Analyzing the Performance of Hierarchical Binary Classifiers for Multi-class Classification Problem Using Biological Data

Salma Begum
Computer Science Department
University of Alabama in Huntsville
Huntsville, Alabama
E-mail: sb0034@uah.edu

Ramazan S. Aygun
Computer Science Department
University of Alabama in Huntsville
Huntsville, Alabama
E-mail: raygun@cs.uah.edu

***Abstract*— Multi-class classification problem has become a challenging problem in bioinformatics research. The problem becomes more difficult as the number of classes increases. Decomposing the problem into a set of binary problems can be a good solution in some cases. One of the popular approaches is to build a hierarchical tree structure where a binary classifier is used at each node of the tree. This paper proposes a new greedy technique for building a hierarchical binary classifier to solve multiclass problem. We use neural networks to build all possible binary classifiers and use this greedy strategy to build the hierarchical tree. This technique is evaluated and compared with two popular standard approaches One-Versus-All, One-Versus-One and a multi-class single neural network based classifier. In addition, these techniques are compared with an exhaustive approach that utilizes all possible binary classifiers to analyze how close those classifiers perform to the exhaustive method.**

Keywords- Hierarchical Binary Classifiers, Neural Networks, Error-Correcting Output Codes, Biological Data

I. INTRODUCTION

There has been significant research on multi-class classification problem [12, 14 15, 16, 21]. In binary classification, each data is classified as true (belongs to) or false (does not belong to). In multi-class classification, a data may be classified into one of many classes. Multi-class classification problem is not a rare problem in bioinformatics. For example, protein crystallization, thyroid diseases, yeast, iris, and breast tissue classification are some of the examples.

The multi-class classification problem has been usually studied in two ways: a) use a (single) multi-class classifier and b) build multiple binary classifiers. The computational complexity increases as the number of classes increases. Using a single multi-class classifier is a challenging task since the classifier needs to develop a model to distinguish a class from any other class. Using a multi-class classifier does not necessarily provide a good accuracy on the classification. Hence, decomposing multi-class problem into multiple or hierarchical binary classification problems has also been considered alternatively. This concept leads to the design of Hierarchical Binary Classifiers (HBCs) where values at a node indicate classes before splitting. Though studies on binary classification problem are relatively frequent in scientific literature, very few studies have been carried out on the performance of hierarchical binary classifiers.

For hierarchical binary classifiers, powerful binary classifiers such as support vector machines [6] are usually used. Although it is rare, multi-class classifiers such as neural networks have also been used as a binary classifier in the hierarchy [15]. Each node in the hierarchy classifies into two subsets of classes which will be called as macro-classes here. Different techniques have been proposed to select the macro-classes at each node. In [16], SVM is used to divide a *K*-class problem into two macro classes (a collection of several classes) where the number of classes in each macro-class is the same. However, this hierarchical separation with equal-sized macro-classes may not be optimal at each node. In [15], hierarchical GMDH-based neural network model was designed for multi-class problem where topological features of handwritten numerals were used for dividing the classes. In their method, they also determine the set of features to be used at each node.

This paper explores the performance of hierarchical binary classifiers to solve multiclass problem. The hierarchical binary classifiers are used in the literature, but their capabilities or limitations and performance in terms of accuracy have not been studied. Apart from hierarchical binary classifiers, two popular techniques for binary classification have also been used: One-Versus-All (OVA) and One-Versus-One (OVO). In addition, we utilize a method of using all binary classifiers to observe the performance of the hierarchical binary classifier. Since building all hierarchical binary classifiers is computation intensive, we propose a greedy technique for building hierarchical binary classifiers. We use neural networks as a binary classifier in our hierarchical binary classifiers for experimental purposes. In this paper, we compare the performance of hierarchical binary classifier with respect to OVA, OVO and multilayer perceptron neural network classifier as well. This work also includes a comparative study with an exhaustive method which is most often avoided due to its requirement of training a high number of classifiers [12]. A popular generalization method known as Error-Correcting Output Codes (ECOC) [13] is used for embedding the results of binary classifiers in OVO, OVA and exhaustive approach.

