

A PATTERN CLASSIFICATION APPROACH BOOSTED WITH GENETIC
ALGORITHMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

İSMET YALABIK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2007

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Volkan Atalay
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Fatoş Tünay
Yarman-Vural
Supervisor

Examining Committee Members

Prof. Dr. Volkan Atalay (METU, CENG) _____

Prof. Dr. Fatoş Tünay Yarman-Vural (METU, CENG) _____

Prof. Dr. Uğur Halıcı (METU, EEE) _____

Assoc. Prof. Dr. Göktürk Üçoluk (METU, CENG) _____

Dr. Onur Tolga Şehitoğlu (METU , CENG) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: İsmet Yalabık

Signature :

ABSTRACT

A PATTERN CLASSIFICATION APPROACH BOOSTED WITH GENETIC ALGORITHMS

Yalabık, İsmet

M.S., Department of Computer Engineering

Supervisor : Prof. Dr. Fatoş Tünay Yarman-Vural

June 2007, 55 pages

Ensemble learning is a multiple-classifier machine learning approach which combines, 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 thesis, a novel boosting technique targeting to solve partial problems of AdaBoost, a well-known boosting algorithm, is proposed. The proposed systems find an elegant way of boosting a bunch of classifiers successively to form a better classifier than each ensembled classifier. AdaBoost algorithm employs a greedy search over hypothesis space to find a good suboptimal solution. On the other hand, this work proposes an evolutionary search with genetic algorithms instead of greedy search. Empirical results show that classification with boosted evolutionary computing outperforms AdaBoost in equivalent experimental environments.

Keywords: Boosting, AdaBoost, Genetic Algorithms, Evolutionary Computing, Classification, Pattern Recognition

ÖZ

GENETİK ALGORİTMA İLE DESTEKLENMİŞ BİR ÖRÜNTÜ SINIFLANDIRMA YAKLAŞIMI

Yalabık, İsmet

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Fatoş Tünay Yarman-Vural

Haziran 2007, 55 sayfa

Birleştirerek öğrenme, istatistiksel sınıflandırıcıları birleştirerek daha doğru bir sınıflandırıcı üreten, koleksiyonlar oluşturan bir makina öğrenme yöntemidir. Çantalama, destekleme ve oylama methodları bu yaklaşımın örneklerindendir. Bu çalışmada, en iyi tanınan destekleme algoritmalarından olan Adaptif Destekleme'nin çeşitli problemlerini çözmeye çalışan yeni bir destekleme tekniği önerilmiştir. Önerilen sistemler, bir grup sınıflandırıcıyı birleştirmenin güzel bir yolunu bularak, birleştirilen sınıflandırıcılardan daha kaliteli bir sınıflandırıcı bulmaktadır. Adaptif destekleme daha iyi bir çözüm bulmak için buluşsal yöntemler kullanmaktadır. Öte yandan, bu çalışma genetik algoritmalar ile evrimsel hesaplama yöntemi kullanarak daha iyi bir eniyileme gerçekleştirmektedir. Deneysel sonuçlar, önerilen sistemlerin Adaptif Destekleme yönteminden daha iyi performans sağladığını göstermektedir.

Anahtar Kelimeler: Destekleme, Adaptif Destekleme, Genetik Algoritmalar, Evrimsel Hesaplama, Sınıflandırma, Örüntü Tanıma

To my parents, my younger brother and Fulya...
For being with me, all the time...

ACKNOWLEDGMENTS

I would like to express my inmost gratitude to my supervisor Prof. Dr. Fatoş Tünay Yarman-Vural. Her patience, vision, sweet communication and friendly approach is the key reason to vitalize this work. It is an honour for me to share her knowledge, wisdom and humanity.

I am grateful to members of Image Processing and Pattern Recognition Laboratory at Department of Computer Engineering, I. Aykut Erdem, M. Erkut Erdem, Utku Erdoğan, Cüneyt Mertayak and Emre Akbaş. They always provided me with insightful comments and helpful hands.

I would like to express my heart-felt thanks to Fulya, my dear love. Without her unconditional love, joy and support, this thesis could not be completed.

Assoc. Prof. Dr. Göktürk Üçoluk and Dr. Onur Tolga Şehitoğlu are the people who give me the taste of evolutionary computing and unconditional support whenever I need. I feel deep gratitude for them.

I am also indebted to Prof. Dr. Volkan Atalay and Ömer Sinan Saraç for Bioinformatics data set and comments.

I also would like to thank my parents and younger brother for their complimentary love and support.

I owe İffet Yalabık and Prof. Dr. Neşe Yalabık very much for their hospitality, kindness and support.

Finally, I would like to dedicate this work particularly to H. Tahsin Yalabık. I will always remember...

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	v
DEDICATION	vi
ACKNOWLEDGMENTS	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	x
LIST OF FIGURES	xi
CHAPTER	
1 INTRODUCTION	1
1.1 Related Work	2
1.2 Motivation Behind the Proposed Systems	3
1.3 Thesis Outline	3
2 BACKGROUND FOR BOOSTING AND EVOLUTIONARY COMPUTING	4
2.1 Ensemble Learning: Multiple Classifier Systems	4
2.2 Boosting	6
2.3 Adaptive Boosting Theory	7
2.3.1 AdaBoost: Freund-Schapire	7
2.3.2 AdaBoost: Viola-Jones	10
2.3.3 Other Boosting Algorithms	14
2.4 Evolutionary Computing	20
2.4.1 Background for Evolutionary Computing	20
2.4.2 Genetic Algorithms	20
2.4.3 AdaBoost and Evolutionary Algorithms	22
2.5 Summary	23

3	A PATTERN CLASSIFICATION APPROACH BOOSTED WITH GENETIC ALGORITHMS	25
3.1	Motivation	25
3.2	Evolutionary Search and Boosting	27
3.2.1	Representation	27
3.2.2	Fitness Function	29
3.2.3	Initialization	32
3.3	Summary	33
4	EMPIRICAL STUDY	34
4.1	Data Sets	36
4.1.1	Bioinformatics Data Set and the Features	36
4.1.2	Face-Nonface Data Set and the Features	37
4.1.3	Feature Extraction	38
4.2	Weak Hypothesis	38
4.3	Experiment Results	40
4.3.1	Bioinformatics Data Set	40
4.3.2	Face-Nonface Data Set	45
4.4	Comparison of AdaBoost and Proposed Methods	49
4.4.1	Generalization vs. Overfitting	49
4.4.2	Dimension of The Feature Space	50
4.4.3	Complexities of Algorithms	50
5	CONCLUSIONS AND FUTURE DIRECTIONS	52
	REFERENCES	54

LIST OF TABLES

TABLES

Table 4.1	Notations and Abbreviations	35
Table 4.2	Distribution of Samples in Bioinformatics Data Set	36
Table 4.3	Distribution of Samples in Face-Nonface Data Set	37
Table 4.4	Convolution matrices of simple Haar wavelet transformation	38
Table 4.5	Experiment-1 on Bioinformatics Data Set	41
Table 4.6	Comparison of AdaBoost and fixed length encoding genetic algorithm in the training set	42
Table 4.7	Comparison of AdaBoost and fixed length encoding with initialization of unselected hypotheses	43
Table 4.8	Experiment-2 on Bioinformatics Data Set	44
Table 4.9	Experiments on Face-Nonface Data Set(Average) with classification rate based fitness function	45
Table 4.10	Experiments on Face-Nonface Data Set-1	46
Table 4.11	Experiments on Face-Nonface Data Set-2	47
Table 4.12	Experiments on Face-Nonface Data Set-3	48

LIST OF FIGURES

FIGURES

Figure 2.1	Bootstrapping	9
Figure 2.2	Integral Image	12
Figure 2.3	Haar Basis Functions	13
Figure 2.4	Overfitting	13
Figure 3.1	Fixed Length Encoding	28
Figure 3.2	Variable Length Encoding	29
Figure 4.1	Subset of images in the face-nonface data set	37
Figure 4.2	Simple Haar Wavelet Transformation	40
Figure 4.3	Overtraining	49

CHAPTER 1

INTRODUCTION

Classification has always been a challenging task throughout the human history. Categorizing phenomenon like good and evil, black and white, poor and rich, healthy and deadly, true and false, make human beings write books, do research and even go war. Everyday life introduces people an enormous number of classification problems. Some of these classification problems are solved at sensory in a moment like classification of objects, taste of a meat, smell of perfume, etc. On the other hand, some problems like classifying cancer cells, military targets, etc. are hard and sometimes impossible problems to be solved.

Literally, in Webster Dictionary[1], classification is defined as the grouping of things into classes or categories. These categories can change with respect to nature of the problem and time.

In the domain of computer vision and pattern recognition, Duda *et al.* [2] define pattern classification as the task of the classifier component is to use the feature vector provided by the feature extractor and to assign the object to a category. Basically, classification is the fundamental problem of pattern recognition.

The classical research on pattern classification gathers around designing single classifier for a specific problem domain. Many approaches have been proposed from simple methods to complex systems. Most of these methods perform well in small sets having limited diversity. But, these methods could not carry on the same performances in many real life problems. Another interesting point is that a set of classifiers with similar training performances may have different generalization powers. Slight or redundant changes in environment affect the performance and generalization ability of these classifiers drastically. In addition to these, “No free lunch theorem” claims that it is impossible to design a single classifier that performs well in all of the search

spaces of different classification tasks.

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 select or fuse best classifiers with respect to nature of the data, are produced under ensemble learning techniques: classifier fusion methods, termed classifier selection, bagging and boosting [3]. Learning any data with combining separate and independent classifiers is called ensemble learning.

1.1 Related Work

Among all ensemble learning techniques, *boosting* has been attracted more attention since mid-90's. Boosting is originally inspired from Valiant's PAC (Probably Approximately Correct) learning theory. Schapire *et al.* [4] define 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 previous round. Based on this fact, error is reduced at each turn of the algorithm. Freund and Schapire are the researchers that come up with the most successful and popular boosting algorithm called AdaBoost.

AdaBoost algorithm starts with initializing equal weights to each sample. At each stage of AdaBoost algorithm, moderate hypothesis **with lowest error** with respect to weight distribution is selected for final classifier, then weights of each samples are updated according to selected classifier as follows: weights of misclassified samples are increased, on the contrary, weights of correctly classified samples are decreased. Eventually, at each round, algorithm emphasizes misclassified samples like border points, outliers [4]. Selection of the hypothesis with lowest error at each step provides a greedy search over the search space. This work was later awarded with Gödel Prize¹ in 2003.

Viola *et al.* adapt AdaBoost algorithm with Haar-based features to make an excellent face detector that can work almost real-time (up to 5 frame per second) [5].

¹ The Gödel prize has rewarded outstanding journal articles in theoretical computer science since 1993. For details, <http://sigact.acm.org/prizes/godel/>

1.2 Motivation Behind the Proposed Systems

The main motivation in this thesis is the question that “Is it possible to find a more accurate or optimal solution without starting or selecting a weak hypothesis that does not have the lowest error among all hypotheses in boosting steps?” That is, whether presence of another search technique other than greedy search used in AdaBoost algorithm comes up to a more elegant final hypothesis with the same number of boosted classifiers or not. Li *et al.* propose a less greedy approach than the classical way of Schapire [4] by combining AdaBoost with floating search. This algorithm deletes some of the selected hypotheses if in absence of these hypothesis performance of final classifier increases [6].

In this study, an evolutionary search is proposed instead of greedy search that the classical AdaBoost applies. Genetic Algorithms, successively boost hypotheses which are coded as genes on the chromosomes (each chromosome represents a boosted final classifier). Many performance evaluating functions calculating fitness values are 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 problems with very large scales in terms of number of data and number of feature dimension is almost impractical in absence of the high performance computing utilities.

