A Pattern Classification Approach for Boosting with Genetic Algorithms

İsmet Yalabık, Fatoş T. Yarman-Vural,
Göktürk Üçoluk, Onur Tolga Şehitoğlu
Department of Computer Engineering
Middle East Technical University, Ankara, Turkey
{ismet,vural,ucoluk,onur}@ceng.metu.edu.tr

***Abstract*— Ensemble learning is a multiple-classifier machine learning approach which produces collections and *ensembles* statistical classifiers to build up more accurate classifier than the individual classifiers. Bagging, boosting and voting methods are the basic examples of ensemble learning. In this study, a novel boosting technique targeting to solve partial problems of AdaBoost, a well-known boosting algorithm, is proposed. The proposed system finds an elegant way of boosting a bunch of classifiers successively to form a "better classifier" than each ensembled classifiers. AdaBoost algorithm employs a greedy search over hypothesis space to find a "good" suboptimal solution. On the hand, the system proposed employs an evolutionary search with genetic algorithms instead of greedy search. Empirical results show that classification with boosted evolutionary computing outperforms the classical AdaBoost in equivalent experimental environments.**

I. INTRODUCTION

The early research on pattern classification gathers around designing single classifier for a specific problem domains. Many approaches have been proposed from simple methods to complex systems. Most of these methods work in small sets of data having limited diversity. However, these methods fall too short to the real life problems. A set of classifiers with similar training performances may have better generalization performances. Moreover, slight or redundant changes in the environment affect the performance and generalization ability of the classifiers drastically. Additionally, "No free lunch theorem" states that it is impossible to design a single classifier that performs well in all of the search spaces of different classification tasks. Therefore, it can be concluded that single classifier systems suffers many problems due to the change in both conditions and nature of the problems.

Based on many theoretical and practical reasons, a new learning paradigm, ensemble learning, has emerged. Ensemble learning is the study of building collections from a set of classifiers to form a more accurate classifier. Intelligent systems and algorithms, which selects or fuses best classifiers with respect to the nature of data, are produced under ensemble learning techniques: classifier fusion methods, termed classifier selection, bagging and boosting [1]. Learning any data with combining single classifiers is called ensemble learning.

Among all ensemble learning techniques, *boosting* has been attracted more attention since mid-90's. Schapire *et al.* [2] defines boosting as a general problem of producing a very accurate prediction rule by combining rough, moderate inaccurate rules of thumb. These rules are combined in such a way that at each step moderate single classifier pays more attention on misclassified samples in the previous round. As a result, overall training error is reduced at each turn of the algorithm. Freund and Schapire are the first researchers that come up with a very successful and popular boosting algorithm called AdaBoost.

In this study, a better search method for boosting using the evolutionary search is proposed to explore over hypothesis space to find a better collection of weak classifiers. Genetic Algorithms, successively boost hypotheses coded as genes on the chromosomes (each chromosome represents a boosted final classifier). Performance evaluating function to calculate fitness values at each boosting step is introduced to decide the strength of individuals in the population or success of the solutions in the hypotheses space.

Proposed systems bear some superiorities compared to the classical AdaBoost algorithms. Evolutionary search achieves higher classification rates in the same data sets, however, complexity of combining boosting with evolutionary search is higher than the complexity of AdaBoost itself. For that reason, performing evolutionary search in hundreds of thousands of elements feature spaces and tens of thousands number of samples make evolutionary search almost impractical in absence of the high performance computing utilities.

Organization for rest of this work is as follows: in the next chapter, we will discuss Boosting and AdaBoost algorithm with its superiorities and weaknesses. Then, we will introduce a novel pattern classification approach using Evolutionary Computing in boosting context. In fourth section, we will consider experimental results that compare proposed systems with the classical AdaBoost in equal setups. Finally, we will conclude outcomes of this work.