II. BACKGROUND

Multiclass classification problem is to map the data samples into more than two classes. There are two main approaches for solving multiclass classification problems. The first approach deals directly with the multiclass problem and uses algorithms like Decision Trees [1, 2], Neural Networks [3], k-Nearest Neighbor [4] and Naive Bayesian classifiers [5]. The main problem with this approach is to determine features that will distinguish classes when the number of classes increases [7]. As a result, this approach is likely to yield lower accuracy.

The second approach solves the multiclass problem by converting it into a set of binary classification problems using binary classifiers such as Support Vector Machines. Several methods have been proposed to decompose the multiple-class problem into binary problems. The One-Versus-All (OVA) and One-Versus-One (OVO) are the two popular methods of decomposition. In One-Versus-All (OVA), $\boldsymbol{K}$ class problem is solved by $\boldsymbol{K}$ binary classifiers, where each classifier discriminates a given class from the other $\boldsymbol{K-1}$ classes [8]. In One-Versus-One (OVO), a binary classifier is built to distinguish a class from each other class. This requires building $\boldsymbol{K(K-1)/2}$binary classifiers [9]. Dense [10] and sparse random [11] schemes are also introduced as a solution to decompose into binary classifiers. Another scheme known as exhaustive method generates all possible binary classifiers for a given multiclass problem [12].

Solving multiclass problem using binary classifiers also has several drawbacks. The main problem is to integrate the results of binary classifiers to classify data. Error-Correcting Output Codes (ECOC) is a general framework to integrate the results of binary classifiers to address the multiclass problem [13]. It consists of two steps: encoding and decoding.

A. Encoding Step

In the encoding stage, a codeword is assigned for each of the classes. If there are $\boldsymbol{n}$ possible binary classifiers for a $\boldsymbol{K}$-class problem, then a codeword of length $\boldsymbol{n}$ is obtained for each class where each position of the code corresponds to a response of a given binary classifier. Arranging the codewords as rows of a matrix, we define a ternary coding matrix $\boldsymbol{M}$, where $\boldsymbol{M} \in \{-1, 0, +1\}^{K \times n}$ .In this matrix $\boldsymbol{M}$, +1 and -1 is defined by the class membership of the left-right part of binary classifiers. For example, +1 for 1-2 classifier indicates that data belongs to class 1 and -1 indicates that data belongs to class 2. The value 0 is used to indicate that the class is not considered as a member of the binary classifier [20]. Fig. 1 shows an example of encoded coding matrix $\boldsymbol{M}$ for 3-class problem.

B. Decoding Step

In the decoding step, applying the $\boldsymbol{n}$ trained binary classifiers, a code is obtained for each data point in the test set. This code is compared to the base codewords of each class defined in the matrix $\boldsymbol{M}$, and the data point is assigned to the class with the "closest" codeword. The most frequently applied decoding designs are: Hamming decoding, Inverse Hamming decoding, and Euclidean decoding [13].

	Classifiers		
Class Name	**1-2**	**1-3**	**2-3**
1	+1	+1	0
2	-1	0	+1
3	0	-1	-1

a) OVO

	Classifiers		
Class Name	**1-23**	**2-13**	**3-12**
1	+1	-1	-1
2	-1	+1	-1
3	-1	-1	+1

(b) OVA

	Classifiers					
Class Name	**1-2**	**1-3**	**2-3**	**1-23**	**2-13**	**3-12**
1	+1	+1	0	+1	-1	-1
2	-1	0	+1	-1	+1	-1
3	0	-1	-1	-1	-1	+1

(c) Exhaustive

Figure 2. Encoded Coding Matrix for ECOC

There is another approach that divides the output space in a hierarchical fashion, i.e., the classes are arranged into a tree where the path from the root node to a leaf node leads to a classification of a new pattern. In Hierarchical Binary Classifiers (HBCs), each node of a tree is a binary classifier that uses $\boldsymbol{K-1}$ binary classifiers to classify a $\boldsymbol{K}$-class problem [14]. For testing, around $\boldsymbol{log_2 K}$ classifiers are required to traverse a path from top to bottom. Thus, the number of required calculations to classify is reduced in this approach. The hierarchical splitting (i.e., the macro-class selection) at each node in the hierarchy should not be done arbitrarily or by intuition. There are two different design approaches: bottom-up and top-down [17]. In this work, these two approaches are used where macro-class selection is based on greedy technique.