1.3 Thesis Outline

Organization of this thesis is as follows, in Chapter 2, a literature survey on boosting, AdaBoost and its variants are provided. Chapter 3 explains motivation of this work, proposed systems and tools to accomplish proposed stems. In Chapter 4, after explaining experimental setup, results of proposed techniques compared to results of AdaBoost. Finally, Chapter 5 concludes and gives future direction to this work and general boosting theory.

CHAPTER 2

BACKGROUND FOR BOOSTING AND EVOLUTIONARY COMPUTING

In this chapter, the fundamental concepts of boosting and evolutionary computing, which is necessary for the background of this study, will be presented. For this purpose, the general problem of pattern classification will be presented and some pioneer works in boosting history will be discussed. The major goal of this chapter is to give reader some fundamental material on boosting and lead to some problems that this work provides addresses partial solutions.

2.1 Ensemble Learning: Multiple Classifier Systems

Throughout the development of pattern recognition systems, many classifiers have been proposed under two fundamental views, namely, statistical (or decision theoretic) and syntactic (or structural). Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by probabilistic sources. Examples for statistical pattern recognition systems are linear classifiers such as Fishers Linear Discriminant, Naive Bayes Classifier, quadratic classifiers, k-nearest neighbor, boosting, decision trees, Neural Networks, Bayesian Networks, Support Vector Machines, Hidden Markov Models [2]. On the other hand, structural pattern recognition is originated from the structural interrelationships and coherency of features, items are presented pattern structures which can take into account more complex interrelationships between features than simple numerical feature vectors used in statistical classification [7]. Problems in syntactic pattern recognition are represented in strings of a formal language or graphs of relationships.

The learning algorithms, mentioned above, have successful applications and work

well under restricted data characteristics. However, most of these techniques offer single hypothesis over data distributions and basically suffer from three major problems, namely, statistical problem, computational problem and representational problem. Statistical problem arises when the search space is too large with respect to the amount of data. A learning algorithm that suffers from the statistical problem is called high **variance**. Sometimes learning algorithm can not guarantee to find the best hypothesis within the search space. In this case, the problem turns out to be a computational one. A method that arises computational problem considered as having **computational variance**. Finally, if the search space does not contain any hypotheses that are fairly good estimations to true solution, representational problem arises [8]. An algorithm having representational problem is said to have high **bias**.

Some real world problems having thousands dimensional feature vectors and thousands of data points decline the performance of classifiers that has impressive accuracies in toy data sets. This fact drives scientist to make up larger training sets. Notwithstanding, researchers have found themselves into a trade of that either relatively simpler methods have to be introduced in order to reduce computational complexity or high degree of computationally complex algorithms are produced to overcome high bias and variance. Due to sharp development in computational power and imaging technologies in last decades, more and more complex systems, even combination of various learning algorithms, are proposed to the literature. Another drawback commonly observed is that base methods (single classifier systems) show varying accuracy due to the changing characteristics of given data. For example, some classifiers can handle nonlinear data distributions, on the other hand, other algorithms are not capable of handling nonlinear data distribution, but very robust to outliers. These variations cause to emerge a new learning schema: Ensemble Learning combines multiple learning methods under the assumption that “two (or more) heads are better than one”. The decisions of multiple hypotheses are combined in ensemble learning to produce “better” results than the hypotheses alone.

Ensemble learning brings some partial solutions to the problems mentioned above. In statistical pattern recognition, single classifier systems can generate a suboptimal solution in consideration of insufficient amount of data. An optimal solution with small amount of training data turns out to be a local minima in the real world data. In such cases, ensemble learning has the ability to choose appropriate solution from

several different hypotheses that all give same accuracy on the training data or simple voting schema of ensemble learning algorithm proposes better solutions. In case of computational problem, forming the best hypothesis that fits the tens of thousands training samples with Neural Networks or decision tree methods is almost impractical process. For this reason, many heuristics are applied to lower dimensionality and complexity of the problem. Bagging and Boosting are two main approaches of Ensemble Learning that basically provides computationally simpler answers. Ensemble learning addresses solution to representation problem by spanning the search space with combining search spaces of base learners. Strategies like weighted voting, majority voting are able to form more accurate approximations [8].

Hereafter, Boosting, an Ensemble Learning Method, is discussed in various aspect.

2.2 Boosting

Boosting was first inspired from Valiant's Probably Approximately Correct Theory [9]. In this approach the learner gets samples that are classified according to a function from a certain class. The aim of the learner is to find a bounded approximation of the function with high probability. The learner must be able to learn the concept given any arbitrary approximation ratio, probability of success, or distribution of the training samples [9]. 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 [10]. It was 1989, when Schapire was able to come up with first polynomial time boosting algorithm [11], then Freund improved much more efficient algorithm that purifies Schapire's algorithm from the certain practical drawbacks [12]. The first experiments with these early boosting algorithms were carried out by Drucker, Schapire and Simard on an Optical Character Recognition task [13].

Schapire summarizes boosting in his famous work "The Boosting Approach to Machine Learning: An overview". In this way, boosting "refers to a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb" [14]. This approach has many connections with swarm robotics. Swarm robotics is a coordination of multi-robot

systems (boosting weak hypotheses) which consists of large numbers of relatively simple physical robots (rough and moderately inaccurate rules of thumbs : weak hypotheses). The goal of swarm robotics is to design a massive number of simple robots to obtain a collective behavior, just like, boosting is designed to combine many weak hypothesis to obtain a strong, more accurate, more general single hypothesis.

Boosting is defined formally as follows: Let, Z be a set of labelled N training examples. $Z = (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$, where x_i is the feature vector and y_i where $y_i = 0, 1$ is the label associated with instance of x_i . On each round $t = 1, 2, \dots, T$, a distribution D_t over a set of examples is devised and this distribution (selection of hypothesis left unspecified in boosting) forms a *weak hypothesis* h_i with low error ε_t with respect to D_t . Therefore, distribution D_t plays an important role in the distribution of next hypothesis to be selected in next round. After T rounds, booster combines the weak hypotheses into a single prediction rule: *strong hypothesis*.

Note that the bound on the performance of the final hypothesis, *strong hypothesis*, improves when *any* weak hypothesis is improved.

2.3 Adaptive Boosting Theory

In this section AdaBoost, one of the most important outcome of boosting theory is discussed. In the first part, advantages of AdaBoost is given with original procedure proposed by Freund-Schapire[12]. In the second part, Viola-Jones's version of AdaBoost is analyzed in detail to show the strength of the learning algorithm. Some proposed problems about AdaBoost are also discussed. In the final section, many boosting techniques that improve AdaBoost procedure are examined in detail.

2.3.1 AdaBoost: Freund-Schapire

Adaptive Boosting Theory is first introduced in 1997 by Freund and Schapire [4]. 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.

AdaBoost starts with weight initialization phase. The algorithm assigns equal weights w_i to all samples. On each round t_i , weak hypotheses are made up with respect to these weights and data distribution. Selection of a certain weak hypotheses is

Algorithm 1 ADABOOST(Freund and Schapire Version)

Require: sequence of N labelled examples $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$

distribution D over N examples

weak learning algorithm **WeakLearn**

integer T specifying number of iterations

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

Initialize the weight vector $w_i^1 = D(i)$ for $i = 1, 2, \dots, N$.

Repeat [1-5] for $t = 1, 2, \dots, T$

1: Set

$$p^t = \frac{w^t}{\sum_{i=1}^N w_i^t}$$

2: Call WeakLearn, providing it with the distribution p^t ; get back a hypothesis $h_t : X \rightarrow [0, 1]$.

3: Calculate the error of $h_t : \varepsilon_t = \sum_{i=1}^N p_i^t |h_t(x_i) - y_i|$.

4: Set $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.

5: Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$$

Output the hypothesis

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

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 low weights, misclassified samples have higher weights. As a result of this fact, misclassified samples are more effective than the correctly classified samples in selection of weak hypothesis on the next round. In other words, the weights tend to concentrate on hard examples. For this reason, hard samples (outliers, border points, noisy data) are closely involved with deciding strong classifier. This approach has many common aspects with bootstrapping method in pattern recognition. A *bootstrap* data set is created by randomly selecting from the training set with replacement. Bootstrap data set eliminates duplication of individual points, this leads to prevent aggregation of training samples, make training data more ho-

homogeneous and force learning algorithm to pay more attention on border point and outliers than ordinary points.

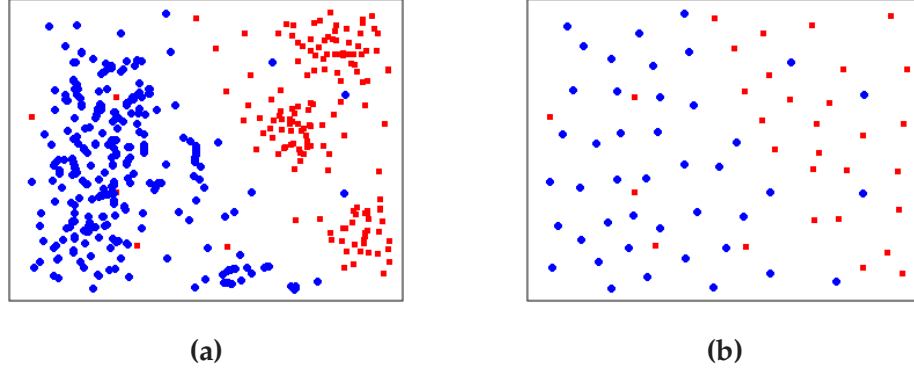


Figure 2.1: Bootstrapping **(a)** Original Data **(b)** A bootstrap data

AdaBoost is a good feature selector with minimizing the upper bound of the error. Boosted hypothesis selects only small set of hypothesis among large set of hypothesis. Suppose the weak learning algorithm WeakLearn, when called by AdaBoost, generates a set of hypotheses with errors $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_t$. Then the error $\varepsilon = \Pr_{i \sim D}[h_f(x_i) \neq y_i]$ 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 WeakLearn 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 represent 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 suffers to find optimal solutions and sticks to suboptimal solutions, because heuristic based search methods prune the search space to make algorithms pay more attention to find a better solutions and sometimes they cut the paths that lead optimal solutions. AdaBoost is one of the heuristic based methods. Selecting correct or optimal over entire data is a *NP-Complete* process. Heuristic that is selection of hypothesis with the lowest error enables AdaBoost to work on higher (180,000) dimensional feature spaces and large amount (20,000) of data samples [5]. Although AdaBoost performs a heuristic based 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 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. Testing of the final classifier is based on only calculation of equation below.

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

AdaBoost almost 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. These methods can be any techniques from simple threshold classifiers to complex systems like Support Vector Machines or Neural Networks.

2.3.2 AdaBoost: Viola-Jones

AdaBoost has been implemented in wide range of application areas with a great varieties [13], [15], [16]. Among experiments, Viola *et al.* application of AdaBoost on face detection problem has been attracted much attention. Viola *et al.* achieved one of the best detector in face detection literature [5].

The first contribution of Viola's *et al.* work is the creation of integral image. Integral image at location x, y contains sum of the pixels both upper and left of x, y , inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Algorithm 2 ADABOOST(Viola and Jones Version)

Require: training images $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ 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 and l are the number of negatives and positives respectively.