II. BOOSTING AND ADABOOST

Boosting was first inspired from Valiant's Probably Approximately Correct Theory [3]. In 1988, Micheal Kearns and Leslie G. Valiant first raise a question of investigating whether a "weak" learning algorithm that is slightly better than random guessing can be boosted into an arbitrarily accurate "strong" learning algorithm [4]. It was 1989, when Schapire was able to come up with first polynomial time boosting algorithm [5]. Then Freund proposed a more efficient algorithm which improves Shapire's algorithm by purifying from practical drawbacks [6]. The first experiments

with these early boosting algorithms were carried out by Drucker, Schapire and Simard on an Optical Character Recognition task [7].

Adaptive Boosting Theory is first introduced in 1997 by Freund and Schapire [2]. AdaBoost is one of the earliest well-formed boosting algorithms. Prefix "Ada-" comes from the word "Adaptive", because the algorithm has the ability to adjust *adaptively* to the errors of the weak hypotheses h_i with respect to distribution of training data.

Given a training data (x_i, y_i) where $i = 1, 2, \cdots, N$, x_i is feature vectors and $y_i = 0, 1$ for negative and positive samples respectively. AdaBoost starts with weight initialization. The algorithm assigns equal weights w_i to all samples. On each round t_i, weak hypotheses are selected with respect to these weights and data distribution. Selection of a weak hypotheses is followed by weight updating schema : $w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$. After each round, correctly classified samples get lower weights, misclassified samples have higher weights. Therefore, misclassified samples are more effective than the correctly classified samples in selection of weak hypothesis on the next round. In other words, the higher weights tend to concentrate on hard examples. For this reason, hard samples (outliers, border points, noisy data) are closely involved with deciding strong classifier in later iterations.

AdaBoost is a good feature selector which minimizes the upper bound of the classification error. Boosted hypothesis selects only small set of hypothesis from a large set of hypothesis. Suppose the weak learning algorithm, when called by AdaBoost, generates a set of hypotheses with errors $\varepsilon_1, \varepsilon_2, ..., \varepsilon_t$. Then, it can be shown that the error $\varepsilon = Pr_{i\ D}\lfloor h_f(x_i) \neq y_i \rfloor$ of the final hypothesis h_f output by AdaBoost is bounded above by

$$\varepsilon = \prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

Note that the errors generated by the weak learning algorithm are not uniform, and the final error depends on the error of all of the weak hypotheses.

Number of data points always leads to a trade of. More data points represents original data much likely than less data, on the other hand, running algorithms with less amount of data requires lower computational power. Moreover, algorithms using heuristics to avoid exponential search suffer to find optimal or good suboptimal solutions due to both large amount of data and huge search space. Because, heuristic based search methods prunes the search space to make algorithms pay more attention to find a better solution, they can avoid to find optimal solution. AdaBoost is one of the heuristic methods. Selecting correct or optimal strong classifier over entire data is a *NP-Complete* process. Heuristic for selection of hypothesis with the lowest error at each iteration enables AdaBoost to work on higher (Viola *et al.* use feature vectors with $180,000$ features) dimensional feature spaces and large amount (Viola *et al.* use $20,000$ data sample for training purposes [8]) of data samples.

Algorithm 1 CLASSICAL ADABOOST

Require: training images $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ where $y_i = 0, 1$ for negative and positive examples respectively.

Ensure: a strong classifier that decides label for given unlabeled data

Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m an l are the number of negatives and positives respectively.

Repeat [1-4] for $t = 1, 2, ..., T$:

1: Normalize weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=i}^{n} w_{t,j}}$$

so that w_t is a probability distribution.

2: For each feature, j, train a classifier h_j is restricted to using a single feature. The error is evaluated with respect to $w_t, \varepsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.

3: Choose the classifier, h_t, with lowest error, ε_t.

4: Update weights:

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.

The final strong classifier is:

$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Although AdaBoost performs a heuristic search, the method is $\mathcal{O}(HTN)$ time algorithm where H is the number of classifiers in hypotheses set, T is the number of rounds that algorithm iterates and $\mathcal{O}(N)$ is the complexity of a hypothesis to calculate error. On the other hand, AdaBoost has a very fast decision making mechanism, $\mathcal{O}(T)$, basically, it requires checking several values and sign of a summation, this fact reduces testing time drastically. For that reason, AdaBoost leads applications to achieve respectable success in real-time.