III. PROPOSED APPROACH

This approach solves the multiclass classification problem by a hierarchical binary classifier where a neural network (as an example) is used at each node to separate the classes. In this method, our major concern is to generate the best hierarchical tree in terms of accuracy. We use neural networks to evaluate all the binary classifiers after the training stage and get the

best tree among all possible hierarchical trees. Since the number of all possible trees becomes high, we also propose two greedy techniques (top-down and bottom-up) to separate the classes at each node in the hierarchical tree structure. Our proposed method has five major steps explained below

A. *Train all possible binary classifiers and Generate encoding codeword for ECOC framework*

First, the dataset is divided into two sets: one for training and the other one for testing. For training, we created all possible binary classifiers for multiclass problem. This approach includes the schemes OVA and OVO. The number of possible combinations for ***K***-class problem can be obtained using the following formula:

$$f(K) = \sum_{i=1}^{\lfloor \frac{K}{2} \rfloor} \sum_{j=i}^{K-i} P_{i,j,K}$$

where $P_{i,j,K} = K_{C_i} \times (K-i)_{C_j}$

if $i = j,\ P_{i,j,K} = \frac{K_{C_i} \times (K-i)_{C_j}}{2}$ **(1)**

Table 1 depicts the number of possible classifiers for class 3-8 problems.

TABLE 1 : POSSIBLE BINARY CLASSIFIERS OF DIFFERENT MULTICLASS PROBLEM

Class	No. of Binary Classifiers
3	6
4	25
5	90
6	301
7	966
8	3025

After decomposing into binary classification problem, we trained all the neural network classifiers with the training datasets and store the corresponding accuracy. Since the number of samples in each class is not equally distributed, we also measure the number of misclassified images to evaluate the performance of each classifier. To combine the results of binary classifiers, a coding matrix ***M*** for three strategies (OVA, OVO, and exhaustive) is generated during the training stage (Fig. 1). In this matrix ***M***, each row represents a code for a class which will be compared in the decoding stage.

B. *Develop a greedy hierarchical classifier*

After training, we build the hierarchical tree where the classes are separated into macro-classes at each node based on the accuracy of the neural binary classifier in the training phase. In our greedy hierarchical model, the tree has ***K-1*** binary classifiers and ***K*** leaf nodes for ***K***-class problem. For top-down approach, at the top node it selects the best binary classifier ${}_i\boldsymbol{b}_j$ that splits into two macro classes ***i*** and ***j*** where $\boldsymbol{i} \cup \boldsymbol{j} = \boldsymbol{K}$. This step is followed recursively for all the macro-classes from top to bottom and a hierarchical binary tree is built with ***K*** leaf nodes where each leaf node corresponds to a given class. For bottom-up approach a similar strategy is used starting from the bottom node. Fig. 2 describes this top-down greedy technique for 5-class problem.

In Fig. 2, at the top node best Binary Classifier ${}_{25}\boldsymbol{b}_{134}$ ({2, 5} in one macro-class, {1, 3, 4} in other macro-class) is selected from all classifiers in ${}_{I}\boldsymbol{B}_{IV}$, and ${}_{II}\boldsymbol{B}_{III}$. Here ${}_{I}\boldsymbol{B}_{IV}$ represents the classifiers that split classes into two group where the first group has one class and the other group has four classes.