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

- 1: Normalize weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

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

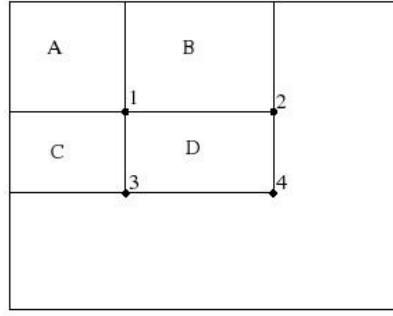


Figure 2.2: Integral Image: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A . The value at location 2 is $A + B$, at location 3 is $A + C$ and at location 4 is $A + B + C + D$. The sum within D can be computed as $I_4 + I_1 - (I_2 + I_3)$.

Complexity Analysis: Let, N be the number of training samples and H be the number of hypotheses to be chosen on each round. Initialization is a $\mathcal{O}(N)$ operation, recursive step-1 is an $\mathcal{O}(N)$ time, recursive step-2 requires to train H classifiers to compute respective errors. Simple threshold based classifiers are used, there are $\mathcal{O}(N)$ methods to find optimum threshold method. For that reason, step-2 complexity is $\mathcal{O}(HN)$. Step-3 and step-4 are both $\mathcal{O}(N)$ time algorithms. Recursive steps are repeated T times. Therefore, complexity of training AdaBoost is $\mathcal{O}(HNT)$. To conclude, the complexity of AdaBoost is directly proportional to complexity of weak classifier.

Integral image is used for fast computation of simple Haar based features. Figure 2.2 shows the usage of integral image. Any Haar coefficient of any N -rectangle feature can easily calculated by few number of summation and subtraction over integral image. Viola-Jones extracted 180,000 simple features from 24×24 face images. Figure 2.3 depicts Haar basis function used by Viola *et al.* [5], [17], [18].

Viola *et al.* reported that a frontal face classifier constructed from 200 features yields a detection rate 95% with a false positive rate 1 in 14084. The speed of final classifier on a 700 MHz Pentium III processor scanning 384×288 pixel image is about 0.64 seconds [5].

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

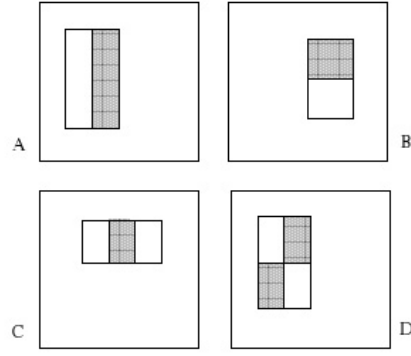


Figure 2.3: Haar Basis Functions: Example rectangle features are shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of the pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

strong match between problem and model is obtained.

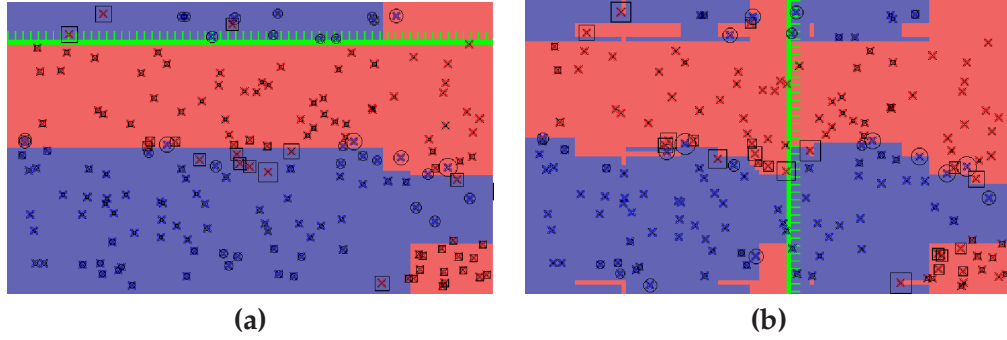


Figure 2.4: Overfitting **(a)** Final classifier over toy training data when $T = 20$ **(b)** Final classifier over toy training data when $T = 300$

When T gets larger and larger, overfitting problem is observed [19]. Simulation experiments in Figure 2.4 that are accomplished over toy data set shows that overfitting 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.

Hypothesis selection on each round depends weight distributions of training data with respect to weak classifiers in data. Based on this fact, selections on very first rounds determines the general behavior of the algorithm, if the emphasis is on learning positive samples, final classifier ends up with high detection rate but also relatively high false alarm rate, if the emphasis is on learning negative samples, final

classifier ends up with low false alarm rate but also relatively low detection rate.

2.3.3 Other Boosting Algorithms

Since Freund and Schapire first proposed AdaBoost in 1995, many research such as [20], [6], [17], [18], [21], [22] are accomplished to fix problems of AdaBoost mentioned in previous sections. In this section, brief summary of these algorithms is provided.

- **FloatBoost:** FloatBoost was first proposed in 2002 by S. Li, Z. Zhang, H. Shum and H. Zhang [6]. Algorithm is almost the same in the initialization and feature selection stage. However, a backtracking mechanism named conditional exclusion is proposed to remove unfavorable weak classifiers from existing classifiers to achieve a low error rate. The idea of backtracking originally comes from Floating Search. The floating search procedure allows some number of backtracking steps to be checked before a fixed beforehand. Due to extension of floating search, algorithm is named as FloatBoost. Li *et al.* reported that FloatBoost comes up with less number of weak classifiers building strong classifiers at the same error rates or lower error rates with same number of weak classifiers. For example, on the test set, by combining 1000 weak classifiers, the false alarm of FloatBoost is 0.427 versus 0.485 of AdaBoost. The lowest test error for AdaBoost is 0.481 with 800 weak classifiers, whereas FloatBoost needs only 230 weak classifiers to achieve the same performance[6].

Initialization and first three steps of the algorithm is the same as the classical AdaBoost, at step-4 a weak classifier is removed from selected classifiers and error is recalculated. If the new error value is not smaller than previous one, a previously removed classifier is randomly selected and re-included to strong classifier. Otherwise, new strong classifier is one left weak classifier less than previous classifier.

One drawback of this approach is increased time complexity training stage, Li reports that FloatBoost is 5 times slower than AdaBoost in term of training time[6].

Algorithm 3 FLOATBOOST

Require: training images $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ where $y_i = -1, 1$ 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 $y_i = -1, 1$ respectively, where m and l are the number of negatives and positives respectively.

set the maximum number M_{max} of weak classifiers

set the error rate $\varepsilon(H_M)$, and acceptance threshold e^*

$M = 0, H = \{\}$

- 1: $M \leftarrow M + 1$
- 2: Choose the h_M according to $\varepsilon_t = \frac{1}{2} \log \frac{P(y=+1|x, w^{M-1})}{P(y=-1|x, w^{M-1})}$.
- 3: Update weights $w_i^{(M)} \leftarrow \exp[-y_i H_M(x_i)]$ and normalize to $\sum_i w_i^{(M)} = 1$
- 4: $H_M = H_{M-1} \cup h_M$; If $\varepsilon_M^{min} > \varepsilon(H_M)$, then $\varepsilon_M^{min} = \varepsilon(H_M)$;
- 5: $h' = \operatorname{argmin}_h \varepsilon(H_M - h)$
- 6: If $\varepsilon(H_M - h) < \varepsilon_{M-1}^{min}$, then $H_{M-1} = H_M - h'$; $\varepsilon_{M-1}^{min} = \varepsilon(H_M - h')$; $M = M - 1$;
 $H_M = \sum_h h$; goto step 5.
- 7: If $M = M_{max}$ or $J(H_M) < J^*$, then finalize.
- 8: $w_i^{(M)} \leftarrow \exp[-y_i H_M(x_i)]$; goto step 1.

The final strong classifier is:

$$H(x) = \operatorname{sign}\left[\sum_{h(x)}^{H_M} h(x)\right]$$

-
- **LogitBoost:** LogitBoost is an adaptive algorithm for fitting a additive logistic regression model by step-wise optimization of the Bernoulli log-likelihood during the hypothesis selection. Due to additive logistic regression model, algorithm is named LogitBoost. The LogitBoost algorithm tries to fit a regression model with respect to working responses and weights [17]. It is claimed that performance of LogitBoost is usually better on noisy data and or when there are misspecification or inhomogeneities of the class labels in the training data [21]. Friedman *et al.* complete experiments that training error of LogitBoost is 0.096 in satellite images where AdaBoost performs 0.106 [17].

Algorithm 4 LOGITBOOST

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

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

Initialize weights $w_{1,i} = \frac{1}{n}$ for $y_i = -1, 1$ respectively, where n is the number of training samples and $p(x_i) = \frac{1}{2}$

Repeat [1-3] for $t = 1, 2, \dots, T$:

- 1: Compute the working response and weights

$$z_i = \frac{y_i^* - p(x_i)}{(p(x_i)(1 - p(x_i)))} w_i = p(x_i)(1 - p(x_i))$$

- 2: Fit the function $h_m(x)$ by a weighted least-squares regression of z_i to x_i using weights w_i .

- 3: Update $H(x) \leftarrow H(x) + \frac{1}{2}h_m(x)$ and $p(x) \leftarrow \frac{e^{H(x)}}{e^{H(x)} + e^{-H(x)}}$.

The final strong classifier is:

$$\text{sign}[H(x)] = \text{sign}\left[\sum_{t=1}^T h_t(x)\right]$$

- **GentleBoost:** In the original AdaBoost algorithm, the hypotheses with lowest error with respect to weight distribution is selected on each round, t by maximizing likelihood. Name GentleBoost comes from the fact that in this algorithm, solution is explored with a “gentler” version that instead takes adaptive Newton steps much like the LogitBoost algorithm just described. GentleBoost can be considered as an application of of Newton-Raphson algorithm optimization algorithm to minimize chi-square error. As the number of examples are increased, minimizing the chi-square error converges to maximizing the likelihood. However, when there exists insufficient number of samples, chi-square estimators are more accurate than maximum likelihood estimators [17], [18]. Friedman *et al.* report that GentleBoost achieves 0.036 training error where AdaBoost achieves 0.037 in breast data set [17].

Algorithm 5 GENTLEBOOST

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

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

Initialize weights $w_{1,i} = \frac{1}{n}$ for $y_i = -1, 1$ respectively, where n is the number of training samples.

Repeat [1-3] for $t = 1, 2, \dots, T$:

- 1: Fit the regression function $h_t(x)$ by weighted least-squares of y_i to x_i with weights w_i .
- 2: Update $H(x) \leftarrow H(x) + H_t(x)$
- 3: Update $w_i \leftarrow w_i e^{-y_i h_t(x_i)}$ and re-normalize.

The final strong classifier is:

$$\text{sign}[H(x)] = \text{sign}\left[\sum_{t=1}^T h_t(x)\right]$$

- **MadaBoost:** MadaBoost stands for “A Modification of AdaBoost”, in presence of outliers and noisy data, misclassified samples have more influence on selection of weak hypothesis. After a certain t some misclassified samples have enormously higher weights than other training samples. Based on this fact, MadaBoost introduces a saturation bound $D_0(x)$ that weight $w_i(x)$ can not be increased beyond $D_0(x)$ [22]. No experimental results are being reported by authors.

Algorithm 6 MADABOOST

Require: training images $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ 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 and l are the number of negatives and positives respectively.

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

1: Normalize weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

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:

$$w_{t+1,i} = \begin{cases} D_0(x) \times \prod_{1 \leq i \leq t+1} \beta_i^{\text{cons}(h_i, x)} & \text{if } \prod_{1 \leq i \leq t+1} \beta_i^{\text{cons}(h_i, x)} < 1 \\ D_0(x) & \text{otherwise} \end{cases}$$

where $D_0(x)$ is the initial weight.