Implementation of AdaBoost algorithm is quite easy. Except the WeakLearn procedures, algorithm consists of finding lowest error value among all weak classifiers, updating floating point weights of each sample and calculation of β_t and $(\log \frac{1}{\beta_t})$ values.

AdaBoost requires no parameter to tune (except T) and no prior knowledge about weak classifiers, so that *any* method for finding a distribution over data can flexibly combined as weak learning methods. These methods can be any techniques from simple learning algorithms like adaptive threshold classifiers to complex systems like Support Vector Machines or Neural Networks.

One sine qua non principle of AdaBoost is that weak hypotheses have to classify data better than chance(in binary

classification, error rate has to be lower than 0.5). In other words, if the performance of weak hypotheses are not better than chance, no strong match between problem and model is obtained.

When T gets larger and larger, overfitting problem is observed [9]. Simulation experiments that are accomplished over toy data set shows that over-fitting generally occurs even when T is extremely large. Moreover, there is no systematic approach for choosing appropriate T. In general, T is calculated using an upper limit on the error over training data.

In presence of noise and outliers, AdaBoost does not have a good performance. Because of the weight update schema, at some stages of the algorithm, some of the samples which are misclassified in some consecutive hypothesis selection rounds, gain far larger weight values than the weight values of samples which are correctly classified and located around those misclassified samples. When this situation occurs, new hypothesis to be boosted is chosen with respect to misclassified ones. This selection ends up with a performance drop, because new hypothesis misclassifies correctly classified samples at prior stages of the algorithm. Freund *et al.* proposes BrownBoost which effectively "gives up" on examples that are repeatedly misclassified to overcome the problem mentioned above [10]. Also, Domingo *et al.* introduces MadaBoost which assigns an upper bound to weight values of each sample not to allow repeatedly misclassified samples suppress correctly classified samples [11].

AdaBoost also forms a suboptimal solution. At each iteration of the algorithm, AdaBoost selects the hypothesis with the lowest error. There is no proof in the literature that heuristic which chooses the hypothesis with lowest error lead algorithm to an optimal solution.

III. MOTIVATION AND PROPOSED SYSTEMS

In order to select a weak learning algorithm, at every step of the algorithm, AdaBoost chooses a classifier with *lowest error* and partitions the search space with respect to this learner. AdaBoost proposes a greedy search in hypotheses space by selecting hypotheses with lowest error as heuristic. At this point, there comes a question that whether selecting the *ONLY* best classifiers at each steps leads algorithm to find optimal solution or not. Li *et al.* attack this question with inserting a floating search after boosting steps to perform a relatively less greedy search in their respectful work FloatBoost [12]. FloatBoost combines floating search with AdaBoost which crops out some hypotheses chosen at some prior steps of the algorithm that reduce performance of final classifier. At each round of FloatBoost, a previously selected classifier is removed from strong classifier. If general training error without removed classifier decreases. Removing a selected classifier proves that at any rounds of AdaBoost, selection of a hypothesis without the constraint that hypothesis has to have the lowest error on weight distribution, may lead to a better solution.

Fundamental motivation which this work is founded on is the question that "Is it possible to find a more accurate or optimum solution without starting or selecting a weak hypothesis that does not have the lowest error among all hypotheses in boosting steps?". In other words, this work claims that selection of a hypotheses other than the best at each round leads to a better solutions compared with heuristic that chooses hypothesis with lowest error.

