Pattern Recognition Using Genetic Programming for Classification of Diabetes and Modulation Data

A Thesis submitted to the University of Liverpool
for the degree of Doctor in Philosophy
in the Faculty of Engineering

By
Muhammad Waqar Aslam
Department of Electrical Engineering & Electronics

February 2013

Abstract

The field of science whose goal is to assign each input object to one of the given set of categories is called *pattern recognition*. A standard pattern recognition system can be divided into two main components, feature extraction and pattern classification. During the process of feature extraction, the information relevant to the problem is extracted from raw data, prepared as features and passed to a classifier for assignment of a label. Generally, the extracted feature vector has fairly large number of dimensions, from the order of hundreds to thousands, increasing the computational complexity significantly. Feature generation is introduced to handle this problem which filters out the unwanted features. The functionality of feature generation has become very important in modern pattern recognition systems as it not only reduces the dimensions of the data but also increases the classification accuracy.

A genetic programming (GP) based framework has been utilised in this thesis for feature generation. GP is a process based on the biological evolution of features in which combination of original features are evolved. The stronger features propagate in this evolution while weaker features are discarded. The process of evolution is optimised in a way to improve the discriminatory power of features in every new generation. The final features generated have more discriminatory power than the original features, making the job of classifier easier.

One of the main problems in GP is a tendency towards suboptimal-convergence. In this thesis, the response of features for each input instance which gives insight into strengths and weaknesses of features is used to avoid suboptimal-convergence. The strengths and weaknesses are utilised to find the right partners during crossover operation which not only helps to avoid suboptimal-convergence but also makes the evolution more effective.

In order to thoroughly examine the capabilities of GP for feature generation and to cover different scenarios, different combinations of GP are designed. Each combination of GP differs in the way, the capability of the features to solve the problem (*the fitness function*) is evaluated. In this research Fisher criterion, Support Vector Machine and Artificial Neural Network have been used to evaluate the fitness function for binary classification problems while K-nearest neighbour classifier has been used for fitness evaluation of multi-class classification problems.

Two Real world classification problems (diabetes detection and modulation classification) are used to evaluate the performance of GP for feature generation. These two problems belong to two different categories; diabetes detection is a binary classification problem while modulation classification is a multi-class classification problem. The application of GP for both the problems helps to evaluate the performance of GP for both categories. A series of experiments are conducted to evaluate and compare the results obtained using GP. The results demonstrate the superiority of GP generated features compared to features generated by conventional methods.

Contents

List of Figures

List of Tables

Declaration

No portion of the work has been submitted in support of an application for another degree or qualification in any other university or other institution of learning. The thesis contains no material previously published by other person except the references.

Dedication

This thesis is dedicated to my parents, for their endless love, support and encouragement.

Copyright

Acknowledgements

I would never have been able to write this thesis without the support and guidance of the kind people around me, who in one way or another played their valuable part in the completion of this study.

Above all, I would like to express my deepest gratitude to my primary supervisor Prof. Asoke K. Nandi for the prompt and wise feedback to guide my research throughout this period. His support and advice have been invaluable on both academic and personal level, for which I am extremely grateful. I must also thank the University of Azad Jammu & Kashmir for the financial support.

Special thanks to Dr. Sergio Bravo, Zhechen Zhu and Dr Rui Fa for maintaining the cluster, without which I would never have been able to run long simulations. Special mention to Mr Hussain Jaafar who started PhD roughly the same time, as myself and with whom I spent most of my time. I am truly indebted and thankful to Zhechen Zhu for all the technical help regarding my research. Thanks must go to my colleagues in the signal processing group with whom I have had the pleasure of working with.

Further special mentions to my house mates (ever changing faces but never ending fun), Ahsan and Laeeq, for providing continuous entertainment and delicious food for dinner; Muneeb, one of the most hard working persons I have ever met; Khawaja, for all the adventures we had; Naveed, for all the laughs; Saad Ali, for his sense of humour; Samad, for waking the romantic side of me; Nauman, one of a kind; Saad Anis, for all the tours around UK (especially for food); Saood, for all the dinner gatherings; Raja Adeel for all the liveliness and entertainment ... there are many more and to list them all here is an impossible task.