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

- **BrownBoost:** BrownBoost is another boosting algorithm that can handle noisy data sets. Freund *et al.* claim that “The method used for making boost-by-majority adaptive is to consider the limit in which each of the boosting iterations makes an infinitesimally small contribution to the process as a whole. This limit can be modeled using the differential equations that govern Brownian motion”. BrownBoost is based on finding solutions to these differential equations. Original version of AdaBoost does not perform well in noisy data sets, that is the result of behavior that AdaBoost concentrates more on misclas-

sified samples at each round. In contrast, BrownBoost effectively “gives up” on examples that are repeatedly misclassified. In preliminary experimental results, BrownBoost outperformed AdaBoost; however, LogitBoost performed as well as BrownBoost [23].

Algorithm 7 BROWNBOOST

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

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

Initialize $s = c, r_1(x_j) = 0$

Repeat [1-4] until $s > 0$:

1: Set weights of each example,

$$W_i(x_j) = \exp^{-\frac{(r_i(x_j)+s)^2}{c}}$$

where $r_i(x_j)$ is the margin of of example.

2: Find a classifier $h : X \rightarrow -1, +1$ such that

$$\sum_j W_i(x_j) h(x_j) y_j > 0$$

3: Find values α^*, t^* that satisfy the differential equation.

$$\frac{dt}{d\alpha} = \sum_j \exp^{-\frac{(r_i(x_j)+\alpha h(x_j)y_j+s-t)^2}{c}} = 0$$

4: Update margins for each sample: $r_{i+1}(x_j) = r(x_j) + \alpha h(x_j) y_j$

5: Update time remaining: $s = s - t$

The final strong classifier is:

$$H(x) = \text{sign}(\sum_i \alpha_i h_i(x))$$

2.4 Evolutionary Computing

In this thesis, evolutionary search is applied to address partial solutions to the problems in AdaBoost which is explained in previous chapters. In this section, firstly a brief introduction to evolutionary computing is given to prepare reader for the algorithms to be proposed in the next chapter. In Chapter 3 the proposed methods that combine AdaBoost and evolutionary search is discussed.

2.4.1 Background for Evolutionary Computing

Every organism in the universe has a set of rules, a blueprint of its architecture describing how that organism built up from various sizes of building blocks of life. These rules are encoded in the genes of *Deoxyribonucleic acid*, shortly *DNA*. DNA is made up of chromosomes as chromosomes are made up of genes. Genetic makeup of an individual organism is called *genotype*. On the other hand, *phenotype* of an individual organism describes one of traits or characteristics that is measurable and that is expressed in only a subset of individuals within that population. Combination of genes represents genotype, however, features like “blue eyes”, “brown hair” is the characteristics that phenotype represents.

In biology, the change in a population’s inherited trait from generation to generation is called *evolution*. As it is mentioned before, traits are the expression of genes that are produced, copied and passed from ancestor to offsprings. Due to reproduction, different combinations of genes are observed, this fact reinforces diversity among population. Another factor that cause diversity is *mutation*. Mutations are the changes to the base pair sequences of genetic material. Most of the mutations are the copying error observed during cell division although cell division is an accurate process. Evolution occurs in three ways, natural selection which measures the difference in reproductive success, genetic drift indicating the statistical effect that decides change of survival alleles and finally gene flow modelling the movement of genes between populations) [24].

2.4.2 Genetic Algorithms

In computer science, evolutionary computation is a subset of Artificial Intelligence that involves combinatorial optimization problems. Genetic algorithm is a subfield

of evolutionary computation, generally involves with implementation of evolution steps such as reproduction, mutation, recombination, natural selection and survival of fittest. These method are inspired from Biology mentioned in previous section.

Genetic algorithms are search techniques to find optimal or approximate solutions to optimization or search problems. Implementation of these algorithms are the computer simulations which includes abstract representations (chromosomes, genotype of genes) of candidate solutions (individuals, phenotype), mutation strategies, reproduction methods and fitness functions that decides which individual survives to next generation.

Biogenetics, computer science, engineering, economics, chemistry, manufacturing, mathematics and physics are the main application areas of Genetic Algorithms.

Two important building blocks for a genetic algorithm is:

- *A genetic representation of the solution:* Representation of a solution can be an array of bits, integers, floating point numbers or strings, trees, etc. Each representation is designed with respect to spirit of the problem. Encoding represents basic composition of a solution. Composition of basic structures(bits,integers,...) forms an individual like, combination of nucleotides based forms DNA in biology. Representation is the genotype of the solution.
- *A fitness function to evaluate performance of the solution:* Fitness function decides which solution achieves higher performance as natural selection determines which individual survive with respect to environmental factors.

Simple Genetic Algorithm is as follows:

Algorithm 8 SIMPLE GENETIC ALGORITHM

Require: Θ_c : crossover probability, Θ_m : mutation probability

Ensure: A solution or a population of solution after termination condition

Let I_λ : parent population, I_μ : offspring population, p_t : population at generation t .

Initialize: $t \leftarrow 0, p_0 = \text{randompopulation}()$

Repeat [1-5] until termination condition is satisfied Termination condition can be a fixed number of iterations or local optima.

- 1: $I_\mu \leftarrow \text{selecttomate}(p_t, \lambda_c)$: Sexual Selection
- 2: $I_k \leftarrow \text{crossover}(I_\mu)$: Crossover
- 3: $I_\lambda \leftarrow \text{mutate}(I_k, \Theta_m)$: Mutation
- 4: $p_{t+1} \leftarrow \text{selecttosurvive}(I_\mu, I_\lambda)$: Ecological Selection, Elitism
- 5: $t \leftarrow t + 1$

Output: Final population P_T and fittest individual of all populations F_T

Genetic algorithms work in many situations as “The Schema Theorem”. Theorem proposes that “Short, low-order, above average fitness schemata receive exponentially increasing trails in subsequent generations”.

2.4.3 AdaBoost and Evolutionary Algorithms

Increasing popularity of boosting techniques, especially AdaBoost, inspires many researchers to find algorithms that outperforms or improves the classical AdaBoost. In evolutionary computing, there also exists some works on boosting [25], [26], [27]. In this section, a brief descriptions of these methods are discussed.

- AdaBoost algorithm searches exhaustively to find a hypotheses with the lowest training error at each round. Treptow *et al.* propose an evolutionary search instead of exhaustive search [25]. Proposed system is suitable for the experiments that have sufficiently large datasets like Viola *et al.* use [5]. Types and coordinates of the Haar based features in Figure 2.3 are coded in chromosomes as genes. Thresholds on 1D histograms based on Haar features are the weak

learners of proposed algorithm. Training error is used as fitness function. Training time for AdaBoost the reported is 42040 seconds, whereas their proposed system runs 10868 seconds on training set, Treptow *et al.* also report slight classification performance [25].

- In genetic algorithms, when number of generations to be evolved and number of individuals in the population gets larger and larger, running these algorithms takes more time. Liu *et al.* [26] propose an boosting method to combine genetic classifiers that are trained with sufficiently and practically less number of individuals and generations. Final classifiers is the boosted genetic classifiers. Idea is to build up a genetic classifier from genetic classifiers with small scaled parameters. This idea has analogy with original boosting that combines weak classifiers to build up a strong classifier. It is reported that 93% classification rate achieves by boosting 3 genetic classifiers 25% less generations and 50% less individuals. On the other hand, single genetic classifier has 90% classification rate [26].
- Another approach proposed by Wang *et al.* is to ensemble base classifiers in such a form that initial weights of samples are calculated by a genetic algorithm [27]. Each chromosomes are the strings of initial weights of the base classifier, as the population evolves discriminative samples such as border points gets higher weight values than common points. Based on this fact, selected hypothesis concentrates more on discriminative points. This approach has many similarities like support vector points in this context. Fitness function is again classification rate over training samples. Wang *et al.* report 93.5% detection rate with 41 false alarms at CMU Face Set¹, whereas, AdaBoost has 94.1% detection rate with 32 false alarms [27].

2.5 Summary

AdaBoost is one of the most popular and successful boosting algorithms that can handle data with any size and dimension. Simple decision making strategy enables algorithm to be integrated in real time systems. AdaBoost provides a linear classifier with a good generalization ability. Despite of these elegant properties, AdaBoost still

¹ CMU Face set is one of the widely face detection benchmark set created by Rowley *et al.* [28].

has some weaknesses. In literature, many researchers have accomplished to find partial solutions to problems addressed. In the next chapter, proposed methods are provided in detail.

CHAPTER 3

A PATTERN CLASSIFICATION APPROACH BOOSTED WITH GENETIC ALGORITHMS

In Chapter 2, boosting algorithms, especially AdaBoost, is discussed in various aspects. Successive error reduction at each stage, ability of dimensionality reduction and straight forward heuristic that make algorithm to classify in remarkably large data sets [5] are proofs for strength of AdaBoost. The original AdaBoost provides a linear classifier with all desirable properties. On the other hand, AdaBoost does not guarantee the optimal solution and can overfit in the presence of noise.

In this chapter, proposed systems which try to answer the questions that “How can AdaBoost be improved without corrupting algorithm’s good properties?” or “Is there another boosting schema that can be alternative to original AdaBoost algorithm?” are described in detail.

Organization of this chapter is as follows: first of all, motivation and inspiration behind this thesis are described, then representation and fitness functions that are necessary in evolutionary computing are discussed and proposed initialization schemas are provided in detail.

3.1 Motivation

Among the boosting techniques described in Chapter 2, AdaBoost draws more attention and praise. In 2003, Freund and Schapire are rewarded with Gödel Price.

AdaBoost is able to combine rough, moderate learning algorithms that perform slightly better than random classifier to generate more accurate, strong learning algorithm. In order to combine weak learners, at every step of the algorithm, it chooses a learner 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 *ONLY* the best classifiers at each steps leads algorithm to find optimum solution or not. Li *et al.* attack this question with inserting a floating search after boosting steps to perform less greedy search in their respectful work FloatBoost [6]. FloatBoost combines floating search with boosting to crop out some hypotheses chosen at some certain steps of the algorithm that reduce performance of final classifier. At each round of Algorithm , 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 that this thesis 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 constraint among all hypotheses in boosting steps?”. In other words, this work claims that selection of not best hypotheses at each round leads better solutions compared with heuristic that chooses hypothesis with lowest error. Another fulcrum of this approach is the experiments of Treptow *et al.* [25]. Their system searches best classifier in hypothesis space with a genetic algorithm and boosts these classifiers, they achieve better classification rates with same number of boosted classifiers. Obviously genetic algorithm finds a classifier other than the classifier with the lowest training error. If evolutionary search finds the lowest one, Treptow’s algorithm can not outperform the original AdaBoost algorithm.

Li *et al.* and Treptow *et al.* practically prove that it is possible to find a more accurate solution without heuristic that is used by AdaBoost. At this point, this thesis 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 in the Algorithm 1. 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, \dots , 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 impossible to search exhaustively in H^T sized-space to find such a permutation that outperforms the original AdaBoost algorithm and also using another heuristic to form such a permutation suffers from the problem mentioned above. Based on this fact, evolutionary search is proposed in order to greedy search that AdaBoost uses. Due to fitness function genetic algorithm is another search algorithm that is more general than greedy search and more restricted than exhaustive search. Mutation and crossover make genetic algorithm to search randomly in search space. However, selection of appropriate fitness function provides a more restricted search. For that reason, evolutionary search can be considered as a middle layer between greedy search and exhaustive search.

In this work, genetic algorithms are used as evolutionary search technique to propose a search strategy among the all possible combination and permutations of possible weak classifiers that outperform greedy search. 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 closer to exact solutions rather 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.

