# Survey on Multiclass Classification Methods

Mohamed Aly <malaa@caltech.edu>

November 2005

**Abstract**

Supervised classification algorithms aim at producing a learning model from a labeled training set. Various successful techniques have been proposed to solve the problem in the binary classification case. The multiclass classification case is more delicate, as many of the algorithms were introduced basically to solve binary classification problems. In this short survey we investigate the various techniques for solving the multiclass classification problem.

## 1 Introduction

Supervised multiclass classification algorithms aim at assigning a class label for each input example. Given a training data set of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the $i$th example and $y_i \in \{1, \ldots, K\}$ is the $i$th class label, we aim at finding a learning model $\mathbb{H}$ such that $\mathbb{H}(\mathbf{x}_i) = y_i$ for new unseen examples. The problem is simply formulated in the two class case, where the labels $y_i$ are just +1 or -1 for the two classes involved. Several algorithms have been proposed to solve this problem in the two class case, some of which can be naturally extended to the multiclass case, and some that need special formulations to be able to solve the latter case. The first category of algorithms include decision trees [5, 16], neural networks [3], $k$-Nearest Neighbor [2], Naive Bayes classifiers [19], and Support Vector Machines [8]. The second category include approaches for converting the multiclass classification problem into a set of binary classification problems that are efficiently solved using binary classifiers e.g. Support Vector Machines [8, 6]. Another approach tries to pose a hierarchy on the output space, the available classes, and performs a series of tests to detect the class label of new patterns.

## 2 Extensible algorithms

The multiclass classification problem can be solved by naturally extending the binary classification technique for some algorithms. These include neural networks, decision trees, $k$-Nearest Neighbor, Naive Bayes, and Support Vector Machines.

Table 1: One-per-class Coding

| | |
|---|---|
| Class 1 | 1000 |
| Class 2 | 0100 |
| Class 3 | 0010 |
| Class 4 | 0001 |

Table 2: Distributed coding

| | |
|---|---|
| Class 1 | 00000 |
| Class 2 | 00111 |
| Class 3 | 11001 |
| Class 4 | 11110 |

## 2.1 Neural Networks

MultiLayer Feedforward Neural Networks provide a natural extension to the multiclass problem. Instead of just having one neuron in the output layer, with binary output, we could have$N$ binary neurons. The output codeword corresponding to each class can be chosen as follows: [10]:

- One-per-class coding: each output neuron is designated the task of identifying a given class. The output code for that class should be 1 at this neuron, and 0 for the others. Therefore, we will need $N = K$ neurons in the output layer, where $K$ is the number of classes. When testing an unknown example, the neuron providing the maximum output is considered the class label for that example. For instance, consider a four class problem, we will have output codes as shown in table 1.

- Distributed output coding: each class is assigned a unique binary codeword from 0 to $2^N - 1$, where $N$ is the number of output neurons. When testing an unknown example, the output codeword is compared to the codewords for the $K$ classes, and the nearest codeword, according to some distance measure, is considered the winning class. Usually the Hamming distance is used in that case, which is the number of different bits between the two codewords. For instance, for a 4 class problem, and using $N = 5$ bit codewords, the coding can be as shown in table 2. The hamming distance between each pair of classes is equal to 3 i.e. each two codes differ in three bits. If we got a codeword for an unknown example as 11101, we compute its distance from the four codewords shown above. The nearest codeword is that for class 3 with a distance of 1, so the class label assigned to that example will be class 3.

## 2.2 Decision Trees

Decision trees are a powerful classification technique. Two widely known algorithms for building decision trees are Classification and Regression Trees [5] and ID3/C4.5 [16]. The tree tries to infer a split of the training data based on the values of the available features to produce a good generalization. The split at each node is based on the feature that gives the maximum information gain. Each leaf node corresponds to a class label. A new example is classified by following a path from the root node to a leaf node, where at each node a test is performed on some feature of that example. The leaf node reached is considered the class label for that example. The algorithm can naturally handle binary or multiclass classification problems. The leaf nodes can refer to either of the $K$ classes concerned.

## 2.3 $k$-Nearest Neighbors

$k$NN [2] is considered among the oldest non-parametric classification algorithms. To classify an unknown example, the distance (using some distance measure e.g. Eculidean) from that example to every other training example is measured. The $k$ smallest distances are identified, and the most represented class in these $k$ classes is considered the output class label. The value of $k$ is normally determined using a validation set or using cross-validation.