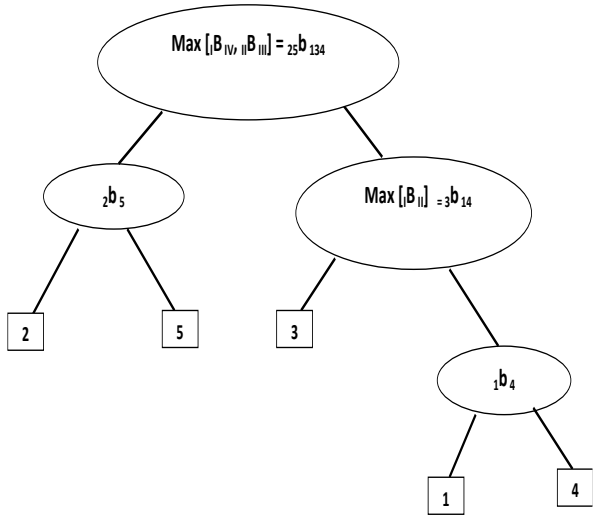

Figure 2. Top-down greedy techniques for 5-class problem

C. *Generate all possible trees*

To understand the nature of binary hierarchical trees we generated all possible trees using the following recursive equation,

$$f(K) = \sum_{i=1}^{\lfloor \frac{K}{2} \rfloor} P_{i,K}$$

where $P_{i,K} = (K_{C_i}) \times f(i) \times f(K-i)$

if $i = (K-i)$, $P_{i,K} = \frac{(K_{C_i}) \times f(i) \times f(K-i)}{2}$

$f(1) = 1$,
$f(2) = 1$ **(2)**
and K is the number of classes.

This is a recursive function and the number of possible trees becomes high with the increase in class number.
Table 2 shows the possible hierarchical binary trees for different number of classes.

TABLE 2 : POSSIBLE HIERARCHICAL BINARY TRESS OF DIFFERENT MULTICLASS PROBLEM

Class	No. of Trees
3	3
4	15
5	105
6	945
7	10395
8	135135

These trees are evaluated and the best tree is obtained by measuring the accuracy of binary classifiers used at each node in the hierarchical design. For example, to get the best tree for a 3-class problem, we need to evaluate all 3 hierarchical trees (Fig. 3). The comparative result of greedy approach with the best tree is also included in the experiments section.

D. Testing with the HBC

For Hierarchical Binary Classifiers, we start testing the samples with the binary classifier at the root node of the tree. Each node actually corresponds to a neural network classifier. If the neural network outputs a value close to 1, the right branch is chosen. Otherwise, the left-branch is chosen. Then we test the sample with the next classifier along the left or right path of the tree structure. This process is continuously followed until a leaf node of the tree where desired class of the sample is obtained.

E. Apply Hamming decoding strategy

Finally, to integrate the test results of OVA, and OVO and exhaustive approach, Hamming decoding technique is used. In this method, a coding matrix M' is obtained by testing the samples in the test set with all possible trained binary classifiers. In the matrix M', i^{th} row represents a codeword for samples i in the test set and column j is the result class value of test samples. For example, if there are 200 samples in test dataset and 6 possible binary classifiers for a 3-class problem[Table 1], then 200X6 M' is generated in exhaustive approach. In the matrix M', 198th row represents the codeword for 198th data sample. This codeword is compared

