

# Aggregation schemes for binarization techniques. Methods' description

---

Mikel Galar<sup>a</sup>, Alberto Fernández<sup>b</sup>, Edurne Barrenechea<sup>a</sup>,  
Humberto Bustince<sup>a</sup>, Francisco Herrera<sup>c</sup>

<sup>a</sup>Dpto. de Automática y Computación, Public University of Navarre, 31006,  
Pamplona, Spain.

<sup>b</sup>Dept. of Computer Science, University of Jaén, 23071, Jaén, Spain.

<sup>c</sup>Dept. of Computer Science and Artificial Intelligence, University of Granada,  
18071, Granada, Spain.

## 1 Introduction

There are many classification problems involving multiple classes. Multi-category classification in Machine Learning has been widely studied, some learning algorithms are designed to tackle both binary and multiclass problems, but there are other learning techniques, which extension to multiclassification problems is not easy (such as SVMs). One way or another, the binary case where only two classes are considered is the simplest classification problem (from the point of view of the number of classes), just as more classes are considered, the difficulty of the problem is increased, this is why decomposition strategies came up. An easy way to undertake a multi-class problem is to use binarization techniques, where the original problem is decomposed in several easier binary problems.

Binary decomposition techniques (ensembles) consist in two different steps. The first one is the decomposition strategy, the most common strategies are the One-vs-One (OVO) and the One-vs-All (OVA) decompositions (other approaches such as, Error Correcting Output Code (ECOC) and hierarchical strategies are not so common in practical approaches). The second one consists in making the final class prediction from the outputs of the binary classifiers, a correct combination of classifiers outputs is crucial to make the correct prediction.

In this document, we focus our attention on the second step, the aggregation of the outputs of the binary classifiers. For this purpose, we consider separately the OVO strategies where the binarization produce  $m(m-1)/2$  binary problems from an  $m$  class problem, considering all the possible two-class combinations. And the OVA decomposition, where the binarization is made by constructing a binary classifier to discriminate each class from all other classes.

The extended descriptions of the aggregation methods follow in the next Sections. Please refer to the reference papers to obtain the full citations and references which come along such descriptions.

## 2 One-vs-One Decomposition Based Methods

OVO decomposition scheme divide an  $m$  class problem into  $m(m-1)/2$  binary problems. Each problem is face up by a binary classifier which is responsible of distinguishing between the pair of classes. The training of the classifiers is done using as training data only the instances from the original data-set which output class is one of both classes, instances with different output classes are ignored.

In validation phase, a pattern is presented to each one of the binary classifiers. The output of a classifier given by  $r_{ij} \in [0, 1]$  is the confidence of the binary classifier discriminating classes  $i$  and  $j$  in favor of the former class.

The confidence of the classifier for the latter is computed by  $r_{ji} = 1 - r_{ij}$  if the classifier do not provide it (the class with the largest confidence is the output class of a classifier). These outputs are represented by a score matrix  $R$ :

$$R = \begin{pmatrix} - & r_{12} & \cdots & r_{1m} \\ r_{21} & - & \cdots & r_{2m} \\ \vdots & & & \vdots \\ r_{m1} & r_{m2} & \cdots & - \end{pmatrix} \quad (1)$$

The final output of the system is derived from the score matrix by different aggregation models, we summarize the state-of-the-art of the combinations to obtain the final output in the following subsections.

### 2.1 Voting strategy (VOTE)

This is the most simple method to compute the output from OVO classifiers, it is also called Max-Wins rule [1]. Each binary classifier gives a vote for the predicted class. Then, the votes received by each class are counted and the class with the largest number of votes is selected as the final output. Formally, the decision rule can be written as:

$$Class = arg \max_{i=1, \dots, m} \sum_{1 \leq j \neq i \leq m} s_{ij} \quad (2)$$

where  $s_{ij}$  is 1 if  $r_{ij} > r_{ji}$  and 0 otherwise.