Li *et al.* practically proves that it is possible to find a more accurate solution without heuristic that is used by AdaBoost. At this point, this work proposes another point of view to this problem. Let, H be the number of weak classifiers in hypotheses space and T is the number of boosting steps. When boosted hypotheses in strong classifier is coded as string of integers, first integer value of string represents first selected classifier in the first round, second integer value represents, second selected classifier in the second round, ..., T^{th} integer value of string represents T^{th} selected classifier at T^{th} round. Final classifier that AdaBoost produces a single permutation of weak learners from H^T possible solutions. It is almost impractical to search exhaustively in H^T sized-space to find such a permutation that outperforms the original AdaBoost algorithm. Using another heuristic to form such a permutation suffers from the similar problems that AdaBoost faces.

In this work, genetic algorithms are employed as a search technique to explore among the all possible permutations of weak classifiers that outperform the permutation that AdaBoost. Genetic algorithm is a computational model of machine learning that derives its behavior from a metaphor of the process of evolution in nature. In order to produce a better solution than AdaBoost, candidate solutions are represented in terms of individuals and performance of these individuals are calculated on samples of training set. As candidate solutions surmount through generations, more successful solutions survive. Finally, fittest individual at final population would lead a better solution. Genetic algorithms are known to be a general search technique compared to the greedy search, which restricts the search space compared to the exhaustive search. Mutation and crossover make genetic algorithm move randomly in the search space. However, fitness function concentrates on better solutions in existing population. This ability restricts genetic algorithms to lose time on useless solutions. For that reason, evolutionary search can be considered as a middle layer between greedy search and exhaustive search.

Representation and fitness function are the two fundamental requirements to form a genetic algorithm.

A. Representation

Every hypothesis that AdaBoost select at each round t_i is an element of hypotheses sets, H. That is $h_i = h_1, h_2, ..., h_M$, where M is the number of hypotheses and $t_i = t_1, t_2, ..., t_T$ where T is the number of rounds. These hypotheses can be complex learners like Neural Networks, Support Vector Machines or simple learners like thresholds on 1D histograms, nearest neighborhood classifier, k-means classifier. Throughout this study, weak hypotheses are the

adaptive thresholds on 1D histograms generated by a predefined feature.

- **Fixed Length Encoding:** This encoding includes a string of T integer values, each value or each gene of an individual represents a hypothesis h_i. T is number of rounds that boosting to be performed. Each element of individual is a weak learner from set H that boosting process selects at each step successively. Mutation and crossover operations change the order and values of gene permutations that produces different individuals. Representation is closed under genetic operators like mutation and crossover, mutation can change single gene value t_i to t_j where $i, j = 1, 2, \cdots, T$.

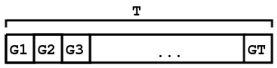

Fig. 1. Length of the encoding is T which is the number of rounds of boosting steps.

- **Variable Length Encoding:** Length of this encoding depends on the first value of the integer string. First gene of the individual can be any number that represents number of genes of the individual, rest of the individuals are in same structure as the individuals in fixed length coding. First element of the individual decides the number of boosting steps. Based on this fact, genetic algorithm finds optimum number of classifiers to be boosted.

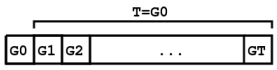

Fig. 2. Length of this encoding is $T + 1$ that is the number of rounds of boosting steps.

B. Fitness Function

Let, I_i be an individual from a population at generation G_j where $i = 1, 2, ..., K$, K is the number of individuals in the population and $j = 1, 2, ..., G$ is the number of generations that population evolves. General structure of a fitness function is as follows:

C. Initialization

Random Initialization: Many genetic algorithms start with a purely random population. Random initialization is a method that is widely used in experiments.

Partial Initialization with AdaBoost: Another approach proposed in this work is to initialize all individuals in initial population with some proportion of final classifier that AdaBoost produces after certain number of iterations. This approach starts genetic algorithm with a more qualified initial population and enhances solution of AdaBoost.

