The Modularity-based Hierarchical Tree Algorithm for Multi-class Classification

Chengwei Gu[†], Bofeng Zhang[†*], Xinyue Wan, Mingqing Huang, Guobing Zou[*]
[†]Contribute equally and share first authorship
[*]Corresponding authors
School of Computer Engineering & Science
Shanghai University
Shanghai 200444, China
email: bfzhang@shu.edu.cn(Bofeng Zhang), gbzou@shu.edu.cn(Guobing Zou)

***Abstract*—Multi-class classification problem is still a research hotspot in machine learning, and researchers dedicate themselves to create new algorithms with higher efficiency and accuracy. A Modularity-based Hierarchical Classification Tree (MHCT) is proposed in this paper, which derived from the idea of community detection, and the structure of the tree is similar to the hierarchical cluster tree. This classification approach is a supervised learning method combined with modularity for its convergence indicator. After the building process of the tree from bottom to top, several classifier predictors are created in the training step. Finally, the comparison experiments are conducted between our methods and other two kinds of popular multi-class classification algorithms (i.e. support vector machine and decision tree), using ten benchmark datasets from UCI (University of California, Irvine) machine learning repository. In this work, the experimental results indicate that the proposed methods have drastically reduced the excessive training time while maintaining accuracy is comparable to the other algorithms.**

***Index Terms*—Multi-class Classification, Hierarchical Tree, Modularity**

I. Introduction

Recently, with the rapid development in the era of Big Data, multi-class classification methods are expected for automatic annotations in order to reduce the labor-cost and effectively increase the accuracy, which become the focus of attention by many domains. Data mining researchers have proposed a larger number of algorithms using supervised and unsupervised learning methods. In general, classification scheme is used to predict the category of instances whose class labels are unknown by using their observable attributes. Based on several traditional classifications algorithms, such as decision tree (DT) classifier, naive Bayes (NB) classifiers and support vector machine (SVM), many improved and advanced algorithm is proposed in the past decade and most of the state-of-the-art algorithms extend by the standard binary SVM, such as One-Versus-One SVM [1] and One-Versus-Rest SVM [2] algorithm.

Modularity [3] is designed to measure the strength of division of a network into communities. A community with high modularity means that it has dense structure and close connections between the nodes, but sparse connections exist between nodes in different communities. Hence, it is often used to detecting community in larger networks. In general, each node in the network has a set of attribute and nodes in the same community always have the common feature, so it can be easily extended to classification scenario, that is, nodes assigned to the same class are likely to be detected in the same community.

The efficiency and accuracy are two most aspects of a computer algorithm. And yet, the quality of the dataset has a significant impact on the performance of relevant algorithms in most cases, since low-quality training dataset may lead to the overfitting or fragile problem. So it is necessarily to perform preprocessing procedure when using real-world datasets, such as removal of noisy data and normalization of data. Another solution to avoid this problem is to use standard datasetm, such as UCI (University of California, Irvine) machine learning repository [4] which have been preprocessed by the provider and used for a lot of authors to test the performance of various classification algorithms.

The rest of this paper is organized as follows. We begin by presenting two classic multi-class algorithms (i.e. SVM and DT) in Section II, and each algorithm has two concrete methods. Section III provides a novel approach called Modularity based Hierarchy Classification Tree (MHCT), which is suitable for multi-class classification task. In Section IV, the comparison experiments are conducted against existing four popular algorithms using ten real benchmark datasets from UCI and analyses the experimental results. Finally, Section V concludes directions of the future work.

II. Multi-class Classification

This section introduces two kinds of famous classifiers for multi-class classification tasks, i.e. support vector machine in Section II-A and decision tree in Section II-B.

A. Support Vector Machine

The Support Vector Machine (SVM) [5] is a well-known binary classifier based on supervised learning models that is extensively used in a number of domains. It makes it possible to find the hyperplane to separate two sets of d-dimensional samples, while maximizing the distance of either sample from the hyperplane, and use the soft margin tolerance to account for misclassifications. When come to nonlinear samples, the kernel technologies [6] can be applied to meet the case. In multi-class classification task, some non-hierarchical schemes are proposed to construct One-versus-one (OVO) and one-versus-all (OVA) binary classifiers which are popular ways to extend SVM classification method for multi-class prediction.

1) One-versus-all: The OVA is also called one-versus-rest (OVR) which is one of the earliest and most widely used methods, and “Winner-Takes-Al” (WTA) [8] is the most common combination strategy for OVA shown in Fig.1. First, it constructs k binary classifier with k classes. The kth binary classifier is trained to separate the ith class with positive labels from all other classes with negative labels respectively.