### 2.2 Weighted voting strategy (WV)

In the Weighted voting strategy each binary classifier votes for both classes. The weight for the vote is given by the confidence of the classifier predicting the class. The class with the largest sum value is the final output class. Hence, the decision rule is:

$$Class = arg \max_{i=1, \dots, m} \sum_{1 \leq j \neq i \leq m} r_{ij} \quad (3)$$

### 2.3 Classification by Pairwise Coupling (PC)

The classification by Pairwise Coupling [2] tries to improve the voting strategy when the outputs of the classifiers are estimated class probabilities. In that cases, this method estimates the joint probability for all classes from the pairwise class probabilities of the binary classifiers. Hence, when  $r_{ij} = \text{Prob}(Class_i | Class_i \text{ or } Class_j)$ , having  $n_{ij}$  number of instances in the

training set used to estimate the conditional probability, the method tries to find the class posterior probabilities  $p_i = Prob(Class_i)$  (where  $\sum_{i=1}^m p_i = 1$ ) which are compatible with the classifiers outputs. Usually, there is not a solution satisfying this constraints and therefore, the method finds the best approximation  $\hat{\mathbf{p}} = (\hat{p}_1, \dots, \hat{p}_m)$  according to the classifiers outputs. To do so, the Kullback-Leibler (KL) distance between  $r_{ij}$  and  $\mu_{ij}$  is minimized:

$$l(\mathbf{p}) = \sum_{1 \leq j \neq i \leq m} n_{ij} r_{ij} \log \frac{r_{ij}}{\mu_{ij}} = \sum_{i < j} n_{ij} \left( r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \mu_{ij}} \right) \quad (4)$$

where  $\mu_{ij} = p_i / (p_i + p_j)$ ,  $r_{ji} = 1 - r_{ij}$  and  $n_{ij}$  is the number of training data in the  $i$ th and  $j$ th classes. They proposed an algorithm to compute the probability estimates by minimizing eq. (4):

1. Initializations:

$$\hat{p}_i = \frac{2 \sum_{1 \leq j \neq i \leq m} r_{ij}}{m(m-1)} \text{ for all } i = 1, \dots, m \quad (5)$$

$$\hat{\mu}_{ij} = \frac{\hat{p}_i}{\hat{p}_i + \hat{p}_j} \text{ for all } i, j = 1, \dots, m \quad (6)$$

2. Repeat until convergence:

(a) Compute  $\hat{\mathbf{p}}$ :

$$\hat{p}_i = \hat{p}_i \frac{\sum_{1 \leq j \neq i \leq m} n_{ij} r_{ij}}{\sum_{1 \leq j \neq i \leq m} n_{ij} \hat{\mu}_{ij}} \text{ for all } i = 1, \dots, m. \quad (7)$$

(b) Normalize  $\hat{\mathbf{p}}$ :

$$\hat{p}_i = \frac{\hat{p}_i}{\sum_{i=1}^m \hat{p}_i} \text{ for all } i = 1, \dots, m \quad (8)$$

(c) Recompute  $\hat{\mu}_{ij}$ :

$$\hat{\mu}_{ij} = \frac{\hat{p}_i}{\hat{p}_i + \hat{p}_j} \text{ for all } i, j = 1, \dots, m \quad (9)$$

The convergence of the algorithm is proved and the obtained probabilities are consistent. The class with the largest probability estimate is the chosen output class:

$$Class = \arg \max_{i=1, \dots, m} \hat{p}_i \quad (10)$$

## 2.4 Decision Directed Acyclic Graph (DDAG)

The Decision Directed Acyclic Graph [3] constructs a rooted binary acyclic graph where each node is associated to a list of classes and a binary classifier. The root node considers all classes in the list and one classifier distinguishing between two of the classes (generally, the first and the last). According to the prediction of the classifier, the class which has not been predicted by the classifier is removed from the list and a new node is reached (the node associated to the new list, which also has another binary classifier discriminating between the first and the last classes from the new list). The last class remaining on the list is the final output class. Figure 1 illustrates the concept of a DDAG for a four class problem.

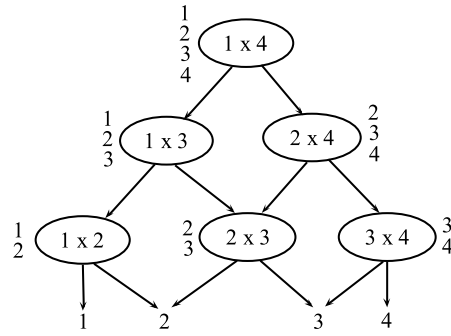


Figure 1: DDAG example for a four class problem[3]

