analysis of the structure created by UPM and its dependence with the S_p parameter. The last three columns show the number of each type of classifier for the AL - linear SVM case.

Dataset	S_p	Error	One-class	Binary	Multiclass
Pendigits	10	0.075	9	15	0
	15	0.076	9	15	0
	20	0.075	9	15	0
	25	0.074	9	15	0
	30	0.075	9	15	0
Letters	10	0.155	198	194	0
	15	0.155	196	194	1
	20	0.153	182	188	12
	25	0.155	164	177	26
	30	0.155	153	162	38
Satimage	10	0.116	34	30	0
	15	0.116	34	30	0
	20	0.113	33	29	1
	25	0.114	29	29	3
	30	0.116	27	27	5
Yeast	10	0.437	23	23	0
	15	0.439	22	22	1
	20	0.439	16	20	5
	25	0.423	16	19	6
	30	0.409	13	19	7

versions, but their performance is limited, as was shown in Tables 2 and 3. On the other hand, UPM usually produces less binary classifiers than the OVO method (usually near a half of OVO's $k(k-1)/2$). Furthermore, UPM with AL produces several clusters that have only one class (no classifier needed).

4.3 Analysis

In Figure 2 we show a plot of the artificial Clouds dataset. It can be noted that classes 2 to 5 have the spatial distribution that is assumed by the other HS strategies. However, points of class 1 have a very different structure, with two well defined clusters each one. In Table 5 we report the error levels for each individual class in this problem. For "simple" classes like 2 or 3 all methods show similar results. On the other hand, for more complex classes like 1 and 5 (located between the two clusters of class 1), the new UPM strategy (in the KM and AL versions) clearly outperforms previous HS methods.

In order to add evidence in this direction, we produced a new problem derived from the Pendigits dataset by joining classes 0 and 1 in a unique class (called "0+1") and keeping the rest of the dataset unchanged. In Table 6 we show the error levels for each class using SVM classifiers and, in the last row, the results of all methods for the new combined class. It is clear from the table that

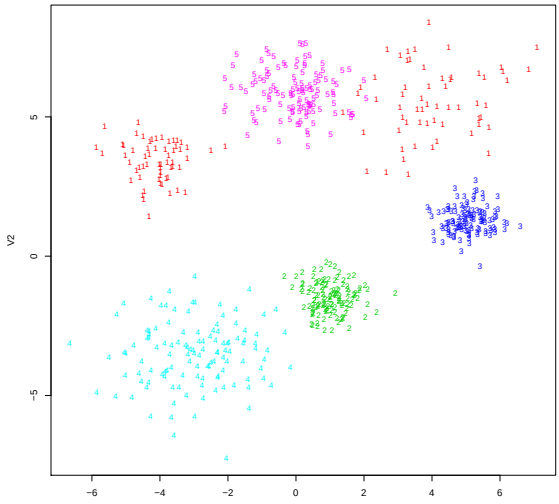

Fig. 2. Visualization of the artificial Clouds dataset.

Table 5. Clouds dataset: detail of error levels per class, using SVM classifiers.

Class	WW	CS	OVO	ALHM	AHM	KM	SL	AL
1	0.325	0.513	0.275	0.313	0.300	0.075	0.275	0.100
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.038	0.050	0.038	0.038	0.013	0.038	0.025	0.013
5	0.213	0.000	0.238	0.213	0.238	0.000	0.238	0.013

previous HS methods have increased considerably their error levels while the UPM strategy can easily cope with this problem.

The same effect can be seen in the Letters dataset. In Table 7 we show the corresponding error levels of six particular classes for all methods. Again, it is clear from the table that the complex classes do not degrade the performance of UPM methods as they do with previous HS methods.

5 Conclusions

In this work we discussed a new strategy to deal with multiclass problems, the UPM method. It has two steps, first it uses a clustering algorithm to construct a hierarchy of easier subproblems, and then it trains classifiers to solve each individual sub-problem. One of the advantages of the new method is that it usually produces clusters containing only one class, where no classifiers are needed.

Using five datasets we compared our new algorithm with previous methods for multiclass problems. Overall, UPM showed the best results in four datasets, being second in performance in the fifth case. We evaluated three different strate-

Table 6. Pendigits dataset: detail of error levels for all classes using SVM classifiers . Also, error level for an artificial class created by joining samples from classes 0 and 1.