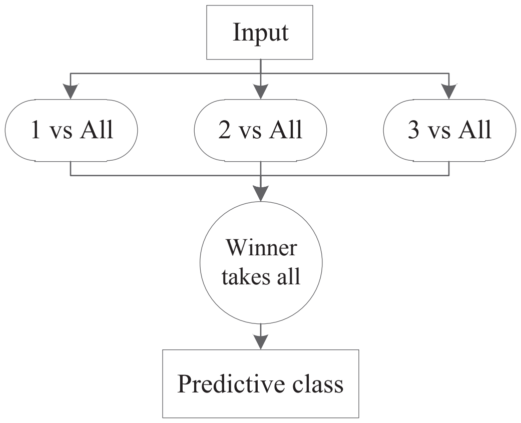

Fig. 1. One-versus-All Classifier

The input data will get k output values from k classifiers, if only the ith classifier gets positive result, then the input datas label is same as the classifiers label, otherwise, its label is up to the classifier with maximum output value.

The benefit of OVA scheme is that we only have to train k binary SVM classifiers for a k-class problem. The less classifier we get, the faster the classification speed is. However, the input data is dramatically unbalanced for each SVM classifier in many cases.

2) One-versus-one: For a k-class classification problem, OVO constructs $k(k-1)/2$ binary classifiers to separate one class from another, each for every distinct pair of classes, such as (1 vs 2, 1 vs 3, 2 vs 3, , (n-1) vs n), as shown in Fig.2. For input samples, each of binary classifier predicts samples as positive and others as negative. If one of classifier predicts sample as class i, then the vote for class i is increased by one. Lastly, the sample is assigned an instance to a class which has the largest votes from $k(k-1)/2$ binary classifiers. This voting category is called “Max-Wins Voting” (MWV) [9] which is the most popular method for the class identification of OVO.

The overall training time is reduced owing to fewer amounts of training examples than OVA for each binary classifier. However, in the voting step, there can be several single classes which get the same votes, and the input sample is assigned to multiple classes simultaneously. This will influence the classification accuracy of OVO-SVM.

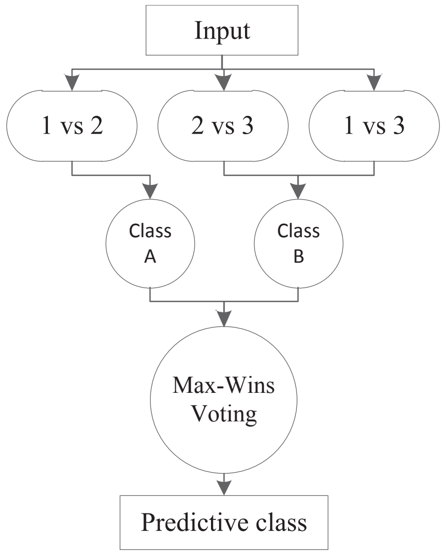

Fig. 2. One-versus-One Classifier

B. Decision Tree

Decision Tree is a prediction model which represents the mapping relationship between the attributes about an item and the items target value. In this tree, each internal node corresponds to one of possible bifurcation condition, branch connects to children node for judging results of that input variables, and the path from the root node to the leaf node denotes the final target value. For multi-class problem, the decision-tree is consists of class-labeled training tuples, and the process of constructing the tree is called ID3 (Iterative Dichotomiser 3) algorithm [7]. The other extended algorithms are introduced as follows.

1) J48 Classification Algorithm: J48 algorithm is a Java implementation of the C4.5 algorithm which inherits from the ID3 algorithm and makes a lot of improvements. Each node of the tree is represented by one of the attribute of the dataset, using the most effective splitting strategy based on the normalized information gain. The algorithm splits its dataset into subset by choosing the attribute which gets the highest normalized information gain to make the decision. In the other words, the attribute can clearly distinguish the samples belonging to its subset from others. Moreover, the data with missing attribute values are allowed by ignoring them in gain and entropy calculations. Once the tree is been created, the algorithm attempts to remove branches with few leaf nodes in order to prevent over-fitting problem.