Initialization of Population by Unselected Hypotheses: To find accurate solutions, initial population has to large enough to cover all possible hypothesis in genetic pool. This requirement inspires to propose a new initialization schema. At each stage of AdaBoost, all hypothesis are sorted and recorded with respect to their error rates. Every round, AdaBoost selects a hypothesis with lowest error. However, there exist such hypotheses that are slightly worse than hypothesis with lowest error. These hypotheses are injected to genetic pool as an initialization step. In initialization phase, first individual is initialized by hypotheses with lowest

Algorithm 2 FITNESS FUNCTION

Require: training images $TR = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ where $y_i = 0, 1$ for negative and positive examples respectively.
An individual I_i from population at certain generation where $I_{i,j}$ is the element (gene) of the individual.

Ensure: a value $f(I_i)$ that represents fitness(strength) of individual I_i

Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m an l are the number of negatives and positives respectively.
Initialize if representation is variable length encoding, $T = I_{i,0}$, otherwise $T = LENGTH(I_i)$
Repeat [1-3] for $t = 1, 2, ..., T$:

1: Normalize weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=i}^{n} w_{t,j}}$$

so that w_t is a probability distribution.

2: Train classifier h_j is restricted to using single feature $I_{i,t}$. The error is evaluated with respect to $w_t, \varepsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.

3: Update weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
Form final strong classifier

$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Evaluate a function with respect to final strong classifier.

$$f(I_i) = FFUNCTION(I_i, h(x), TR) = DR - FA$$

where

$$DR = \frac{numberoftruepositives}{numberofpositivesamples},$$

$$FA = \frac{numberoffalsepositives}{numberofnegativesamples}.$$

errors, second individual is initialized by hypotheses with second lowest errors, and so on. This initialization schema is expected to improve performance of AdaBoost.

IV. EXPERIMENTS

In this study, two experimental setup is used to evaluate and compare performances of proposed systems and the classical AdaBoost.

A. Data Sets and Features

Bioinformatics data set consists of two kind of proteins: nuclear proteins and cytoplasmic proteins. Because of the fact that there are some proteins that can be found both in nucleus and in cytoplasm, this is a difficult binary classification problem. Probabilistic profiles extracted from subsequent profile map are used as features of protein sequences in this experiment. This method produces 1445 dimensional feature vector per a protein sequence.

Other data set extensively used in experiments is Face-nonface data set. In this data set, face and nonface images are collected, cropped and scaled to 32×32 resolutional images. All of these images are histogram equalized and normalized to enhance from several lighting problems. Simple Haar Wavelet Decomposition is applied to all images and 1024 dimensional feature vector is extracted from each face or non face image.

Data distributions for training and testing sets are as follows:

TABLE I
DATA DISTRIBUTION IN DATA SETS

Types of Data	*Training Set*	*Testing Set*
Nuclear Proteins	1335	890
Cytoplasmic Proteins	1335	445
Face Images	4916	4916
Nonface Images	4916	4916

B. Experiment Results

In Table II, AdaBoost (AB), fixed length encoding initialized randomly (EB), variable length encoding initialized randomly (VEB) and fixed length encoding initialized with the partial results of AdaBoost (SEB) is compared. In evolutionary algorithms, classification rate based fitness function is used to search over hypotheses space. 600 individuals are evolved for 300 generations. Single point crossover, swap mutation and roulette wheel selection mechanism are given as parameter to genetic algorithms.

Classification rate is $CR = DR - FA$.

AB (AB_{46}) combines 46 classifiers to achieve best performance on training set. Whereas, VEB (VEB_{37}) boosts 37 weak classifiers in order to achieve almost same performance that AdaBoost does. 30 weak classifiers are boosted in EB (EB_{30}) and SEB (SEB_{30}). In training set, SEB (SEB_{30}) achieves best performance in classification and detection rate by 0.693 and 0.807, respectively. On the other hand, the lowest false alarm rate is found by AB with 0.010. In case of testing set, best performance is gathered by EB with 0.603 classification rate and 0.767 detection rate. The lowest false alarm rate is again achieved by AB(AB_{46}).