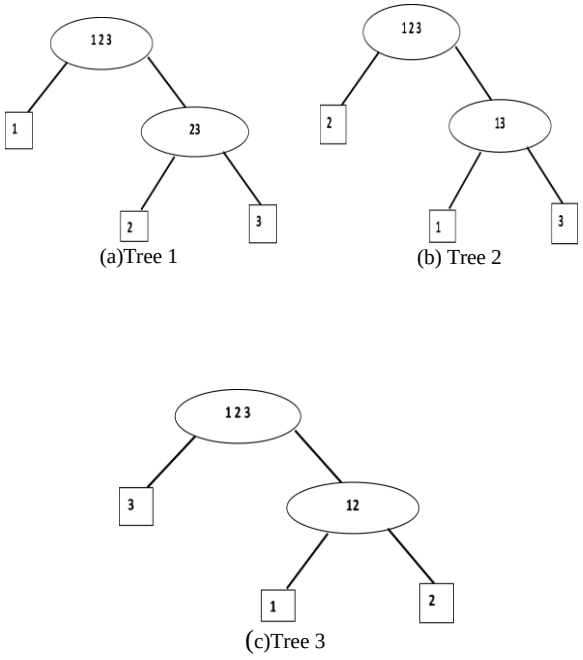

Figure 3. All hierarchical binary trees for 3-class problem

with each base codeword generated in the encoding step by finding the hamming distance [13]. The minimum distance codeword is considered to be the result class of the sample dataset.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

To solve the multiclass classification problem with different strategies and make a comparative study, we used various datasets [18]. 5 different sets of data of different classes were experimented using MATLAB [19]. The number of samples and features of different biological data for both training and testing is shown in Table 3. Among the datasets, only *Protein Crystallization* and *Iris* are equally distributed, i.e., each class has the same number of samples.

TABLE 3 : EXPERIMENTED BIOLOGICAL DATASET

No	Data set	Classes	Images	Features
1	Iris	3	150	4
2	Thyroid	3	2978	22
3	Protein Crystallization	5	100	45
4	Breast Tissue	6	106	9
5	Ecoli	8	336	7

For each dataset all possible binary classifiers based on neural networks are trained using (1). Based on the performance of trained classifiers, both top-down and bottom-up greedy hierarchical trees are created for all datasets. As it

is untraceable to show all greedy hierarchical trees, we only compare top-down and bottom-up greedy structure for *Protein Crystallization* dataset [Table 4]. In this table, ***b*** is the binary classifier and ***MS*** is the misclassified samples at that level of the tree. Note that, both the hierarchical top-down and bottom-up trees start with one-versus-all and go downward in this way. It can be also seen that, number of missed samples are less in top-down structure than bottom-up.

.

TABLE 4: TOP- DOWN AND BOTTOM-UP TREE STRUCTURES FOR PROTIEN CRYSTALLIZATION DATASET

Name	Greedy (Top-Down)	Greedy (Bottom-Up)
Tree Structure	2 **b** 1345 (MS-2) 5 **b** 134 (MS-3) 3 **b** 14 (MS-4) 1 **b** 4 (MS-1)	4 **b** 1235 (MS-6) 3 **b** 125 (MS-5) 5 **b** 12 (MS-0) 1 **b** 2 (MS-0)
Missed Samples	**10**	**11**

Table 5 shows the performance comparison of greedy strategy with OVO, OVA, exhaustive and multi-layer perceptron (MLP) network for different datasets of different classes. To integrate the results of OVO, OVA and exhaustive approach, ECOC with Hamming decoding method has been used. We also generated all possible binary hierarchical trees for *Thyroid*, *Iris* and *Protein Crystallization* using (2) and make comparison of best and worst tree with other strategies. Note that, for 3-classes performance accuracy is almost the same for all strategies. From the third row in Table 5, we see that best hierarchical binary tree outperforms greedy, MLP, OVO and OVA and the performance of worst hierarchical binary tree is very close to OVA for 5-class problem.

It can be also seen from Figure 4 that, performance accuracy of greedy strategy is high for most multiclass problems comparing to MLP, OVO and OVA and this strategy is significant for 8-class problem (93% accuracy).