2) Random Tree Classification Algorithm: A Random Tree [8] constructs a sets of tree predictors to make decisions, which is kind of ensemble learning method for classification and regression tasks. Because a single decision tree can grow very deep that lead to overfit the dataset, Random Tree splits the dataset into different parts and each part is trained on a tree, which reduce the deep of a signal decision trees effectively. After the trees are modeled, the class label for samples can

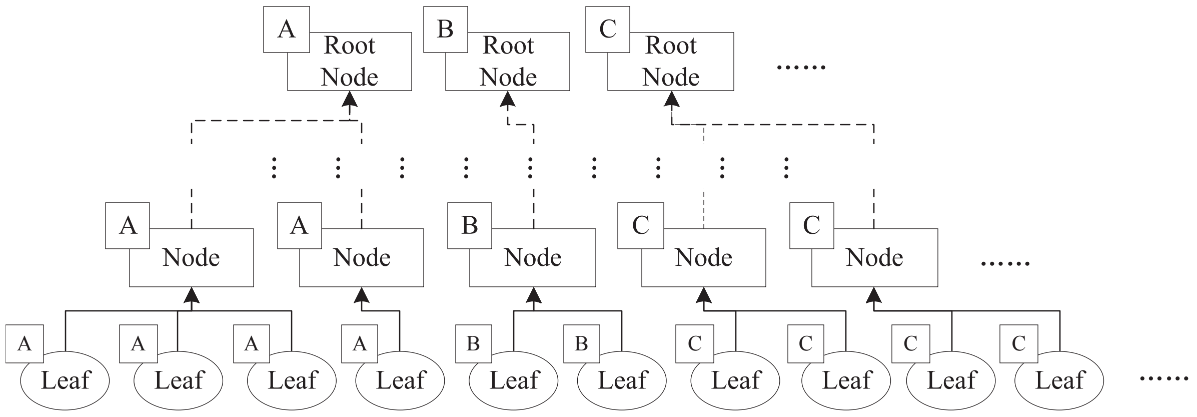

Fig. 3. Hierarchy Classification Tree

be decided by taking the majority vote among all the trees. In general, the Random Tree is constructed to distinguish the original unlabeled data from the synthetic data drawn from a reference distribution, and this algorithm has a performance of slightly increasing of bias compared with the traditional decision tree.

III. Hierarchy Classification Tree

In this paper, the MHCT is exploited as a supervised classification algorithm derived from the idea of community detection. The structure of the tree shown in Fig. 3 is similar to the hierarchical clustering tree [9], which is one of unsupervised learning methods for finding community structures in a network. During the process of training, only the agglomerative strategy is embedded in MHCT, and each leaf node, internal node and root node is labeled by a specific class (i.e. A, B, C). On the top of tree, several root nodes represents the final prediction model by using some statistic distribution of its subset. Specifically, each root node is a predictor for one specific class. When come to prediction phase, each sample in test set is assigned to a class by these predictors.

In the process of initialization, the first thing that the weight is calculated between every two leaf node represented by a set of attributes. There are a number of ways to handle this calculation, such as Euclidean distance and Cosine distance. Each node selects one of leaf nodes with largest weight and builds links between them. In the other words, this process can be described as constructing a weighting network.

At the first time, each node is an isolated leaf of a branch which has a specific class-label. Take one of nodes out of branch and put it into one of its neighboring branch, then the gain of whole trees modularity is calculated. The modularity Q is given by (1) which was proposed in [10] by Newman.

$$Q = \frac{1}{2m}\sum_{ij}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) \tag{1}$$

where A_{ij} represents the weight between node i and j, k_i is the sum of weights of links that connect to node i, and $m = \frac{1}{2}\sum_{ij} A_{ij}$. c_i is the branch to which node i belong, $\delta(c_i, c_j)$ equals 1 if $c_i = c_j$ and 0 otherwise. To further enhance efficiency, the gain in modularity ΔQ obtained by moving an isolated node i into a branch can be evaluated by (2) [11].

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{out} + k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{out}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{2}$$

where $\sum_{in}$ denotes the sum of weights of the links inside the branch, $\sum_{in}$ is the sum of the weights of the links that connect to the branch. $k_{i,in}$ is the sum of the weights of links from node i to the branch.

After traversing all the neighboring branches, one neighboring branch is selected to incorporate which the corresponding gain ΔQ is maximum and if the ΔQ is not positive or its class-label is not the same as the neighbor node, just let the node stay in its original branch. If the branch has no child node, then remove the branch. The new merged branch will repeat the process until the gain of trees ΔQ is less than the custom threshold. For more detail, the whole iterative process is shown in Algorithm 1.