3.2 Evolutionary Search and Boosting

As mentioned in section 2.4.2, representation and fitness function are the two fundamental requirements to form a genetic algorithm. In section 3.1, finding a more accurate strong classifier problem is redefined as finding a permutation in H^T dimensional search space.

3.2.1 Representation

In order to use genetic algorithms, every individual has to be successfully represented. In connection with biology, representation is the genotype of a chromosome that encodes appearance, behavior and physical capabilities of individuals. Genetic representation is a way of expressing solutions of the problems in evolutionary computing context. A good design of a genetic representation make genetic algorithm

to converge rapidly and make evolutionary search more intelligent. Representations can be array of bits, arrays of integers and even trees. Basic elements of these structures are called genes. Each gene is traditionally a bit, but in more complex representations, it can be an integer, floating point or character. A good representation has to be closed under genetic operations and preserve locality. Representations in this thesis can be described as follows.

Every hypothesis that AdaBoost select at each round t_i is an element of hypotheses sets, H . That is $h_i = h_1, h_2, \dots, h_M$, where M is the number of hypotheses and $t_i = t_1, t_2, \dots, t_T$ 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 thesis, a hypothesis is the adaptive threshold on 1D histograms generated by a pre-defined 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 $i, j = 1, 2, \dots, T$. Fixed length encoding has only one drawback that length of the encoding is given as parameter to learning algorithm. It is not always possible to predict suitable length in advance. For that reason, some of the training set can be used as the validation set to decide suitable parameter for length of coding.

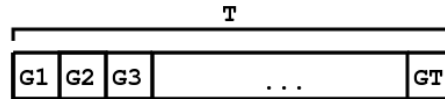


Figure 3.1: Length of the encoding is T which is the number of rounds of boosting steps.

- **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. This representation proposes a parameterless classification. Variable length coding provides a parameterless search methodology because optimum length of coding is decided by evolutionary search.

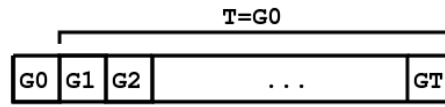


Figure 3.2: Length of this encoding is $T + 1$ that is the number of rounds of boosting steps.

3.2.2 Fitness Function

Fitness function is an objective function that quantifies the quality of individuals in population. In other words, fitness function is a transition from genotype (representation) to phenotype (solution). An eligible fitness function revolves around algorithms goal, execution speed of a fitness algorithm is also important. Higher valued fitness functions come up with more accurate solutions. It is also important that well-formed fitness functions make genetic algorithm to converge more rapidly. In case of classification problems, classification rate based fitness function is more suitable.

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

Algorithm 9 FITNESS FUNCTION

Require: training images $TR = (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ 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 and 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 $t = 1, 2, \dots, T$:

1: Normalize weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

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 $w_t, \epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.

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 $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

Form final strong classifier

Evaluate a function with respect to final strong classifier.

$$f(I_i) = FFUNCTION(I_i, h(x), TR)$$

In this study, two approaches are used to define the fittest function $FFUNCTION(I_i, h(x), TR)$, first one is based on the detection and false alarm rates of the classifier.

- Mathematically, form the final classifier as follows

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

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

- Compute detection rate DR and false alarm rate FA of final classifier with respect to training set TR .

$$f(I_i) = FFUNCTION(I_i, h(x), TR) = \beta_1 DR - \beta_2 FA$$

where $DR = \frac{\text{number of true positives}}{\text{number of positive samples}}$, $FA = \frac{\text{number of false positives}}{\text{number of negative samples}}$

and $\beta_1 + \beta_2 = 1$.

As mentioned in section 3.2.1, representation are fixed-length or variable-length permutations over H^T dimensional space. Quantities of representation qualities are easily computed in boosting context. Another ability of fitness function above is that β_1 and β_2 values are configurable and these parameters provides researchers to form a final classifier in nature of application. For example, in case of such applications like cancer cell detection, detection rate is more important than false alarm rate. Increasing β_1 makes algorithm concentrate more on increasing detection rate. Another point that reader has to pay more attention on is the computational complexity of the fitness function. Although it is strongly advised that fitness function has to be as simple as possible, fitness function has $\mathcal{O}(NT)$ time complexity where $\mathcal{O}(N)$ is the complexity of calculating error of a certain classifier and T is the number of boosting steps or length of the representation. This fitness function schema is the most suitable fitness evaluation method for boosting.

The second approach to define fitness function is based on Receiver Operating Characteristic(ROC) Analysis, confidence rates of each training samples can be calculated by formula below.

- Mathematically, form the final classifier as follows

$$h(x) = \sum_{t=1}^T \alpha_t h_t(x) - \frac{1}{2} \sum_{t=1}^T \alpha_t$$

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

- Compute confidence rates of each sample in training set TR with final classifier.
- Form Receiver Operating Characteristic curve ROC with respect to confidence rates.

$$f(I_i) = FFUNCTION(I_i, h(x), TR) = AREA_UNDER_CURVE(TR_{ROC})$$

This approach has several drawbacks. First one is the calculation of area under roc curve has an extra computational load. Another drawback is that ROC curve analysis is more challenging problem. Because requires a projection to M dimensional feature space to one dimensional space. After projection, detection rates are calculated with respect to false alarm rates. Thus, positions of points in one dimensional position is more important than position with respect to certain threshold.

3.2.3 Initialization

Random Initialization: Genetic algorithms are more general heuristic methods. Randomization and mutations are the fundamental factors that make genetic algorithm perform much like exhaustive search. Many genetic algorithms start with a purely random population. Random initialization is a method that is widely used in experiments in Chapter 4.

Partial Initialization: Viola *et al.* report that, it is observed that AdaBoost has a good classification ability in even very first selections of weak hypothesis. 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, more 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 different from hypothesis with lowest

error. These hypotheses are injected to genetic pool as initialization step. In initialization phase, first individual is initialized by hypotheses with lowest errors, second individual is initialized by hypotheses with second lowest errors, and so on. This initialization schema is expected to improve performance of AdaBoost.

3.3 Summary

In this chapter, evolutionary search by using genetic algorithms is proposed instead of heuristic based search that the classical AdaBoost uses. This approach sacrifices some of the best solutions in prior or middle selections of strong classifier and tries to find better solutions. With variable length encoding, suitable number of rounds T is selected by genetic algorithm. Initialization schema improves the classical AdaBoost in terms of classification ability.

CHAPTER 4

EMPIRICAL STUDY

In this chapter, the experimental setup for the methods proposed in Chapter 3, in which evolutionary search is combined with boosting, is presented. An empirical analysis is conducted to demonstrate strengths and weaknesses of the proposed algorithms compared to the classical AdaBoost. In experiments, performance of algorithms are discussed for the binary classification problem.

For this purpose, two data sets are used as training and testings environments. The first data set is a face-nonface dataset, which is used by Viola *et al.* [5] during training of AdaBoost. Face detection is one of the benchmark problems in computer vision literature [27], [5], [6], [28], [29]. An important characteristic of this data set is that number training samples are relatively high compared to the dimension of feature space. Other data set, which is from Bioinformatics area, composed of two classes, nucleic proteins and cytoplasmic proteins. Biologically, classification of this data set is harder because there exists some proteins that can be travel from cytoplasm to nucleus or vice versa.

All experiments conducted in this work are conducted on GNU/Linux 2.6 and GNU/Linux 2.4, 2.4 GHz Dual core and Pentium-4 3.0 GHz platforms. *GAlib* [30] is used as the genetic algorithms library. C++ and C programming languages are the cores of implementation of proposed algorithms. *MatLab* language is, also, used in feature extraction phase of the implementation.

Organization of this chapter is as follows, in the first section, general information about data sets and feature extraction methods used in experimental setup is discussed. Then, weak hypotheses used in this thesis is represented. Finally, experimental results with their discussion are provided.

Table 4.1: Notations and Abbreviations

<i>Basic Notations</i>	
I_i :	i_{th} individual
N :	number of training samples
N_p :	number of positive training samples
N_n :	number of negative training samples
P_{mut} :	mutation Probability
P_{cross} :	crossover Probability
NG :	number of Generations
NP :	number of Individuals in the population
G_i :	i_{th} generation of the population
$F(I_i, G_j)$:	fitness value of i_{th} individual in j_{th} generation
<i>Classification Methods</i>	
AB :	AdaBoost
EB :	evolutionary boosting with fixed-sized encoding
VEB :	evolutionary boosting with variable-sized encoding
SEB :	evolutionary boosting initialized with partial results of AdaBoost and with fixed-sized encoding
$MSEB$:	evolutionary boosting with initialized population of ranked classifiers of AdaBoost
ALG_i	an algorithm ALG is boosted with i number of weak hypothesis where $ALG = AB, EB, VEB, SEB$
<i>Fitness Functions</i>	
FF_{CR} :	classification rate fitness evaluation schema
FF_{ROC} :	area under ROC curve fitness evaluation schema
<i>Evaluation Abbreviations</i>	
DR :	detection rate
FA :	false alarm
CR :	classification rate
ROC :	receiver operating characteristics curve
A_{ROC} :	area under receiver operating characteristics curve
$AV_j(I)$:	average fitness value in i_{th} generation
B_i :	best fitness value in i_{th} generation

Table 4.2: Distribution of Samples in Bioinformatics Data Set

	<i>Training Data</i>	<i>Testing Data</i>
Nuclear Proteins	1335	890
Cytoplasmic Proteins	1335	445

4.1 Data Sets

In this work, two data sets are extensively used to evaluate and compare performances of proposed systems and the classical AdaBoost. In this section, general information of data sets, feature extraction techniques and weak classifiers used in these data sets are provided.

4.1.1 Bioinformatics Data Set and the Features

Bioinformatics data set is composed of features from two kinds of proteins, nuclear proteins and cytoplasmic proteins. Throughout this work, nuclear proteins are used as positive samples and cytoplasmic proteins are used as negative samples. Organization of data set is as follows:

Construction of fixed dimensional vectors requires two steps. At the first step, a subsequent profile map is constructed. All possible subsequences of a given length are extracted from positive training sequences. These subsequences are clustered as similar subsequences fall into same clusters. Then a probabilistic profile for each cluster is generated. At the second step, each subsequence of proteins that occupy a place in feature space as subsequence distributions over generated profile maps is compared with each probabilistic profile and a probability is calculated as,

$$P(x|PP_k) = \sum_{i=0}^l PP_k(i, x(i)),$$

where l is the length of subsequences and x is a single subsequence. The value for k^{th} dimension of the feature vector is set to

$$P_{\max}(S) = \max_{x_i} P(x_i|PP_k)$$

probability of highest scoring subsequence of protein S on probabilistic profile PP_k .



Figure 4.1: Subset of images in face-nonface data set

4.1.2 Face-Nonface Data Set and the Features

Face and nonface data set is the same training set as the one used by Viola *et al.* used to train the original AdaBoost algorithm. All of these images are cropped, scaled to 32×32 resolution and histogram equalized to enhance from certain undesirable lighting conditions. Mirror images of all face images are taken in order to increase the amount of face images. Entire data is split up into two data set for training and testing purposes.

Table 4.3: Distribution of Samples in Face-Nonface Data Set

	<i>Training Data</i>	<i>Testing Data</i>
Face Images	4916	4916
Nonface Images	4916	4916

Ten randomly generated training and testing sets are produced to evaluate the performances of proposed methods and compare them with the classical AdaBoost algorithm.

Table 4.4: Convolution matrices of simple Haar wavelet transformation

$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{4}$	$\frac{1}{4}$	$-\frac{1}{2}$	$-\frac{1}{2}$
(a) Averaging		(b) Alpha	
$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
(c) Beta		(d) Gamma	

4.1.3 Feature Extraction

Simple Haar wavelet transformation of the images are used as features of face-nonface data set. Haar wavelet transformation provides multi-resolutional, simple intensity difference in based features for image compression, noise removal and filtering.