Special thanks are due to my uncle, aunt and their family for making me feel home all these years. I can never pay back for all the support and love they provided.

In the department there have been many people who have made my life more enjoyable. It is a great pleasure to thank, Neda, for her company making my stay

wonderful and for introducing me to Persian food, language and culture; Farhad, for his versatile talks about politics at afternoon tea; Moin, for all the discussion about cricket and other sports, and Obilor, for all the loud songs he used to sing.

Finally I would like to thank my family for all the support, love and belief over all these years, particularly my mother for bearing the separation for four years, my lovely sisters for all the love, and Dad for all the interest and support.

Glossary of Abbreviations

AF	Accuracy Fitness
ANN	Artificial Neural Network
AWGN	Additive White Gaussian Noise
BP	Back-Propagation
BPSK	Binary Phase Shift Keying
CDF	Cumulative Distribution Function
CPS	Comparative Partner Selection
EWF	Equal Weighted Fitness
FB	Feature Based
GA	Genetic Algorithm
GF	Geometric Mean Based Fitness
GP	Genetic Programming
ICA	Independent Component Analysis
KS	Kolmogorov-Smirnov
KNN	K-Nearest Neighbour
MLP	Multi Layer Perceptron
PR	Pattern Recognition
PCA	Principal Component Analysis
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RBF	Radial Basis Function
SVM	Support Vector Machine
SNR	Signal to Noise Ratio
VWF	Variable Weighted Fitness
WSF	Weighted Square Fitness

Chapter 1

Introduction

1.1 Motivation

Pattern recognition is a scientific discipline which aims to classify the data or objects into different classes using some information from either a priori knowledge or statistical distribution of the data. It has become increasingly important in recent times and is employed in a vast spectrum of applications including speech recognition, image processing, face recognition, target recognition, medical diagnosis, etc.

A simple pattern recognition system follows three main steps. First, a sensor (transducer) is used at the input to take measurements or to extract the raw data. Second, the raw data from the sensor is processed to pick up *the relevant* information also known as feature extraction. It is important to extract the features relevant to the task to be carried out. Generally the number of features available from the raw data is large but only relevant features are extracted so the process is also sometimes called dimensionality reduction. Finally, a classifier is used to divide the data samples into different categories by using the discriminatory information from the extracted features. The classifier could be as simple as a simple decision boundary in a feature space and could be as complex as a neural network.

It is clear from the above description that the process of feature extraction from the raw data is of vital importance in pattern recognition system as it serves as a bridge between the raw data and the classifier. Ideally, the extracted features should have enough discriminatory information so that the job of classifier becomes easier. Unfortunately, not all the extracted features contain useful information and some of them are unrelated, redundant, and noisy. Such noisy features need to be filtered out so that the classifier can perform classification easily without confusion by noise.

The filtering of noisy features is performed by adding another block which removes redundant features either by feature selection or feature generation. Normally, the statistical distribution of the data assists the feature selection process in which useful features are kept and noisy features are discarded. The process of feature generation is more complex in which extracted features are transformed to less number of new features such that the maximum discriminatory information is kept. A large number of feature generation methods have been developed for various problem domains such as Principal Component Analysis (PCA) [1], Independent Component Analysis (ICA), Semidefinite Embedding, etc.

Genetic Programming (GP) has been used for feature generation in this study. GP belongs to the class of evolutionary algorithms, based on the process of biological evolution. It differs from other traditional machine learning methods in a way that it evolves computer programs (individuals) during evolution which are the potential solutions to the problem presented to GP. During evolution, GP tries to optimise the population of individuals with the help of fitness value (individual's ability to solve the problem). GP has gained much attention in recent years for certain problem domains, especially for the problem domains that other traditional machine learning techniques are unable to solve. It has certain advantages over other machine learning methods for complex problem domains. It does not require any a priori information about the distribution of the data. The preprocessing of the data is not required and data can be used directly in its raw form. Moreover, not only the parameters but the actual structure of solutions can be completely changed during evolution.