## 2.5 Learning Valued Preference for Classification (LVPC)

Learning Valued Preference for Classification [4, 5] derives some new values from the initial confidence obtained from the binary classifiers. In this case, it is not required that the confidence in each class within a classifier to be normalized ( $r_{ji} = 1 - r_{ij}$ ), also if it is normalized, this method is a weighted voting penalizing the classifiers which have not got a certain confidence in their decision. They use a decomposition based in fuzzy preference modeling to decompose the outputs of the classifiers (weak preferences) in three values, the strict preference, the conflict (indifference in fuzzy preference modeling) and the ignorance (indistinguishability):

$$\begin{aligned}
 P_{ij} &= r_{ij} - \min\{r_{ij}, r_{ji}\} \\
 P_{ji} &= r_{ji} - \min\{r_{ij}, r_{ji}\} \\
 C_{ij} &= \min\{r_{ij}, r_{ji}\} \\
 I_{ij} &= 1 - \max\{r_{ij}, r_{ji}\}
 \end{aligned} \tag{11}$$

$C_{ij}$  is the degree of conflict (the degree to which both classes are supported),  $I_{ij}$  is the degree of ignorance (the degree to which none of the classes

is supported) and finally,  $P_{ij}$  and  $P_{ji}$  are respectively the strict preference for  $i$  and  $j$ . Note that at least one of these two degrees is zero, and that  $P_{ij} + P_{ji} + C_{ij} + I_{ij} = 1$  and that both conflict and ignorance are symmetric. Computing this values for each classifier produce three fuzzy preference relations, in [5] the authors propose the following decision rule based on a voting strategy to obtain the output class from them:

$$Class = \arg \max_{i=1, \dots, m} \sum_{1 \leq j \neq i \leq m} P_{ij} + \frac{1}{2} C_{ij} + \frac{N_i}{N_i + N_j} I_{ij} \quad (12)$$

where  $N_i$  is the number of examples from class  $i$  in the training data (and hence, an unbiased estimate of the class probability).

## 2.6 Preference Relations Solved by Non-Dominance Criterion (ND)

The Non-Dominance Criterion was originally defined for decision making with fuzzy preference relations [6], in [7] the same criterion is applied in an OVO classification systems. In this case, a fuzzy preference relation is created from the outputs of the classifiers. If this relation is not normalized ( $r_{ji} = 1 - r_{ij}$ ), then is a weak fuzzy preference relation, so it has to be normalized:

$$\bar{r}_{ij} = \frac{r_{ij}}{r_{ij} + r_{ji}} \quad (13)$$

From the normalized preference relation, the maximal non-dominated elements are calculated with the following operations:

1. Compute the fuzzy strict preference relation whose elements are  $r'_{ij}$ :

$$r'_{ij} = \begin{cases} \bar{r}_{ij} - \bar{r}_{ji}, & \text{when } \bar{r}_{ij} > \bar{r}_{ji} \\ 0, & \text{otherwise.} \end{cases} \quad (14)$$

2. Compute the non-dominance degree of each class  $ND_i$ :

$$ND_i = 1 - \sup_{j \in C} [r'_{ji}] \quad (15)$$

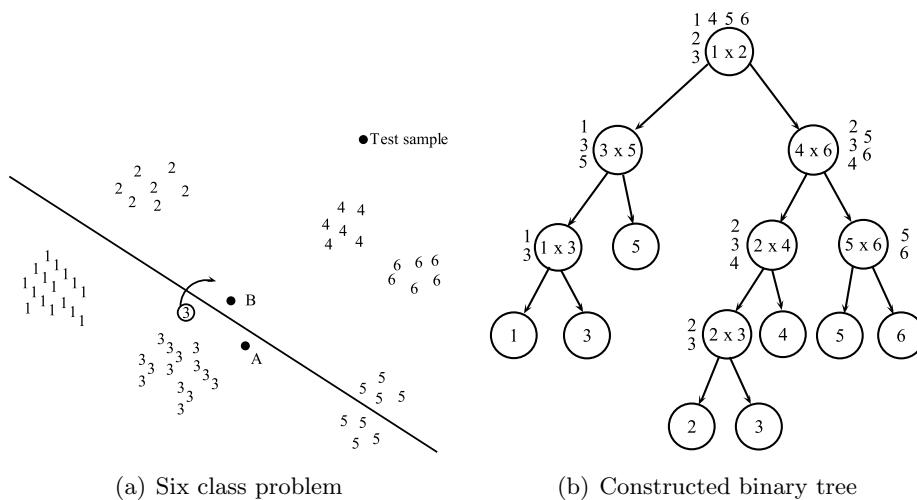
This value represents the degree to which the class  $i$  is dominated by no one of the remaining classes.  $C$  stands for the set of total classes in the data-set.