At each stage, upper-left quarter of the image holds down-sampled version of original image (averaging), upper-right quarter of the image holds vertical intensity changes, lower-left quarter holds horizontal intensity changes and lower-right quarter holds diagonal intensity changes. Computed intensity differences is simple 2×2 convolution matrices emphasizing changes of intensity values in horizontal, vertical and diagonal fashion [31].

Computational load of extracting haar features is negligible compared to complexities of learning algorithms. Application of simple Haar wavelet transformation produces N^2 wavelet coefficients from $N \times N$ image. Thus, 32×32 face and nonface images in this data set have 1024 dimensional feature vectors.

4.2 Weak Hypothesis

Throughout the experiments thresholding technique over 1D histograms is extensively used. Finding an optimum threshold over one dimensional data is reduced to $\mathcal{O}(N)$ with a bucket sort approach. First of all, histograms of positive and negative samples are formed, then cumulative histograms are calculated. Finally, the best detection and false alarm rate are computed by traversing on cumulative histograms of positive and negative samples at the same time.

Algorithm 10 SIMPLE HAAR WAVELET DECOMPOSITION

Require: $n \times n$ square image Im

intensity value $Im(i, j)$ at a single point coordinated i in x axis, j in y axis

t number of steps

Ensure: H_{Im} simple haar wavelet decomposition of given input image

Initialize $T = \log(\frac{n}{2}) + 1$

```
1: for all  $t$  such that  $t = 1, 2, \dots, T$  do
2:    $size = 2^{T-t}$ 
3:   for all  $i$  such that  $i = 0, 2, \dots, size$  do
4:     for all  $j$  such that  $j = 0, 2, \dots, size$  do
5:        $offset = size/2$ 
6:        $A = Im(i, j)$ 
7:        $B = Im(i + 1, j)$ 
8:        $C = Im(i, j + 1)$ 
9:        $D = Im(i + 1, j + 1)$ 
10:       $H_{Im}(i/2, j/2) = \frac{A+B+C+D}{4}$  : averaging
11:       $H_{Im}(offset + i/2, j/2) = \frac{A-B+C-D}{2}$  : alpha
12:       $H_{Im}(i/2, offset + j/2) = \frac{A+B-C-D}{2}$  : beta
13:       $H_{Im}(offset + i/2, offset + j/2) = \frac{A-B-C+D}{2}$  : gamma
14:    end for
15:  end for
16: end for
```

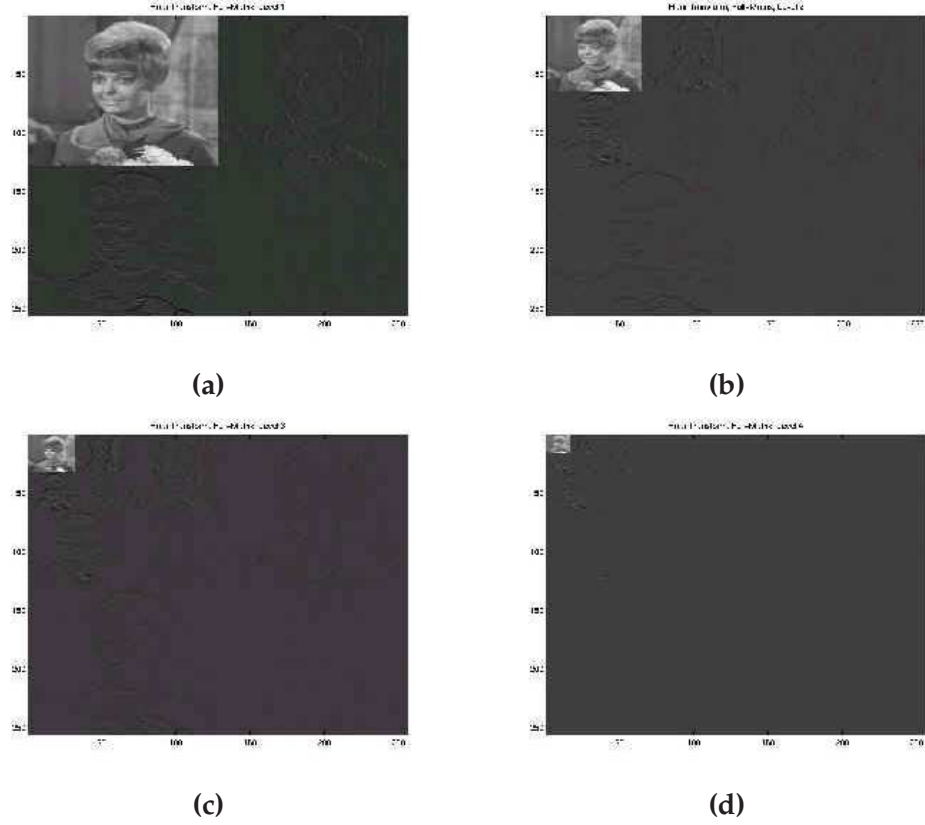


Figure 4.2: (a) Level-1 Transformation (b) Level-2 Transformation (c) Level-3 Transformation (d) Level-4 Transformation

4.3 Experiment Results

In this section experiment results are provided for Bioinformatics data set and Face-nonface data set, consecutively.

4.3.1 Bioinformatics Data Set

In Table 4.5, AdaBoost(AB), fixed length encoding(EB), variable length encoding(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. Detection rate is computed by

$$DR = \frac{\text{number of true positives}}{\text{number of positive samples}}$$

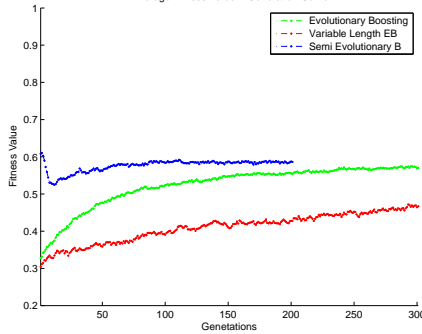
False alarm rate is computed with

$$FA = \frac{\text{number of false positives}}{\text{number of negative samples}}$$

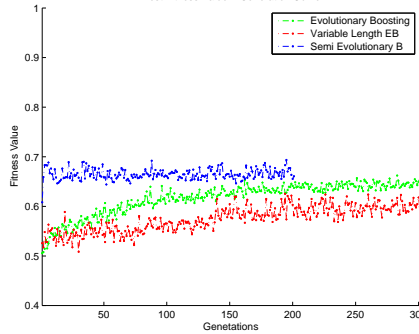
Table 4.5: Experiment-1 on Bioinformatics Data Set

Algorithm	CR	DR	FA	Algorithm	CR	DR	FA
AB ₄₆	0.635	0.645	0.010	AB ₄₆	0.541	0.597	0.056
VEB ₃₇	0.638	0.759	0.120	VEB ₃₇	0.564	0.712	0.148
AB ₃₇	0.610	0.626	0.016	AB ₃₇	0.515	0.583	0.067
EB ₃₀	0.665	0.788	0.122	EB ₃₀	0.603	0.767	0.164
SEB ₃₀	0.693	0.807	0.113	SEB ₃₀	0.585	0.760	0.175
AB ₃₀	0.606	0.640	0.033	AB ₃₀	0.541	0.622	0.080
(a) Performances in Training Set				(b) Performances in Testing Set			
NP:		600(SEB-400)		NG:		300(SEB-200)	
P _{mut} :		0.1		P _{cross} :		0.9	
FitnessFunction:		FF _{CR}		Selection:		Roulette-Wheel	
Mutation:		Swap Mutation		Crossover:		Single Point	
(c) Parameters							

Average Fitness Value – Generation Curve



Best Fitness Value – Generation Curve



(d) Mean Fitness - Generation Curve	(e) Best Fitness - Generation Curve
--	--

Classification rate is $CR = DR - FA$.

AB (AB_{46}) combines 46 classifiers to achieve best performance on training set. On the other hand, 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 achieves best performance in classification and detection rate by 0.693 and 0.807, respectively. On the other hand, best false alarm rate is found by AB (AB_{46}) with 0.010. In case of testing set, best performance is gathered by EB (EB_{30}) with 0.603 classification rate and 0.767 detection rate. Best false alarm rate is again achieved by AB (AB_{46}).

Remarkable point in this experiments are all evolutionary search boosts better permutations than AdaBoost does. For example, in 30 weak classifier boosted classifiers, SEB (SEB_{30}), EB (EB_{30}) and AB (AB_{30}) perform 0.693, 0.665 and 0.606 classification rates in training set respectively. Moreover, same classifiers achieves 0.585,

Table 4.6: Comparison of AdaBoost and fixed length encoding genetic algorithm in the training set

T	<i>AdaBoost</i>			<i>Fixed Length Encoding</i>		
	<i>DR</i>	<i>FA</i>	<i>CR</i>	<i>DR</i>	<i>FA</i>	<i>CR</i>
2	0.614232	0.219476	0.394757	0.773783	0.416479	0.357303
3	0.707116	0.199251	0.507865	0.643446	0.241199	0.402247
4	0.701873	0.213483	0.48839	0.692884	0.25618	0.436704
5	0.715356	0.183521	0.531835	0.645693	0.233708	0.411985
6	0.68839	0.161798	0.526592	0.722846	0.199251	0.523596
7	0.6397	0.116105	0.523596	0.770787	0.259176	0.51161
8	0.652434	0.102622	0.549813	0.752809	0.220974	0.531835
9	0.6397	0.100375	0.539326	0.713109	0.179026	0.534082
10	0.659925	0.10412	0.555805	0.720599	0.172285	0.548315

T is the number of boosted weak classifiers

0.603 and 0.541 in testing set, successively. This experiment proves that better permutations with same number of classifiers can be formed other than AdaBoost does with greedy search.

Mean fitness - generation curve represents average value of individuals fitness scores in generations, best fitness - generation curve represents best fitness valued individual at generation g .

Another interesting point is that AdaBoost achieves lowest false alarms rates, whereas, evolutionary algorithms concentrate on high detection rates. Table 4.6 depicts that this behavior difference is basically depends on the selection of hypothesis in very first iterations. Fixed length encoding starts with a group of hypothesis that cares more on detection rate than on false alarm rate, otherwise is true for AdaBoost.

Another experiment conducted on Bioinformatics data set is the procedure of initialization of population by unselected hypotheses proposed in Section 3.2.3. Experimental results show that proposed system performs slightly better than AdaBoost with relatively less number of classifiers. However, there is a drastic difference in testing data (AdaBoost has 0.541 classification rate, Genetic Algorithm achieves 0.632).