After the iteration in generative process, we finally get several root nodes which includes a varying number of nodes. Each root nodes is represented by a set of attributes using all the child nodes, and each attributes r_i is computed by (3).

$$r_i = \frac{\sum\limits_{N} a_i}{N} \tag{3}$$

where a_i is the ith attribute value of child node and N is the number of child nodes. Then one of samples in testing dataset is assigned to a class which decided by a root node with the minimum cosine distance between them.

IV. Experiments

In this work, the proposed MHCT algorithm was compared with traditional multi-class classification methods with public UCI datasets shown in Section IV-A. We used the mean

Algorithm 1: Modularity-based Hierarchy Classification Tree

```
Input: $n$, Number of branches; $n_i$, Number of
       neighborhood branches $i$; $C_k$, Class of branch $k$;
       $C_j$, Class of node $j$
Output: a set of branches
$\Delta Q_{tree} = 1$;
while $\Delta Q_{tree} > threadhold$ do
    for $i = 1$ to $n$ do
        $Q_{temp} = -1$;
        $B_{incorporate} = i$;
        for $j = 1$ to $n_i$ do
            if $\Delta Q > Q_{temp}$ and $C_k == C_j$ then
                $Q_{temp} = \Delta Q$;
                $B_{incorporate} = j$;
        Move $j$ to $B_{incorporate}$;
    all merged branches become a new branch;
    $\Delta Q_{tree} = Q_{after\_incorporate} - Q_{before\_incorprate}$;
return a set of branches;
```

classification accuracy and the mean training time over three runs to show the performance of our MHCT. The Spider[1] package was used in our experiments for implementing all the multi-class classifiers, and all of our experiments were implemented in Matlab7.9 on a PC with an Intel Core i3 running at 3.40GHz and 4GB of RAM.

A. UCI Datasets

The comparative experiment was conducted among the UCI datasets. We made full use of the segment, balance, glass, letter, page, vehicle, teaching, liver, iris, wine that we briefly describe as follows. The image segment dataset is extracted randomly from a database of 7 outdoor images that were segmented into a 3x3 region, and each pixel in the region refers to a class with 19 features. The dataset contains 210 instances for training data and 2100 instances for testing data. The balance dataset consists of the results of psychological experiment about the three balance scale statuses, which has 625 instances. The glass dataset consists of 214 instances that each one is a result of chemical analysis which contains 10 attributes. The letter dataset is used to identify the 26 capital English characters (A-Z) based on 20 different fonts, and it consists of 20000 instances with 16 features. The page dataset has 5473 instances that come from 54 documents, and each instance contains some blocks of the page layout which can be identified into five different parts of a document. The vehicle dataset is about the silhouettes data of four types of model vehicles, which were extracted from constrained elevation but all angles of rotation by a camera. Each instance has 18 attributes and 846 instances. The teaching dataset is the result of evaluations of teaching performance in the university. 151 teaching assistants were graded into three categories within five semesters. The liver dataset is about the results of blood tests about alcohol that might lead to liver disorders, and it conducted a test of 345 male testers. The iris plant dataset has 150 instances with 4 attributes each, which are divided into three classes, and each class refers to a type of iris plant. Each instance in the wine dataset is one of results of a chemical analysis. There are three different types of wine grown in the same region and 178 instances represented by 13 continuous variables in the dataset.

[1]The Spider integrates many base learning algorithms for machine learning in Matlab. See http://people.kyb.tuebingen.mpg.de/spider/

TABLE I
PROPERTIES OF THE UCI DATASETS

Datasets	**Training data**	**Testing data**	**Features**	**Classes**
Segment	210	2100	19	7
Balance	125	500	4	3
Glass	100	114	9	7
Letter	2000	18000	16	26
Page	1148	4325	10	5
Vehicle	300	546	18	4
Teaching	75	76	5	3
Liver	310	35	6	2
Iris	15	135	4	3
Wine	35	143	13	73

Each of ten UCI benchmark datasets was randomly divided into two parts including training data and testing date, except the segment dataset that has already provided officially. Then the experiment performed a three-time repeated about randomly dividing and takes the average of these results.

B. Baseline and Parameter Selection

Two kinds of traditional multi-class classification algorithms described in Section II were chosen as baseline. Specifically, OVO and OVA based on standard SVM classifier are widely used as comparison experiment in multi-class classification task. Moreover, J48 tree and Random Tree based on standard DT are also two popular approaches for this task.