The motivation of using GP for feature generation is driven by the fact that GP performs feature selection and transformation at the same time. GP has the inherent capability to select useful features (as a part of evolution) and ignore others. The feature transformation is performed by making non-linear combinations of the extracted features with the aid of available mathematical functions during evolution.

With recent advancement in technology, the modern computers have become so fast that even a home personal computer is capable of carrying out complex computations in short period of time, making computational complexity of GP (often questioned) less relevant. More advanced computers, especially designed for computations can be used for more sophisticated and intelligent search for optimal solutions.

This research has been conducted with the objective to examine and evaluate the existing GP techniques, and develop new novel strategies to help GP solve the real world complex classification problems more efficiently. A pairwise mating strategy based on phenotype variance of individuals is built to assist GP in classification task. Moreover, machine learning classifiers are used to evaluate the fitness of individuals by quantifying their ability of classification. The novel strategies developed in this research are generalised by applying them on real world classification problems of diabetes detection and modulation classification.

1.2 Summary of Contributions

Upon reaching the final stage of this research, it is believed that the following contributions have been made during this research:

- The introduction of the *Binary String Fitness Characterisation (BSFC)* parameter for evaluating the discriminatory power of solutions, proposed by GP for classification problems. Based on the *BSFC*, a pairwise mating strategy (*Comparative Partner Selection (CPS)*) for classification problems is introduced. This strategy helps genetic operator crossover to find the right partners for crossover. It increases the possibility of optimal convergence by reducing the population phenotype variance. (Chapter 5)

- Introduction of the improved CPS algorithm in which the parents selected by CPS criteria are allowed to produce more offspring. It increases the possibility of finding the optimum solution by exploiting the search space around optimum solutions. (Chapter 6)

- Introduction of the novel technique for evaluating the fitness function of solutions proposed by GP for multi-class classification problems. Instead of using traditional methods, K-nearest neighbour (KNN) is used for fitness evaluation which enables GP to handle multi-class classification. (Chapter 7)

- Introduction of machine learning classifiers (Support vector machine (SVM) and Artificial neural network (ANN)) for evaluating the discriminatory power of GP solutions. (Chapter 8)

- For the first time GP based feature generation approach has been introduced for modulation classification and remarkable improvement in classification accuracy of noisy modulated signals have been achieved. New single dimension feature by making combination of the existing higher order cumulant features has been proposed using GP. (Chapter 8)

1.3 Thesis Organisation

The major tasks carried out in this research are the feature generation and dimensionality reduction using GP, by employing different methods for fitness evaluation. In order to present the findings of this research in a readable way, the thesis has been divided into nine chapters, which are listed as follows:

Chapter 1 presents the motivation, contributions, publications, and organisation of the thesis.

Chapter 2 presents an overview of pattern recognitions systems. The basic components of a general pattern recognition system are discussed. Different feature selection, feature generation and classification methods used in this research are presented. These methods will be later used in the thesis for different classification problems.

Chapter 3 presents an overview of genetic programming, its basic structure, its essential elements, and the implementation issues related to the design of the program for feature generation task. The applications of genetic programming for classification problems are discussed in detail. The dimensionality reduction from multiple features to a single feature named as *super-feature* is discussed with a practical example.

Chapter 4 provides a basic description of the classification problems considered in this thesis. The dataset and the methods used for pre-processing of the data are presented. The importance and the applications of the problems are discussed. It also provides the mathematical preliminaries which might be useful for those readers, not having necessary pre-requisites towards this research.

Chapter 5 introduces the concept of binary string fitness measure, a parameter used to indicate the behavioural diversity of the population. A novel technique for calculating the binary string for classification problems is introduced. It also describes the CPS scheme which is used to help the genetic operator crossover, in finding the right partners for crossover. The concept of binary string and CPS are applied on the diabetes detection, a practical binary classification problem, highlighting the advantages offered by these concepts in finding the optimum solution.