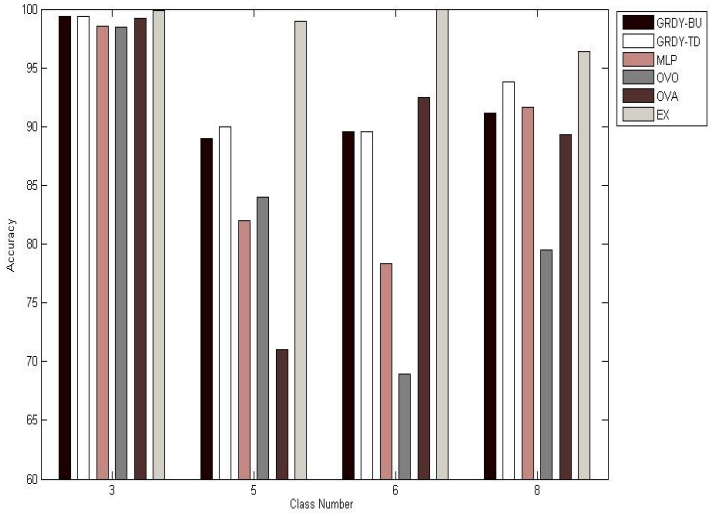

Figure 4. Comparative results of GRDY-BU(Greedy Bottom-Up), GRDY-TD (Greedy Top-Down), MLP(Multi-Layer Perceptron), OVO(One-Versus-One), OVA(One-Versus-All) and EX(Exhaustive) approach

When the number of classes becomes higher we can expect larger differences between greedy and other strategies (MLP, OVO and OVA).

Though the performance of exhaustive approach is more than 95% for all dataset, the number of classifiers for test data set increases dramatically with the increases in number of classes [Table 6]. Notice that, greedy strategy requires less number of classifiers than OVA, OVO and exhaustive approach. Though

MLP requires only one classifier solving multiclass problem, this strategy has quite lower performance than the greedy techniques.

Table 6 lists the training times of different strategies for different multiclass problems. We can see that MLP requires least training time among all strategies. It is also noticeable that training time for the greedy and exhaustive approach is same and significantly high for higher number of classes.

TABLE 6: NUMBER OF CLASSIFIERS FOR DIFFERENT STRATEGIES TO SOLVE MULTICLASS CLASSIFICATION PROBLEM

Name	(3 class)	(5 class)	(6 Class)	(8 class)
Greedy	2	4	5	7
MLP	1	1	1	1
OVO	3	10	15	28
OVA	3	5	6	8
EX	6	90	301	3025

TABLE 5: COMPARISON OF TEST RESULTS FOR DIFFERENT STRATEGIES TO SOLVE MULTICLASS CLASSIFICATION PROBLEM

No. of Classes	Best Hierarchical Binary Tree	Worst Hierarchical Binary Tree	Greedy (Bottom-Up)	Greedy (Top-Down)	Multi- Layer Perceptron	One-Versus - One	One-Versus -All	Exhaustive Approach
2(Thyroid)	100	97.84	100	100	99.7314	97	99.8	99.8
3 (Iris)	98.7	98.7	98.7	98.7	97.3	100	98.7	100
5(Protein Crystallizatio n)	92	68	89	90	82	84	71	99
6(Breast Tissue)	×	×	89.6	89.6	78.301	68.9	92.5	100
8(Ecoli)	×	×	91.17	93.79	91.667	79.5	89.3	96.42

TABLE 7: TRAINING TIME (in munities) FOR DIFFERENT STRATEGIES TO SOLVE MULTICLASS CLASSIFICATION PROBLEM

Name	Iris	Protein Crystallization	Breast Tissue	Ecoli
	(3 Class)	(5 class)	(6 Class)	(8 class)
Greedy	2 m	20m	120m	1080m
MLP	0.5m	~1m	~2m	~2m
OVO	1m	2.22m	~6m	~10m
OVA	1m	~1m	2.3m	~3m
EX	2m	20m	120m	1080m

V. CONCLUSION

The main purpose of this paper is to analyze the hierarchical binary trees for multiclass classification problem. We propose a new greedy technique to create the hierarchical tree structure where each node in the tree is a neural network based binary classifier. To compare the performance of this new approach, two other standard solution to multiclass problem OVO and OVA has been considered in this paper. Moreover, we experiment the dataset with exhaustive approach using ECOC framework to combine the results. All these approaches are conducted with 5 different biological datasets of multiple classes.