Class	WW	CS	OVO	ALHM	AHM	KM	SL	AL
0	0.110	0.105	0.085	0.143	0.077	0.058	0.030	0.041
1	0.121	0.091	0.047	0.404	0.135	0.071	0.041	0.025
2	0.016	0.019	0.022	0.113	0.055	0.025	0.008	0.036
3	0.021	0.021	0.012	0.024	0.021	0.018	0.012	0.086
4	0.019	0.022	0.011	0.025	0.069	0.047	0.025	0.115
5	0.057	0.107	0.027	0.269	0.266	0.063	0.039	0.054
6	0.042	0.045	0.015	0.030	0.033	0.021	0.003	0.033
7	0.168	0.170	0.118	0.173	0.247	0.118	0.085	0.091
8	0.152	0.140	0.098	0.110	0.193	0.012	0.006	0.051
9	0.068	0.086	0.042	0.042	0.176	0.063	0.036	0.140
0 + 1	0.226	0.199	0.109	0.287	0.354	0.066	0.032	0.037

Table 7. Letters dataset: detail of error levels for some classes using SVM classifiers.

Class	WW	CS	OVO	ALHM	AHM	KM	SL	AL
G	0.591	0.513	0.234	0.364	0.474	0.169	0.182	0.130
L	0.250	0.197	0.151	0.276	0.217	0.125	0.112	0.118
O	0.576	0.483	0.272	0.285	0.298	0.185	0.172	0.099
Q	0.293	0.248	0.172	0.242	0.401	0.159	0.127	0.140
S	0.497	0.424	0.344	0.430	0.457	0.212	0.166	0.179
T	0.231	0.169	0.113	0.244	0.238	0.113	0.069	0.081

gies for constructing the hierarchy, one divisive and two agglomerative. The AL strategy seems to be the most useful, as it shows good results in all cases with a limited number of classifiers. The evaluation of two type of classifiers showed that the performance of UPM is similar in all cases, being only slightly lower when using FDA–GenRidge classifiers. This suggests that UPM's success is independent of the classifier selection, being more related to the use of an intelligent clustering strategy to solve multiclass problems.

We also showed that UPM's unsupervised strategy is more efficient when facing the problem of classes with complex spatial structures than previous methods, like for example some classes of the Clouds, Pendigits or Letters datasets.

Future work includes an extended evaluation of the UPM method, including other datasets and classifiers.

Acknowledgments

We acknowledge partial supported from ANPCyT by grant PICT 237/2008.

References

1. H. Ahumada, G.L. Grinblat, L.C. Uzal, P.M. Granitto, and A. Ceccatto. Repmac: A new hybrid approach to highly imbalanced classification problems. In *Eighth*

Int. Conference on Hybrid Intelligent Systems, pages 386–391, 2008.
2. A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.
3. L.R. Bahl, F. Jelinek, and R.L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE T. on Pattern Analysis and Machine Intelligence*, (2):179–190, 2009.
4. K. Benabdeslem and Y. Bennani. Dendogram-based SVM for Multi-Class Classification. *J. of Computing and Information Technology*, 14(4):283–289, 2006.
5. K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.
6. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.
7. Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, 2:263–286, January 1995.
8. B. Fei and J. Liu. Binary tree of SVM: a new fast multiclass training and classification algorithm. *IEEE T. on Neural Networks*, 17(3):696–704, 2006.
9. Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Proc. of COLT*, pages 23–37, 1995.
10. P.M. Granitto, P.F. Verdes, and H.A. Ceccatto. Large-scale investigation of weed seed identification by machine vision. *Computers and Electronics in Agriculture*, 47(1):15–24, 2005.
11. T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23:73–102, 1995.
12. T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.
13. T. Hastie, R. Tibshirani, and J.H. Friedman. *The elements of statistical learning*. Springer-Verlag, New York, 2001.
14. C.W. Hsu and C.J. Lin. A comparison of methods for multiclass support vector machines. *IEEE T. on Neural Networks*, 13(2):415–425, 2002.
15. B. King. Step-wise clustering procedures. *J. of the American Statistical Association*, 69:86–101, 1967.
16. S. Liu, H. Yi, L.T. Chia, and D. Rajan. Adaptive hierarchical multi-class SVM classifier for texture-based image classification. In *IEEE Int. Conf. on Multimedia and Expo*, pages 1–4, 2005.
17. A.C. Lorena, A.C. Carvalho, and J.M. Gama. A review on the combination of binary classifiers in multiclass problems. *Artificial Intell. Review*, 30:19–37, 2008.
18. J. McQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability*, pages 281–297, 1967.
19. J.C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Proc. of NIPS*, volume 12, pages 547–553, 2000.
20. S. Ramaswamy, P. Tamayo, R. Rifkin, et al. Multiclass cancer diagnosis using tumor gene expression signatures. *Proc. of the National Academy of Sciences USA*, 98(26):15149, 2001.
21. R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.
22. P H A Sneath and R R Sokal. *Numerical Taxonomy*. W.H. Freeman and Company, San Francisco, 1973.
23. P. Songsiri, B. Kijsirikul, and T. Phetkaew. Information-based dichotomization: A method for multiclass Support Vector Machines. In *IEEE International Joint Conference on Neural Networks*, pages 3284–3291, 2008.

24. J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proc. of 7th European Symposium On Art. Neural Networks 4-6*, 1999.