## 2.4 Naive Bayes

Naive Bayes [19] is a successful classifier based upon the principle of Maximum A Posteriori (MAP). Given a problem with $K$ classes $\{C_1, \ldots, C_K\}$ with so-called prior probabilities $P(C_1), \ldots, P(C_K)$, we can assign the class label $c$ to an unknown example with features $\mathbf{x} = (x_1, \ldots, x_N)$ such that $c = argmax_c P(C = c \| x_1, \ldots, x_N)$, that is choose the class with the maximum a posterior probability given the observed data. This aposterior probability can be formulated, using Bayes theorem, as follows: $P(C = c \| x_1, \ldots, x_N) = \frac{P(C=c)P(x_1,\ldots,x_N \| C=c)}{P(x_1,\ldots,x_N)}$. As the denominator is the same for all clases, it can be dropped from the comparison. Now, we should compute the so-called class conditional probabilities of the features given the available classes. This can be quite difficult taking into account the dependencies between features. The naive bayes approach is to assume class conditional independence i.e. $x_1, \ldots, x_N$ are independent given the class. This simplifies the numerator to be $P(C = c)P(x_1 \| C = c) \ldots P(x_N \| C = c)$, and then choosing the class $c$ that maximizes this value over all the classes $c = 1, \ldots, K$. Clearly this approach is naturally extensible to the case of having more than two classes, and was shown to perform well inspite of the underlying simplifying assumption of conditional independence.

### 2.5 Support Vector Machines

Support Vector Machines are among the most robust and successful classification algorithms [8, 6]. They are based upon the idea of maximizing the margin i.e. maximizing the minimum distance from the separating hyperplane to the nearest example. The basic SVM supports only binary classification, but extensions [21, 4, 9, 15] have been proposed to handle the multiclass classification case as well. In these extensions, additional parameters and constraints are added to the optimization problem to handle the separation of the different classes. The formulation of [22, 4] can result in a large optimization problem, which may be impractical for a large number of classes. On the other hand, [9] reported a better formulation with a more efficient implementation.

## 3 Decomposing into binary classification

The multiclass classification problem can be decomposed into several binary classification tasks that can be solved efficiently using binary classifiers. The most successful and widely used binary classifers are the Support Vector Machines [8, 6]. The idea is similar to that of using codewords for each class and then using a number binary classifiers in solving several binary classification problems, whose results can determine the class label for new data. Several methods have been proposed for such a decomposition [1, 10, 11, 13].

### 3.1 One-versus-all (OVA)

The simplest approach is to reduce the problem of classifying among $K$ classes into $K$ binary problems, where each problem discriminates a given class from the other $K-1$ classes [18]. For this approach, we require $N = K$ binary classifiers, where the $k^{th}$ classifier is trained with positive examples belonging to class $k$ and negative examples belonging to the other $K-1$ classes. When testing an unknown example, the classifier producing the maximum ouput is considered the winner, and this class label is assigned to that example. Rifkin and Klautau [18] state that this approach, although simple, provides performance that is comparable to other more complicated approaches when the binary classifier is tuned well.

### 3.2 All-versus-all (AVA)

In this approach, each class is compared to each other class [11, 12]. A binary classifier is built to discriminate between each pair of classes, while discarding the rest of the classes. This requires building $\frac{K(K-1)}{2}$ binary classifiers. When testing a new example, a voting is performed among the classifiers and the class with the maximum number of votes wins. Results [1, 13]show that this approach is in general better than the one-versus-all approach.

Table 3: ECOC example

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ |
|---|---|---|---|---|---|---|---|
| Class 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Class 2 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| Class 3 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| Class 4 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| Class 5 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

### 3.3 Error-Correcting Output-Coding (ECOC)

This approach incorporates the idea of error-correcting codes discussed above for neural networks [10]. It works by training $N$ binary classifiers to distinguish between the $K$ different classes. Each class is given a codeword of length $N$ according to a binary matrix $M$. Each row of $M$ corresponds to a certain class. Table 3 shows an example for $K = 5$ classes and $N = 7$ bit codewords. Each class is given a row of the matrix. Each column is used to train a distinct binary classifier. When testing an unseen example, the output codeword from the $N$ classifiers is compared to the given $K$ codewords, and the one with the minimum hamming distance is considered the class label for that example. Diettrich and Bakiri [10] report improved generalization ability of this method over the above two techniques.