In Table 4.8, same experiments are done with same parameters except a different fitness function is used. This time, area under Receiver Operating Analysis(ROC) curve is used instead of classification rates. In all cases, AdaBoost outperforms the evolutionary algorithms. One of the reason for this is that computing ROC curve

Table 4.7: Comparison of AdaBoost and fixed length encoding with initialization of unselected hypotheses

<i>Algorithm</i>	<i>Training Data Set</i>			<i>Testing Data Set</i>		
	<i>DR</i>	<i>FA</i>	<i>CR</i>	<i>DR</i>	<i>FA</i>	<i>CR</i>
AB_{46}	0.645693	0.0104869	0.635206	0.597753	0.0561798	0.541573
EB_{25}	0.783521	0.14382	0.6397	0.762921	0.130337	0.632584
AB_{25}	0.613483	0.0374532	0.57603	0.583146	0.0808989	0.502247

requires a projection from feature hyper-planes to one dimensional confidence rates. This is even a harder problem than classification and the search for a better ROC curve is almost impractical with these parameters of genetic algorithms. Other reason is that in very first steps of the algorithm, AdaBoost concentrates on general behavior of the data, then at later stages, classifiers are successively chosen for hard samples. Confidence rates that are needed to draw ROC curves are calculated by

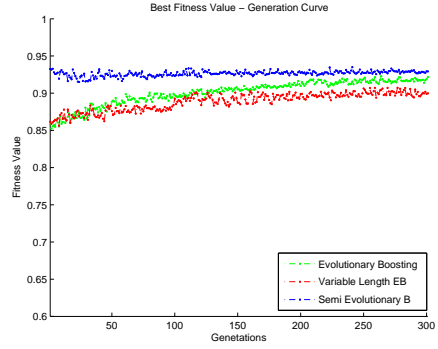
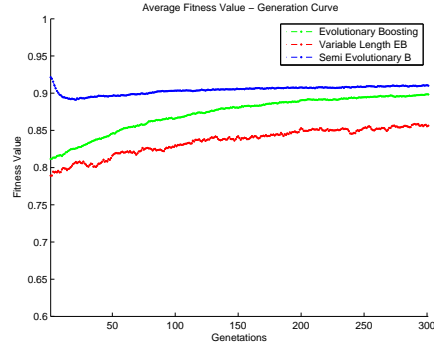
$$\sum_{t=1}^T \alpha_t h_t(x) - \frac{1}{2} \sum_{t=1}^T \alpha_t$$

where $\alpha_t = \frac{\varepsilon_t}{1-\varepsilon_t}$. As training error rate decreases, the value α_t also decreases. In first iterations, AdaBoost assigns large α_t values to correctly classified samples and this approach provides a good projection on one dimensional confidence rate space. Genetic algorithm counterparts, there is no such behavior. Thus, in terms of ROC curves, AdaBoost will be better than any other methods that this work proposes. Moreover, AdaBoost classifies most of the common data in very few iterations. Then algorithm concentrates on difficult samples. In proposed systems, some of the easily classifiable data is sacrificed in order to achieve better classification rates. In addition to this, constructing a ROC curve over training set requires a projection from M dimensional space to one dimensional space. Finding a suitable projection over one dimensional space from a hyperspace is more challenging task than classification of data. To outperform the classical AdaBoost, more individuals has to be present in the initial population and more generations population has to be evolved.

Table 4.8: Experiment-2 on Bioinformatics Data Set

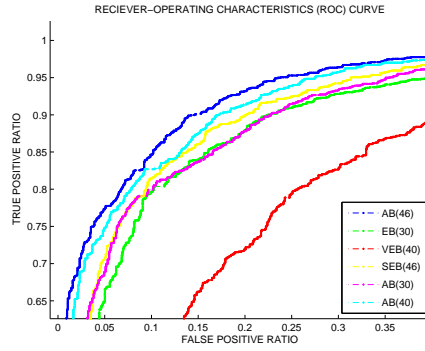
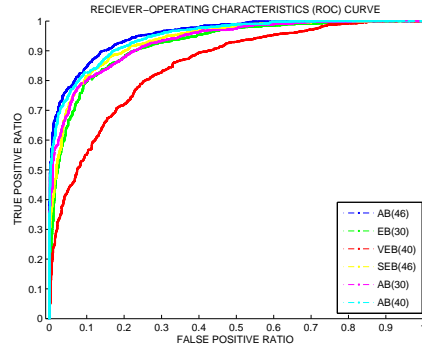
NP :	600	NG :	300
P_{mut} :	0.1	P_{cross} :	0.9
$FitnessFunction$:	FF_{ROC}	Selection:	Roulette-Wheel
Mutation:	Swap Mutation	Crossover:	Single Point

(a) Parameters



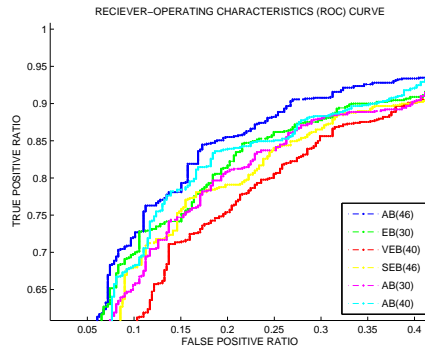
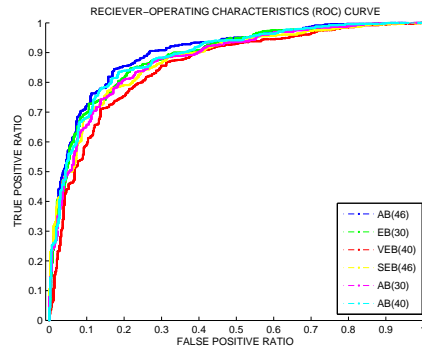
(b) Mean Fitness - Generation Curve

(c) Best Fitness - Generation Curve



(d) ROC Curve over Training Set

(e) ROC Curve over Training Set (Detailed)



(f) ROC Curve over Testing Set

(g) ROC Curve over Testing Set (Detailed)

Table 4.9: Experiments on Face-Nonface Data Set(Average) with classification rate based fitness function

<i>Algorithm</i>	<i>Training Set</i>			<i>Testing Set</i>		
	<i>CR</i>	<i>DR</i>	<i>FA</i>	<i>CR</i>	<i>DR</i>	<i>FA</i>
AB_{42}	0.871	0.988	0.117	0.852	0.980	0.128
EB_{20}	0.880	0.939	0.059	0.864	0.932	0.067
AB_{20}	0.873	0.965	0.091	0.860	0.960	0.100
VEB_{37}	0.885	0.940	0.055	0.871	0.933	0.062
AB_{37}	0.873	0.988	0.115	0.853	0.980	0.127
(a) Average Classification, Detection, False Alarm Rates						
<i>Algorithm</i>	<i>Training Set</i>			<i>Testing Set</i>		
	<i>CR</i>	<i>DR</i>	<i>FA</i>	<i>CR</i>	<i>DR</i>	<i>FA</i>
AB_{42}	0.026	0.009	0.032	0.025	0.008	0.031
EB_{20}	0.004	0.006	0.005	0.005	0.009	0.007
AB_{20}	0.010	0.011	0.020	0.010	0.012	0.020
VEB_{37}	0.005	0.008	0.010	0.006	0.009	0.010
AB_{37}	0.015	0.003	0.017	0.016	0.004	0.019
(b) Standard Deviation of Classification, Detection, False Alarm Rates						

4.3.2 Face-Nonface Data Set

Table 4.9 shows average values of 10 experiments performed on face-nonface data set. Classification rate is used as fitness function. The first row AB_{42} is the best average AdaBoost performance achieved on training set. Average number of boosted weak classifiers in all experiments is 42. The second and third rows EB_{20} and AB_{20} are the performances of fixed length encoding genetic algorithm and AdaBoost by 20 boosted weak classifiers, respectively. Finally last two rows VEB_{37} and AB_{37} are variable length encoding genetic algorithm and AdaBoost. In last two rows, variable length encoding genetic algorithm and AdaBoost with same number of weak classifiers are compared. Genetic algorithms achieve better performances even with same and less number of boosted weak classifiers.

In this data set, AB accomplishes 0.988 detection rate which is higher than EB and VEB, 0.939 and 0.940, respectively. This is opposite behavior observed in Bioinformatics data set. Details of some selected experiments are given below.

Table 4.10: Experiments on Face-Nonface Data Set-1

Algorithm	CR	DR	FA	Algorithm	CR	DR	FA
AB_{32}	0.889	0.977	0.088	AB_{36}	0.870	0.974	0.104
EB_{20}	0.881	0.941	0.059	EB_{20}	0.864	0.935	0.070
AB_{20}	0.878	0.965	0.086	AB_{20}	0.865	0.963	0.098
VEB_{38}	0.876	0.927	0.051	VEB_{40}	0.865	0.925	0.060
AB_{38}	0.867	0.985	0.117	AB_{40}	0.847	0.985	0.137

(a) Performances in Training Set

(b) Performances in Testing Set

$NP:$ 600

$P_{mut}:$ 0.1

Fitness Function: FF_{CR}

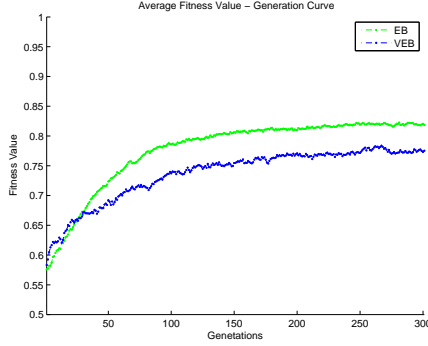
$NG:$ 300

$P_{cross}:$ 0.9

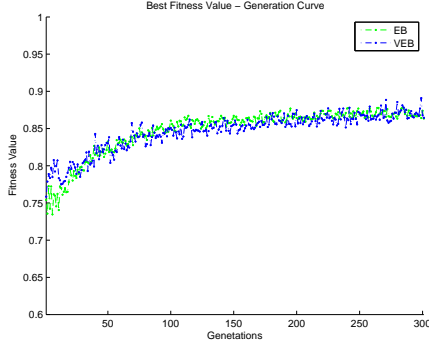
Selection: Roulette-Wheel

(c) Parameters

Average Fitness Value – Generation Curve



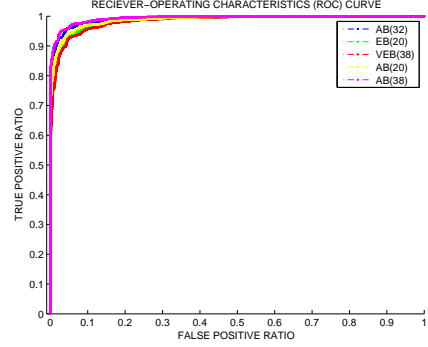
Best Fitness Value – Generation Curve



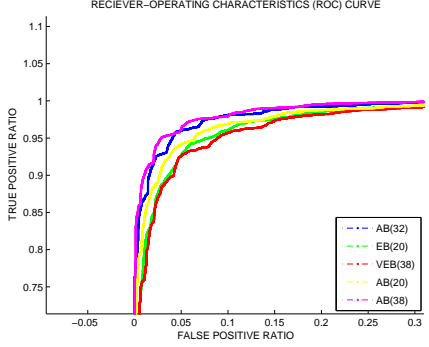
(d) Mean Fitness-Generation Curve

(e) Best Fitness-Generation Curve

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE



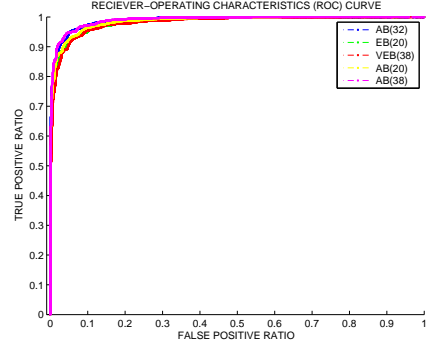
RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE



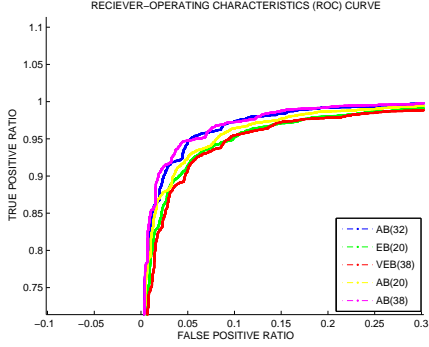
(f) ROC Curve over Training Set

(g) ROC Curve over Training Set(Detailed)