TABLE II
EXPERIMENTS IN BIOINFORMATICS DATA SET

Algorithm	CR	DR	FA
AB_{46}	0.635	0.645	**0.010**
VEB_{37}	0.638	0.759	0.120
AB_{37}	0.610	0.626	0.016
EB_{30}	0.665	0.788	0.122
SEB_{30}	**0.693**	**0.807**	0.113
AB_{30}	0.606	0.640	0.033

(a) Performances in Training Set

Algorithm	CR	DR	FA
AB_{46}	0.541	0.597	**0.056**
VEB_{37}	0.564	0.712	0.148
AB_{37}	0.515	0.583	0.067
EB_{30}	**0.603**	**0.767**	0.164
SEB_{30}	0.585	0.760	0.175
AB_{30}	0.541	0.622	0.080

(b) Performances in Testing Set

TABLE III
COMPARISON OF ADABOOST AND FIXED LENGTH ENCODING WITH INITIALIZATION OF UNSELECTED HYPOTHESES.

Algorithm	DR	FA	CR
AB_{46}	0.645693	**0.0104869**	0.635206
EB_{25}	**0.783521**	0.14382	**0.6397**
AB_{25}	0.613483	0.0374532	0.57603

(a) Performances in Training Set

Algorithm	DR	FA	CR
AB_{46}	0.597753	**0.0561798**	0.541573
EB_{25}	**0.762921**	0.130337	**0.632584**
AB_{25}	0.583146	0.0808989	0.502247

(b) Performances in Testing Set

Remarkable point in this experiments is that all genetic algorithms boost better permutations than AdaBoost does. For example, in 30 weak classifier boosted classifiers, SEB (SEB_{30}), EB (EB_{30}) and AB (AB_{46}) perform 0.693, 0.665 and 0.606 classification rates in training set, respectively. Moreover, same classifiers achieve 0.585, 0.603 and 0.541 classification rates in testing set, successively. This experiment proves that better permutations with same number of classifiers can be formed other than the classical AdaBoost does with greedy search.

Another experiment conducted on Bioinformatics data set is the procedure of initialization of population by unselected hypotheses proposed in Section III-C. Experimental results show that proposed system performs slightly better than AdaBoost with relatively less number of classifiers. AdaBoost achieves best classification error by boosting 46 classifier, but genetic algorithm selects only 25 classifiers to achieve almost the same training set performance. However, there is a drastic difference in testing data (AdaBoost has 0.541 classification rate, Genetic Algorithm achieves 0.632). It can be concluded that genetic algorithm have more generalization ability.

TABLE IV

EXPERIMENTS ON FACE-NONFACE DATA SET(AVERAGE) WITH CLASSIFICATION RATE BASED FITNESS FUNCTION.

Algorithm	CR	DR	FA
AB_{42}	0.871	**0.988**	0.117
EB_{20}	0.880	0.939	0.059
AB_{20}	0.873	0.965	0.091
VEB_{37}	**0.885**	0.940	**0.055**
AB_{37}	0.870	0.985	0.115

(a) Performances in Training Set

Algorithm	CR	DR	FA
AB_{42}	0.852	**0.980**	0.128
EB_{20}	0.864	0.932	0.067
AB_{20}	0.860	0.960	0.100
VEB_{37}	**0.871**	0.933	**0.062**
AB_{37}	0.853	**0.980**	0.127

(b) Performances in Testing Set

Table IV shows average values of 10 experiments performed on face-nonface data set. The first row AB_{42} is the best average AdaBoost performance achieved on training data. Average number of boosted weak classifiers in all experiments by AdaBoost is 42. The second and third rows EB_{20} and AB_{20} are the performances of genetic algorithm with fixed length encoding initialized randomly and AdaBoost by 20 boosted weak classifiers,respectively. Finally last two rows VEB_{37} and AB_{37} are variable length encoding initialized randomly and AdaBoost. In last two rows, variable length encoding genetic algorithm and AdaBoost with same number of weak classifiers are compared. Genetic algorithms achieves better performances even with same and less number of boosted weak classifiers.