### 3.4 Generalized Coding

Allwein et al. [1] generalized the idea of ECOC to apply general coding techniques, where the coding matrix $M$ is allowed to take values $\{-1, 0, +1\}$. The value of $+1$ in the entry $M(k, n)$ means that examples belonging to class $k$ are considered as positive examples to classifier $n$. A value of -1 denotes that these examples are considered negative examples. A value of 0 instructs the classifier to ignore that class altogether. Clearly this scheme is the general case of the above three coding strategies. The OVA approach has a matrix $M$ such that each comlumn contains exactly one $+1$ value with the rest filled with -1. The AVA scheme has columns with exactly one $+1$ value and one -1 value with the rest set to zero's. The ECOC has the matrix $M$ filled with $+1$ and -1 values.

When testing an unknown example, the codeword "closest" to the output corresponding to that example is chosen as the class label. There is the usual hamming distance between the two codewords. In addition, Allwein et al. propose another approach for margin-based classifiers, where the output of each binary classifier is real valued and denotes the "confidence" in classification. The distance function is a certain loss function of the margin $d(M(k), f(x)) = \sum_{i=1}^{N} L(M(k, i) f_i(x))$ where $M(k)$ is the $k^{th}$row in the matrix $M$, $M(k, i)$ is the $i^{th}$entry in the $k^{th}$row, $f_i(x)$ is the output of the $i^{th}$classifier, and $L(.)$ is a specific loss function. The class having the minimum of such

distances is consifered the chosen class label.

Allwein et al. propose two methods for generating the matrix $M$. The first, called the dense method, generates codewords of length $\lceil 10 \log_2 K \rceil$. Each element is generated randomly from $\{+1, -1\}$. A code matrix is generated by randomly generating 10,000 matrices and choosing the one with the highest minimum hamming distance among its rows. The second method, called the sparse method, has codewords of length $\lceil 15 \log_2 K \rceil$. Each entry in the matrix $M$ is generated randomly to get the value of 0 with probability $\frac{1}{2}$ and {-1,+1} with probablility of $\frac{1}{4}$ each. As before, the matrix with the highest minimum hamming distance is chosen among 10,000 randomly generated matrices. Experiments performed show that the OVA scheme is generally inferior to the other approaches, while no one approach generally outperforms the others. This suggests that the best coding strategy is problem dependent.

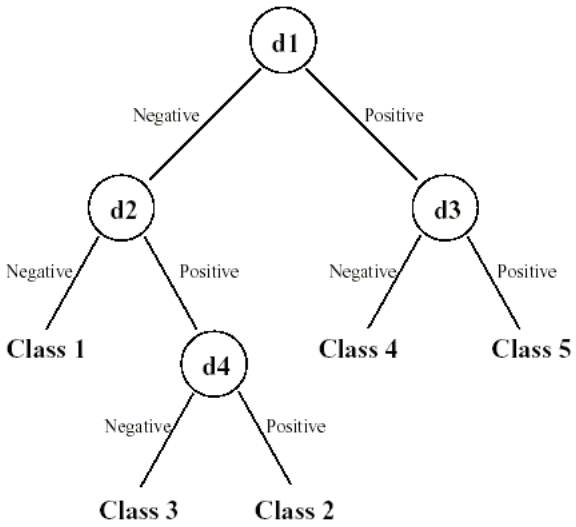
# 4 Hierarchical Classification

Yet another approach for tackling the multiclass classification problem utilizes a hierarchical division of the output space i.e. the classes are arranged into a tree. The tree is created such that the classes at each parent node are divided into a number of clusters, one for each child node. The process continues until the leaf nodes contain only a single class. At each node of the tree, a simple classifier, usually a binary classifier, makes the discrimination between the different child class clusters. Following a path from the root node to a leaf node leads to a classification of a new pattern. Several methods have been proposed for this hierarchical classification. Figure 1 shows an example tree for a 5-class problem.