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE



RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE



(h) ROC Curve over Testing Set

(i) ROC Curve over Testing Set(Detailed)

Table 4.11: Experiments on Face-Nonface Data Set-2

Algorithm	CR	DR	FA	Algorithm	CR	DR	FA
AB_{30}	0.840	0.988	0.147	AB_{30}	0.819	0.980	0.161
EB_{20}	0.878	0.943	0.065	EB_{20}	0.865	0.938	0.073
AB_{20}	0.862	0.970	0.107	AB_{20}	0.859	0.966	0.107
VEB_{36}	0.890	0.927	0.037	VEB_{36}	0.870	0.912	0.042
AB_{36}	0.858	0.989	0.130	AB_{36}	0.836	0.982	0.145

(a) Performances in Training Set

(b) Performances in Testing Set

$NP:$ 600

$P_{mut}:$ 0.1

Fitness Function: FF_{CR}

$NG:$ 300

$P_{cross}:$ 0.9

Selection: Roulette-Wheel

(c) Parameters

Average Fitness Value – Generation Curve

Best Fitness Value – Generation Curve

(d) Mean Fitness-Generation Curve

(e) Best Fitness-Generation Curve

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

(f) ROC Curve over Training Set

(g) ROC Curve over Training Set(Detailed)

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

(h) ROC Curve over Testing Set

(i) ROC Curve over Testing Set(Detailed)

Table 4.12: Experiments on Face-Nonface Data Set-3

Algorithm	CR	DR	FA	Algorithm	CR	DR	FA
AB_{34}	0.880	0.984	0.104	AB_{34}	0.861	0.973	0.112
EB_{20}	0.879	0.941	0.061	EB_{20}	0.864	0.930	0.066
AB_{20}	0.883	0.943	0.060	AB_{20}	0.868	0.938	0.069
VEB_{34}	0.891	0.937	0.045	VEB_{34}	0.879	0.932	0.052
AB_{34}	0.880	0.984	0.104	AB_{34}	0.861	0.973	0.112

(a) Performances in Training Set

(b) Performances in Testing Set

$NP:$ 600

$P_{mut}:$ 0.1

Fitness Function: FF_{CR}

$NG:$ 300

$P_{cross}:$ 0.9

Selection: Roulette-Wheel

(c) Parameters

Average Fitness Value – Generation Curve

Best Fitness Value – Generation Curve

(d) Mean Fitness-Generation Curve

(e) Best Fitness-Generation Curve

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

(f) ROC Curve over Training Set

(g) ROC Curve over Training Set (Detailed)

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

RECEIVER-OPERATING CHARACTERISTICS (ROC) CURVE

(h) ROC Curve over Testing Set

(i) ROC Curve over Testing Set (Detailed)

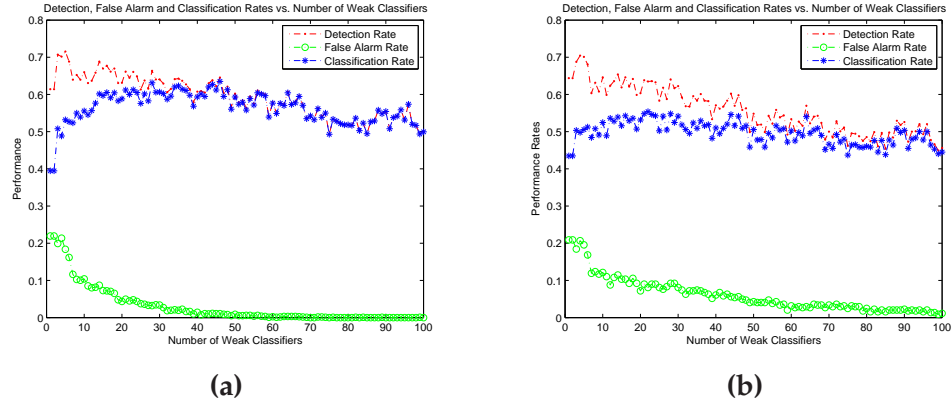


Figure 4.3: Performance Rates vs. Number of Boosted Weak Classifiers Graph. (a) Training Set (b) Testing Set

4.4 Comparison of AdaBoost and Proposed Methods

In this section, a comparison of AdaBoost and proposed genetic algorithms is discussed under several title, generalization vs. overfitting, dimension of feature space, size of data set and complexity.

4.4.1 Generalization vs. Overfitting

Boosting algorithms suffer from overfitting, when the number of hypotheses to be selected T gets larger and larger. There exists no rule of thumb that decides suitable number of boosting steps. When overfitting happens, generalization capability of the algorithms drops sharply. Because algorithm memorizes the training data.

In Figure 4.3, after ensembling 46 classifiers, performance of AdaBoost either remains the same or decreases. Moreover, testing performance decreases rapidly as the number of ensembled classifiers increased. On the other hand, independent of number of boosted classifiers, genetic algorithms try to maximize classification function. In Table 4.10, AdaBoost ensembles 32 weak classifiers to achieve 0.889 classification rate on the training set and classification rate drops 0.867 with 38 boosted weak hypotheses, however variable length encoding still performs 0.876 classification rate with 38 ensembled classifiers. One drawback of increasing number of hypotheses is increasing complexity and search space of the problem.

Table 4.7 shows that the performance of evolutionary search differs less slightly than AdaBoost does from training data to testing data. This indicates that the pro-

posed methods have stronger generalization ability.

4.4.2 Dimension of The Feature Space

Dimension of feature space is the same as to the number of weak hypotheses. Because of greedy search methodology, AdaBoost does not face any difficulty in training. However, when the number of hypothesis gets larger and larger, search space (permutation space mentioned at Section 3.1) also grows exponentially. For that reason, population size and number of evolving generations parameters of genetic algorithms have also been increased, this operation can be impractical without presence of high performance computing systems.

4.4.3 Complexities of Algorithms

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 solution from H^T dimensional space. Fixed length solution tries to find fittest individual in this space. As H and T becomes larger and larger, hypothesis space grows exponentially.

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 than 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.

CHAPTER 5

CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis, a novel approach to boosting is proposed. This approach presents a disciplined evolutionary search that can be alternative to both exhaustive and greedy search in hypothesis space. Boosting provides a good dimensionality reduction and a linear classifier with all desired properties. After forming final classifier, boosting concentrates on only selected hypotheses among all possible hypotheses.

In Chapter 2, AdaBoost and Boosting theory are discussed with both superiorities and weaknesses. Other proposed systems in the literature which try to solve partial problems of AdaBoost is also considered.

In Chapter 3, idea of combining evolutionary search with genetic algorithms in boosting is provided with both representations, fitness functions and initialization schema. All modifications in these concepts are done to find solutions to the problems of the classical AdaBoost.

Throughout the experiments demonstrated in Chapter 4, proposed system achieves better performances compared to the classical AdaBoost. In case of fixed length coding, evolutionary search finds good solutions compared to AdaBoost at the same number of boosting rounds and almost same solutions with low number of boosted classifier. Although complexity of evolutionary search at high number of boosting steps make evolutionary search impractical, in small sets or medium problems evolutionary search is a good alternative to AdaBoost.

Variable length encoding suffers from complexity of the search space. However, it provides a classifier without parameters, even number of rounds. According to experiments, evolutionary boosting achieves better classification rates, but AdaBoost is capable of generating better ROC curves. To achieve such curves, more individuals in the initial population with high number of generations to be evolved are needed.

For the future work, pattern classification for boosting with genetic algorithms can be improved in several ways,

- In the current system, with the genetic algorithms, computational complexity of fitness function is high, this makes evolutionary search almost impractical in very high dimensional feature spaces and large amount of training data. On the other hand, computation of fitness values are completely independent tasks, based on this task, algorithm can be implemented in parallel computing fashion. Computation of each fitness value in grid, cluster like systems enables to attempt more complex problems.
- New representations that are compatible with nature of genetic algorithms can be proposed in this context. Due to new representations, new mutation and crossover function that favor more qualified individuals can be defined. Good representation with appropriate operations produces good results.
- Fitness functions can be adjusted to perform more accurate solutions. Different selection strategies may also be applied.

REFERENCES

- [1] "Webster online dictionary - <http://www.websters-online-dictionary.org/>."
- [2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, 2000.
- [3] L. Kuncheva, *Combining Pattern Classifiers*. John Wiley and Sons Inc, 2004.
- [4] 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.
- [5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision Pattern Recognition(CVPR)*, 2001.
- [6] S. Li, Z. Zhang, H. Shum, and H. Zhang, "Floatboost learning for classification," *NIPS 15.*, 2002.
- [7] R. Schalkoff, *Pattern recognition - statistical, structural and neural approaches*. John Wiley and Sons, 1992.
- [8] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*. The MIT Press, 2002.
- [9] L. G. Valiant, "A theory of learnable," *Communacations of the ACM*, vol. 27, pp. 1134–1142, November 1984.
- [10] M. Kearns and L. G. Valiant, "Learning boolean formula and finite automata," tech. rep., Harvard University Aiken Computation Laboratory putation Laboratory, 1988.
- [11] R. E. Schapire, "The strength of weak learnability," *International Conference on Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.
- [12] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.
- [13] 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.
- [14] R. Schapire, "The boosting approach to machine learning: An overview," *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.
- [15] D. Cristinacce and T. Cootes, "Facial feature detection using adaboost with shape constraints," *British Machine Vision Conference*, 2003.

- [16] W. Fan, J. Zhang, and S. J. Stolfo, "The application of adaboost for distributed, scalable and on-line learning," *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999.
- [17] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," tech. rep., Stanford University, 1998.
- [18] J. Movellan, B. Fortenberry, and I. Fasel, "A generative framework for real time object detection," tech. rep., Machine Perception Laboratory, Institute for Neural Computation, University of California San Diego, 2003.
- [19] G. Ratsch, T. Onoda, and K. R. Muller, "An improvement of adaboost to avoid overfitting," *Advances in Neural Information Processing Systems*, 1999.
- [20] Y. Freund, "An adaptive version of the boost by majority algorithm," *Machine Learning*, vol. 43, pp. 293–318, June 2001.
- [21] M. Dettling and P. Buehlmann, "Boosting for tumor classification with gene expression data," *Bioinformatics*, 2003.
- [22] C. Domingo and O. Watanabe, "Madaboost: A modification of adaboost," in *Proc. 13th Annu. Conference on Comput. Learning Theory*, 2000.
- [23] 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.
- [24] "Genetic algorithms - wikipedia."
- [25] A. Treptow and A. Zell, "Combining adaboost learning and evolutionary search to select features for real-time object detection."
- [26] B. Liu, M. B., and A. H.A., "Improving genetic classifiers with a boosting algorithm," *Congress on Evolutionary Computation*, vol. 4, pp. 2596– 2602, 2003.
- [27] X. Wang and H. Wang, "Classification by evolutionary ensembles," *Pattern Recognition (Elsevier)*, vol. 39, pp. 595–607, April 2006.
- [28] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on (PAMI)*, vol. 20, no. 1, pp. 23–38, 1998.
- [29] H. Rowley, S. Baluja, and T. Kanade, "Rotation invariant neural network-based face detection," tech. rep., Computer Science Department of Carnegie Mellon University, 1997.
- [30] M. Wall, "Galib:mathews genetic algorithms library - <http://lancet.mit.edu/ga/>."
- [31] C. Mulcahy, "Image compression using the haar wavelet transform," *Spelman College Science and Mathematics*, vol. 1, pp. 22–31, April 1997.