The output of the system is finally obtained as the class with the maximal non-dominance value:

$$Class = \arg \max_{i=1, \dots, m} \{ND_i\} \quad (16)$$

## 2.7 Binary Tree of Classifiers (BTC)

Binary Tree of SVM (BTS) [8], easily can be extent to any type of binary classifier. The idea behind this method is to reduce the number of classifiers and increase the global accuracy using some of the binary classifiers which discriminate between two classes to distinguish other classes at the same time. The tree is constructed recursively and in similar way to DDAG approach, each node has associated a binary classifier and a list of classes. But in this case, the decision of the classifier can distinguish other classes as well as the pair of classes used for training. So, in each node, when the decision is done, more than one class can be removed from the list. In order to avoid false assumptions, a probability is used when the examples from a class are near the boundary so the class cannot be removed from the lists in the following level. Figure 2(b) illustrates this concept applied on the six class problem in Figure 2(a). The first node classifier discriminates classes 1 and 2. On the one hand, when class 1 is predicted classes 4 and 6 are removed. On the other hand, when class 2 is predicted only class 1 is taken out. Therefore, classes 3 and 5 are maintained in both next nodes, class 3 because it is near the decision function and class 5 because it cannot be distinguished with the classifier in the root node.



**Figure 2:** Six class problem solved by a binary tree[8]. Classes 3 and 5 are assigned to two leaf nodes, class 3 by reassignment and class 5 by the decision function between class 1 and 2.

The algorithm to construct the binary tree use the following symbols and data structures:

- Trained\_list: The list containing the trained binary classifiers.
- $\delta$ : The threshold to decide whether an example is near the separating

decision function.

- $r_{ij}$  and  $r_{ji}$ : The outputs of the classifier (normalized,  $r_{ji} = 1 - r_{ij}$ ).
- $\Delta P_{ij}(x_p)$ : The reasonability of a sample  $x_i$  in a node separating classes  $i$  and  $j$  to belong to the node given by the classifier output. It is defined as

$$\Delta P_{ij}(x_p) = \begin{cases} r_{ij} - 0.5 & \text{if } r_{ij} > r_{ji} \\ r_{ji} - 0.5 & \text{otherwise} \end{cases} \quad (17)$$

Finally, the tree is obtained with the following operations:

1. Initialization:
  - (a) All the classes are assigned to the root node.
  - (b) `Trained_list` =  $\emptyset$ .
  - (c)  $K = 0$ .
2. Build the subtree of node  $K$ :
  - (a) Check if node  $K$  contains different classes, else subtree finished.
  - (b) Randomly select two classes  $i$  and  $j$ .
  - (c) Create two new child nodes 0 (for class  $i$ ) and 1 (for class  $j$ ).
  - (d) If there is not a binary classifier trained for this pair in the `Trained_list` then train it.
  - (e) Test the classifier with the training data from the rest of classes in the node assigning them to one of both child nodes.
  - (f) Compute  $\Delta P_{ij}(x_p)$  for each training data.
  - (g) For each class if all the patterns are assigned to the same child node, but at least one of them satisfy that  $|\Delta P_{ij}(x_p)| < \delta$  then reassigning the class to both child nodes.
  - (h) If all the classes are scattered into two child nodes, clear the new nodes
    - i. Select another pair in node  $K$  and go to 2(c)
    - ii. If all pairs have been evaluated, then subtree finished (“worst situation”)
  - (i) Else the pair is accepted and 2. is called recursively for both child nodes.

We develop the centered version where the classes instead of being selected randomly, are selected to create a more balanced tree. For this purpose, the centers of all classes are computed, and then, the mean center is



obtained. The nearest classes from the center are taken until an accepted pair is obtained.

The classification in the binary tree consists in starting from the root node, applying the classifier in the current node until a leaf node is reached. If the leaf contains more than one class (“worst situation”) the output is computed with the voting strategy.