Kumar et al. [14] propose a method called Binary Hierarchical Classifier (BHS). The method uses $K-1$ binary classifiiers to classifiy a $K$-class problem. The binary classifiers are arranged in a binary tree with $K$ leaf nodes, each corresponding to a given class. The binary class is recursively built top-down starting from the root node, where at each node the cluster of classes available at that node is split into two clusters. The binary split of clusters is done via a simulated-annealing like process, where the final split is the one that best discriminates the two sub-clusters according to the Fisher Discriminant. This approach was applied to various classification datasets [17], and reported comparable performance to ECOC, with the added advantage of using fewer classifiers.

Chen et al. [7] use a similar approach of clustering the classes into a binary tree called **H**ierarchical SVM (HSVM). However, the clustering is performed via arranging classes into an undirected graph, with edge weights representing the Kullback-Leibler distances between the classes, and employing a max-cut algorithm to split the classes into two sub-clusters that are most distant from each other. SVMs are used as the binary classifier at each node of the tree. They reported improved performance versus bagged classifiers using remote sensing data.

Figure 1: Example tree for 5-class problem




Vural and Dy [20] work on a similar approach of building a binary tree of $K-1$ binary classifiers, which they call Divide-By-2 (DB2). The split of classes into two clusters at each step is performed by either using k-means algorithm for clustering the class means into two groups or by using the classes grand mean as a threshold and putting classes with means smaller to the grand mean in one cluster and those with larger mean into the other. At each node in the tree, a binary SVM was used to distinguish between the two child clusters. When testing a new pattern, a path is followed from root to a leaf node, indicating the winning class label. They reported learning time to AVA approach, while testing time was superior. Moreover, DB2 exhibited better performance than OVA and AVA schemese.

# 5 Summary

This survey presented the different approaches employed to solve the problem of multiclass classification. The first approach relied on extending binary classification problems to handle the multiclass case directly. This included neural networks, decision trees, support vector machines, naive bayes, and $k$-nearest neighbors. The second approach decomposes the problem into several binary classification tasks. Several methods are used for this decomposition: one-versus-all, all-versus-all, erorr-correcting output coding, and generalized coding. The third one relied on arranging the classes in a tree, usually a binary tree,

and utilizing a number of binary classifiers at the nodes of the tree till a leaf node is reached.

# References


[1] Erin Allwein, Robert Shapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, pages 113–141, 2000.

[2] Stephen D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *Proceedings of the* $17^{th}$ *International Conference on Machine Learning*, pages 37–45, Madison, WI, 1998.

[3] Christopher M. Bishop, editor. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] Erin J. Bredensteiner and Kristin P. Bennett. Multicategory classification by support vector machines. *Computational Optimization and Applications*, 12:53–79, January 1999.

[5] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, 1984.

[6] C. J. Burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*, pages 1–47, 1998.

[7] Yangchi Chen, M. Crawford, and J. Ghosh. Integrating support vector machines in a hierarchical output space decomposition framework. In *Proceedings of Geoscience and Remote Sensing Symposium*, volume 2, pages 949–952, 2004.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, pages 273–297, 1995.

[9] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, pages 265–292, 2001.

[10] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error correcting output codes. *Journal of Artificial Intelligence Research*, 39:1–38, 1995.

[11] J. Friedman. Another approach to polychotomous classification. Technical report, Stanford University, 1996.

[12] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[13] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. In *IEEE TRANSACTIONS ON NEURAL NETWORKS*, volume 13, pages 415–425, March 2002.

[14] Shailesh Kumar, Joydeep Ghosh, and Melba M. Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, 5:210–220, 2002.

[15] Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, March 2004.

[16] J. R. Quinlan. *C4.5: Programs for Mahine Learning*. Morgan Kaufmann, 1993.

[17] Suju Rajan and Joydeep Ghosh. An empirical comparison of hierarchical vs. two-level approaches to multiclass problems. In *Multiple Classifier Systems*, pages 283–292, 2004.

[18] Ryan Rifkin and Aldebaro Klautau. Parallel networks that learn to pronounce english text. *Journal of Machine Learning Research*, pages 101–141, 2004.

[19] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI Workshop on Empirical Methods in Artificial Intelligence*, 2001.

[20] Volkan Vural and Jennifer G. Dy. A hierarchical method for multi-class support vector machines. In *Proceedings of the twenty-first international conference on Machine learning*, pages 105–112, 2004.

[21] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.

[22] J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 4 1999.