In this data set, AB (AB_{37}) accomplishes 0.988 detection rate which is higher than EB (EB_{20}) and VEB (VEB_{37}), 0.939 and 0.940, respectively. This is the opposite behavior observed in Bioinformatics data set. On average performances, genetic algorithms have slightly better performance than the classical AdaBoost does with the same number of boosted weak classifiers. T best performances are also produced by genetic algorithms.

V. CONCLUSION

At first glance, combining boosting idea with evolutionary search seems to be a good alternative to AdaBoost the existence of better permutations in search space enables one to find it by evolving generations. Let H be the number of classifiers in the hypothesis space and T be the number boosted weak classifiers. AdaBoost forms a good solution from H^T dimensional space with greedy search. Fixed length solution tries to find fittest individual in this space. As H and T becomes larger and larger, hypothesis space grows exponentially. Despite of this large space, in all experiments, genetic algorithms are able to come up with a better strong classifiers (classifiers with lower training and testing errors).

In case of variable length encoding, dimension of search space is $\sum_{i=1}^{T} H^i$ because first gene of the encoding can be any value from 1 to T. Variable length encoding searches in a drastically larger space. In order to find good solutions, more and more individual has to be in populations and there has to exist more and more generations. On the contrary, wider space means more diversity and more possible solutions.

In terms of complexities, AdaBoost tries to form a strong classifier by T iterations, among H hypotheses. Complexity for AdaBoost is $\mathcal{O}(HTN)$, where N is the number of samples and $\mathcal{O}(N)$ is the complexity for calculating error of a classifier. In the proposed method, complexity for fitness function is $\mathcal{O}(TN)$, where T is the number of boosted classifiers and again $\mathcal{O}(N)$ is the complexity for calculating error of a classifier. For this reason, complexity of the genetic algorithm is $\mathcal{O}(GPTN)$, where G is the number of evolving generations and P is the number of populations. Complexities of proposed systems and AdaBoost differs from two entities, PG and H. In experiments, PG(600 individuals with 300 generations) is far larger that H (1445 in Bioinformatics data set and 1024 in Face-Nonface data set).

Empirical results show that training time for AdaBoost is about 20 minutes on average, whereas evolutionary algorithms need 70 hours on average. However, calculation of fitness scores are completely independent processes. For that reason, genetic algorithms can be easily computed with high performance computing systems.

Our main goal in this study is to find better permutations or collections of weak classifiers that build up strong classifiers. Experiment results show that classification for boosting with genetic algorithms is another alternative boosting technique that produces better solutions than the classical AdaBoost produces.

REFERENCES

[1] L. Kuncheva, *Combining Pattern Classifiers*. John Wiley and Sons Inc, 2004.

[2] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, August 1997.

[3] L. G. Valiant, "A theory of learnable," *Communacations of the ACM*, vol. 27, pp. 1134–1142, November 1984.

[4] M. Kearns and L. G. Valiant, "Learning boolean formula and finite automata," tech. rep., Harvard University Aiken Computation Laboratory putation Laboratory, 1988.

[5] R. E. Schapire, "The strength of weak learnability," *International Conference on Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[6] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.

[7] H. Drucker, R. Schapire, and P. Simard, "Boosting performance in neural networks," *International Conference on Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 705–719, 1993.

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision Pattern Recognition(CVPR)*, 2001.

[9] G. Ratsch, T. Onoda, and K. R. Muller, "An improvement of adaboost to avoid overfitting," *Advances in Neutral Information Processing Systems*, 1999.

[10] R. McDonald, D. Hand, and I. Eckley, "An empirical comparison of three boosting algorithms on real data sets with artificial class noise," *Multiple Classifier Systems*, 2003.

[11] C. Domingo and O. Watanabe, "Madaboost: A modification of adaboost," in *Proc. 13th Annu. Conference on Comput. Learning Theory*, 2000.

[12] S. Li, Z. Zhang, H. Shum, and H. Zhang, "Floatboost learning for classification," *NIPS 15.*, 2002.