## 2.8 Nesting One-vs-One (NEST)

Nesting One-vs-One algorithm [9, 10] is directly developed to tackle the unclassifiable region produced in voting strategy (it is easy to see that in a three class problem, if each binary classifier votes for a different class, there is not a winner so, some tie-breaking technique has to be applied). Nesting OVO uses the voting strategy, but when there exist examples within the unclassifiable region, a new OVO system is constructed using only the examples in the region in order to make them classifiable. This process is made until no examples remain in the unclassifiable region of the nested OVO. When there are only examples from one class in that region, there is no need to construct a new OVO, so the region is assigned to this class. Also when there are examples from two classes, a simple binary classifier which discriminates the examples from this classes in the area is just enough. The convergence of the algorithm is proved in [10], so there is no need to establish a maximum number of nesting OVO systems. The algorithm to construct the nested OVO system is the following:

1. Construct an OVO system with voting strategy.
2. Test the training examples in the OVO system.
3. Select the training examples within the unclassifiable region.
4. If there are examples from three or more classes
  - (a) Construct a new OVO system with these examples.
  - (b) Go to (2)
5. Else if there are examples from two classes, then construct a new binary classifier which discriminates between the examples from this classes in the unclassifiable region.
6. Else if there are examples from one class only, then assign the unclassifiable region to this class.
7. When the unclassifiable region disappear the algorithm is finished.

## 2.9 Wu, Lin and Weng Probability Estimates by Pairwise Coupling approach (PE)

. Probability Estimates by Pairwise Coupling [12] is similar to PC, it estimates the posterior probabilities ( $\mathbf{p}$ ) of each class starting from the pairwise probabilities. In this case, while the decision rule is equivalent (predicting the class with the largest probability), the optimization formulation is different. PE optimizes the following problem:

$$\min_{\mathbf{p}} \sum_{i=1}^m \sum_{1 \leq j \neq i \leq m} (r_{ji}p_i - r_{ij}p_j)^2 \quad \text{subject to} \quad \sum_{i=1}^k p_i = 1, p_i \geq 0, \forall i. \quad (18)$$

This problem is equivalent to:

$$\min_{\mathbf{p}} \sum_{i=1}^m \sum_{1 \leq j \neq i \leq m} (r_{ji}p_i - r_{ij}p_j)^2 \quad \text{subject to} \quad \sum_{i=1}^k p_i = 1. \quad (19)$$

It can be rewritten as

$$\min_{\mathbf{p}} 2\mathbf{p}^T Q \mathbf{p} \equiv \min_{\mathbf{p}} \frac{1}{2} \mathbf{p}^T Q \mathbf{p} \quad (20)$$

where

$$Q_{ij} = \begin{cases} \sum_{1 \leq s \neq i \leq m} r_{si}^2 & \text{if } i = j, \\ -r_{ji}r_{ij} & \text{if } i \neq j. \end{cases} \quad (21)$$

With the assumption of  $r_{ij} > 0, \forall i \neq j$ , a simple iterative method for solving the minimization is proposed in [12]:

1. Start with some initial  $p_i \geq 0, \forall i$  and  $\sum_{i=1}^k p_i = 1$ .
2. Repeat ( $t = 1, \dots, k, 1, \dots$ )

$$p_t \leftarrow \frac{1}{Q_{tt}} \left[ - \sum_{1 \leq j \neq t \leq m} Q_{tj} p_j + \mathbf{p}^T Q \mathbf{p} \right] \quad (22)$$

$$\text{normalize } \mathbf{p} \quad (23)$$

$$\text{until } \|\mathbf{Q}\mathbf{p} - \mathbf{p}^T \mathbf{Q}\mathbf{p}\mathbf{e}\|_1 = \max_t |(\mathbf{Q}\mathbf{p})_t - \mathbf{p}^T \mathbf{Q}\mathbf{p}| < 0.005/m.$$

There also exists some implementation notes in the Appendix D of [12] to reduce the computational cost.

### 3 One-vs-All Decomposition Based Methods

One-vs-All (OVA) decomposition divide an  $m$  class problem into  $m$  binary problems. Each problem is face up by a binary classifier which is responsible of distinguishing one of the classes from all other classes. The training of the classifiers is done using the whole training data, considering the patterns from the single class as positives and all other examples as negative (this can cause imbalanced training data).