Though the exhaustive approach requires high computational resources and time, it can be used to solve critical multiclass problems where high performance is required. Surprisingly, although OVO and OVA are used frequently, they do not provide the best results. The performance of our greedy technique is better than OVO and OVA approaches for 4 different datasets. Between the two greedy structures performance result of top-down is better than bottom-up.

In this paper, we introduce two equations two generate all possible binary classifiers and hierarchical binary trees with different macro-classes which can be a good research line. Besides, we depicted the nature of hierarchical binary trees by creating all trees which is conducted with 3 different datasets. It can be seen that, the new greedy technique is a good solution rather than generating all possible hierarchical trees for high number of multiclass problem. However, greedy techniques require generation of all binary classifiers for training. Our future plan is to apply the greedy technique with other datasets of different features and improve the performance by reducing training time.

ACKNOWLEDGMENT

We would like to acknowledge Marc Pusey, Ph.D., of iXpressGenes, Inc. for providing the Protein Crystallization dataset and Madhav Sigdel for extracting features from this dataset.

REFERENCES

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

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

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

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

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

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

[7] I. Guyon, S. Gunn, M. Nikravesh and L. Zadeh, "Feature extraction," Foundations and applications, Springer, 2006.

[8] R.O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification," New York:Wiley- Interscience, 2000.

[9] T. Hastie and R. Tibshirani, “Classification by pairwise coupling,” Advances in neural information processing systems, vol. 10, MIT Press, , pp. 507—513, 1998.

[10] Allwein, R. Schapire, and Y. Singer, “Reducing multiclass to binary: A unifying approach for margin classifiers,” Journal of Machine Learning Research, 1:113–141, 2002.

[11] Escalera, O. Pujol, and P. Radeva, “Separability of ternary codes for sparse designs of error correcting output codes,” Pattern Recognition Letters, 30:285–297, 2009.

[12] N. Sánchez-Maroño, A. Alonso-Betanzos, P. Garcia-Gonzalez, and Verónica Bolón-Canedo, “Multiclass classifiers vs multiple binary classifiers using filters for feature selection,” IJCNNIEEE , p. 1-8, 2010.

[13] S. Escalera, O.l Pujol, and P. Radeva, “Error-Correcting Ouput Codes Library,” In: J. Mach. Learn. Res., Vol. 11 Cambridge, MA, USA: MIT Press, p. 661—664, March 2010.

[14] Y.-C.F. Wang and D. Casasent, “Hierarchical k-means clustering using new support vector machines for multi-class classification” In Proceedings of the international joint conf. on neural networks pp. 3457–3464, 2006.

[15] E. M. El-Alfy, “A Hierarchical GMDH-Based Polynomial Neural Network for Handwritten Numeral Recognition Using Topological Features,” In IJCNN IEEE, p. 1-7, 2010.

[16] D. Casasent and Y. -C. Wang, “A hierarchical classifier using new support vector machine for automatic target recognition” Neural Networks, 18(2), 541–548, 2005M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[17] R. Duda, P. Hart, and D. Stork, “Pattern classification,” New York:Wiley-Interscience, 2000.

[18] A. Asuncion and D.J. Newman,”UCI machine learning repository, “ University of California, Irvine, School of Information and Computer Sciences, http://mlearn.ics.uci.edu/MLRepository.html. Last access: May 2012.

[19] H. Demuth and M. Baele, “Neural Network Toolbox. User's guide,” The MathWorks, Inc., Natick, MA, 1994.

[20] S. Escalera, O. Pujol, and P. Radeva, “On the Decoding Process in Ternary Error-Correcting Output Codes,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 32, No. 1. pp. 120-134, 07 November 2008.

[21] P. Jain, P. Wadhwa, R. Aygun, and G. Podila, “Vector–G: Multi-Modular SVM-Based Heterotrimeric G-Protein Prediction,” In Silico Biol. Vol. 8, Number 2, pp. 141-155, 2008.