For OVO and OVA algorithms, the optimal values for penalty c parameters were selected from the range of $\{2^i | 1 < i < 10, i \in N\}$ and radial basis function (RBF) $k(x, x_i) = \exp(-\|x - x_i\|^2/\sigma^2)$ was selected as their kernel function to evaluate the performance, then the parameter σ was selected from $\{2^i | -10 < i < 10, i \in N\}$. For our MHCT algorithms, the 0.000001 was assigned to the threshold as a restrictive condition for the gain in modularity.

C. Result Comparisons and Discussion

The experimental results are shown in Table II. In the table, the value with bold fonts shows that the mean accuracy values are better than the other four classification methods, while the training time is shorter than others. It can be observed that all of datasets, the training time of MHCT are dramatically shorter than the SVM-based classification methods, and the time reduced for the most of the datasets is considerable. Meanwhile, the test accuracy is comparable to all the other multi-class classification technologies.

TABLE II
COMPARISON OF THE TESTING ACCURACY OF VARIOUS METHODS

UCI Dataset		SVM-OVA	SVM-OVO	J48	Random Tree	MHCT
Segment	Acc	90.19	88.90	**91.00**	88.33	85.65
	TT(s)	113.9688	286.4844	0.1406	0.0754	**0.0579**
Balance	Acc	85.80	**86.20**	77.40	76.40	81.28
	TT(s)	47.5781	42.9219	0.1094	0.0364	**0.0067**
Glass	Acc	60.53	68.42	50.00	65.89	**71.23**
	TT(s)	69.8594	157.9531	0.0781	0.0634	**0.0579**
Letter	Acc	80.3	81.39	50.00	70.92	**83.37**
	TT(s)	4994.2813	5300.4688	**0.6875**	0.1361	1.7677
Page	Acc	92.24	93.97	94.32	93.81	**94.37**
	TT(s)	649.59	544.98	0.43	0.2646	**0.2**
Vehicle	Acc	82.35	**88.24**	77.65	75.32	72.42
	TT(s)	193.25	146.9375	0.2188	**0.1729**	0.2373
Teaching	Acc	48.68	47.37	51.32	**59.21**	48.37
	TT(s)	28.2969	27.2344	0.0469	0.0232	**0.0099**
Liver	Acc	54.29	54.29	60.00	**65.71**	63.14
	TT(s)	56.75	35.0938	0.2188	0.045	**0.0171**
Iris	Acc	93.56	92.59	86.67	82.96	**95.10**
	TT(s)	27.6097	27.3281	0.0313	0.013	**0.0017**
Wine	Acc	72.03	71.33	**85.31**	83.29	74.13
	TT(s)	26.875	27.0313	0.0313	0.0143	**0.0041**

Note: Acc means the accuracy of the algorithm and TT denotes the training time.

V. CONCLUSION

In conclusion, a novel supervised algorithm for multi-class classification is discussed in this paper. This algorithm is mainly derived from the ideal of community detection and use modularity for its convergence indicator. We compared the proposed method to the other four classical methods. The experimental results have shown that MHCT algorithm we proposed is a competitive method for multi-class classification problem, that under the guarantee of the accuracy, the training time is sharply reduced compared to the classifiers based on SVM (i.e. OVA and OVO) owing to eliminating the process of selecting the optimal parameters, such as penalty parameters and kernel parameters, and the reduction of time is considerable compared to those classification methods based on decision tree (i.e. J48 and Random Tree), because the complexity of building the tree model will greatly grow with the increasing categories of datasets. MHCT has good generalization ability and does not require parameter optimization, hence, its performance is outstanding. However, there must be some possibility of optimization in the algorithm and still need further experimental validation, such as fixing the height of the tree and removing some branches with few nodes.

REFERENCES

[1] R. Debnath, N. Takahide, and H. Takahashi, “A decision based one-against-one method for multi-class support vector machine,” *Pattern Analysis and Applications*, vol. 7, no. 2, pp. 164–175, 2004.

[2] V. N. Vladimir and V. Vapnik, “The nature of statistical learning theory,” 1995.

[3] M. E. Newman, “Modularity and community structure in networks,” *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[4] A. Frank and A. Asuncion, “Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california,” *School of Information and Computer Science*, vol. 213, 2010.

[5] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[6] B. Schölkopf, A. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[7] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[8] T. K. Ho, “Random decision forests,” in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[9] M. Girvan and M. E. Newman, “Community structure in social and biological networks,” *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[10] M. E. Newman, “Analysis of weighted networks,” *Physical Review E*, vol. 70, no. 5, p. 056131, 2004.

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.