In testing phase, a pattern is presented to each one of the binary classifiers and then, the classifier which gives a positive output indicates the output class. In many cases, the positive output is not unique and some tie-breaking technique has to be applied, the most common approach use the confidence of the classifiers to decide the final output. In the following subsections we summarize the state-of-the-art of OVA approach, even though OVA methods have not got the same attention in the literature like OVO ones. Instead of having a score matrix, when dealing with the outputs of OVA classifiers (where  $r_i \in [0, 1]$  is the confidence for class  $i$ ) a score vector is used:

$$R = (r_1, r_2, \dots, r_i, \dots, r_m) \quad (24)$$

#### 3.1 Maximum confidence strategy (MAX)

The Maximum confidence strategy is the most common and simple OVA method. It is similar to the weighted voting strategy from OVO systems.  $m$  classifiers are trained to distinguish each class from all others. A test example is submitted to each classifier and then, the output class is taken from the classifier with the largest positive answer:

$$Class = arg \max_{i=1, \dots, m} r_i \quad (25)$$

#### 3.2 Dinamically Ordered One-vs-All Classifiers (DOO)

The Dinamically Ordered One-vs-All[11] does not base its decision in the confidence of the OVA classifiers. OVA classifiers are trained in the same way as in the voting strategy, but in this method a Naïve Bayes classifier is also trained (using samples from all classes). This new classifier establish the order in which the OVA classifiers are executed, the OVA classifiers are executed in this order until a positive answer is obtained, which indicates the final output class. This is done dynamically for each testing example:

1. The example is submitted to the Naïve Bayes classifier.

2. OVA classifiers list is ordered with the Naïve Bayes output probabilities in descending order.
3. The example is submitted to the OVA classifiers in the list order.
4. The discriminating class from the first classifier giving a positive answer is predicted.

With this method, ties are avoided a priori by the Naïve Bayes classifier instead of relying in the degree of confidence given by the outputs of the classifiers.

## References

- [1] J.H. Friedman, Another approach to polychotomous classification. Technical report, Stanford Department of Statistics, 1996.
- [2] T. Hastie and R. Tibshirani, Classification by Pairwise Coupling, *The Annals of Statistics* 26(2) (1998) 451-471.
- [3] J. C. Platt, N. Cristianini and J. Shawe-Taylor, Large Margin DAGs for Multiclass Classification, *Proc. Neural Information Processing Systems (NIPS'99)*, S.A. Solla, T.K. Leen and K.-R. Müller (eds.), (2000) 547-553.
- [4] E. Hüllermeier and K. Brinker, Learning valued preference structures for solving classification problems, *Fuzzy Sets and Systems* 159 (2008) 2337-2352.
- [5] J.C. Hühn and E. Hüllermeier, FR3: A Fuzzy Rule Learner for Inducing Reliable Classifiers, *IEEE Transactions on Fuzzy Systems* 17(1) (2009) 138-149.
- [6] S. Orlovsky, Decision-making with a fuzzy preference relation, *Fuzzy Sets and Systems* 1 (1978) 155-167.
- [7] A. Fernández, M. Calderón, E. Barrenechea, H. Bustince, and F. Herrera. Enhancing fuzzy rule based systems in multi-classification using pairwise coupling with preference relations. In *EUROFUSE'09: Workshop on Preference Modelling and Decision Analysis*, pages 39-46, (2009).
- [8] B. Fei and J. Liu, Binary Tree of SVM: A New Fast Multiclass Training and Classification Algorithm, *IEEE Transactions on Neural Networks* 17(3) (2006) 696-704.
- [9] B. Liu, Z. Hao and X. Yang, Nesting algorithm for multi-classification problems, *Soft Computing* 11 (2007) 383-389.

- 
- [10] B. Liu, Z. Hao and E.C.C. Tsang, Nesting One-Against-One Algorithm Based on SVM for Pattern Classification, *IEEE Transactions on Neural Networks* 19(12) (2008) 2044-2052.
  - [11] J. Hong, J. Min, U. Cho and S. Cho, Fingerprint classification using one-vs-all support vector machines dynamically ordered with naïve Bayes classifiers, *Pattern Recognition* 41 (2008) 662-671.
  - [12] T.-F. Wu, C.-J. Lin, R.C. Weng, Probability estimates for multi-class classification by pairwise coupling, *Journal of Machine Learning Research* 5 (2004) 975–1005.