Chapter 6 presents a detailed analysis about the application of GP for classification of imbalanced datasets. Different fitness functions are used during training which aim to maintain a balance between specificity and sensitivity during evolution, resulting in unbiased classifiers. It also proposes a novel technique for improving the performance of traditional CPS process. In new improved crossover the parents selected for crossover through CPS process are allowed to produce more children to explore the search space more efficiently.

Chapter 7 provides an insight into the application of GP for multi-class classification problems. Automatic modulation classification, a communication systems problem is taken as a case study. The chapter also discusses the methods used in the literature for modulation classification problem and its applications. A unique GP structure for multi-class pattern classification is proposed which uses K-nearest neighbour rule for evaluating the solutions proposed by GP. A non-linear solution for multi-class recognition is produced by favouring the solutions having more discriminatory information about classes.

Chapter 8 presents a multi-stage genetic programming strategy for handling classes, difficult to separate at a single stage. GP shows a drift towards easily separable classes during evolution, resulting in low accuracy for other classes. This has been dealt by setting a separate stage for classes, neglected by GP at the first stage. The strategy has been applied on multi-class modulation classification problem.

Chapter 9 states the conclusions and summarises the contributions of this thesis. This is followed by a summary of possible future research directions.

1.4 List of Publications

Journal Papers

- **Muhammad Waqar Aslam**, Zhechen Zhu and Asoke K. Nandi, "Autmomatic Modulation Classification Using Combination of Genetic Programming and KNN", *IEEE Transactions on Wireless Communications*, vol. 11, no. 8, pp. 2742-2750, August 2012.
- **Muhammad Waqar Aslam**, Zhechen Zhu and Asoke K. Nandi, "Diabetes Classification Using Genetic Programming with Comparative Partner Selection", Submitted to *Expert Systems with Applications.*
- **Muhammad Waqar Aslam**, Zhechen Zhu and Asoke K. Nandi, "Diverse Partner Selection with Brood Recombination in Genetic Programming", Submitted to *IEEE Transactions on Evolutionary Computation.*
- **Zhechen Zhu**, Muhammad Waqar Aslam and Asoke K. Nandi, "Genetic Algorithm Optimized Distribution Sampling Test for M-QAM Modulation Classification", Submitted to *Signal Processing.*

Conference Papers

- **Muhammad Waqar Aslam**, Zhechen Zhu and Asoke K. Nandi, "Robust QAM Classification Using Genetic Programming and Fisher Criterion", *19th European Signal Processig Conference, Eusipco 2011,* Barcelona, Spain, 28 August - 02 September 2011.
- Zhechen Zhu, **Muhammad Waqar Aslam** and Asoke K. Nandi, "Support Vector Machine Assisted Genetic Programming for MQAM Classification", *International Symposium on Signals, Circuits and System (ISSCS 2011)*, Romania, 30 June - 01 July 2011.
- **Muhammad Waqar Aslam**, Zhechen Zhu and Asoke K. Nandi, "Automatic Digital Modulation Classification Using Genetic Programming with

K-Nearest Neighbor”, in *Proc. of the Military Communications Conference (MILCOM-2010),* San Jose, CA, USA, pp. 512-517, 31 October - 03 November 2010.

- Zhechen Zhu, **Muhammad Waqar Aslam**, and Asoke K. Nandi,“Augmented Genetic Programming for Automatic Digital Modulation Classification”, in *Proc. of the IEEE International workshop on Machine Learning for Signal Processing (MLSP-2010),* Kittila, Finland, pp. 391-396, 29 August - 1 September 2010.
- **Muhammad Waqar Aslam** and Asoke K. Nandi, “Detection of Diabetes Using Genetic Programming”, *18th European Signal Processig Conference, Eusipco 2010,* Aalborg, Denmark, pp.1184-1188, 23 - 27 August 2010.

Chapter 2

Pattern Recognition Systems

Pattern recognition (PR) is the field of science in which machines (or algorithms) are built to perform perceptual tasks which human are good at, e.g., speech recognition, face recognition, character recognition, etc. It has a broad range of applications including fingerprint identification, DNA sequence identification, machine vision, computer-aided medical diagnosis and much more. It is clear that for all these applications a reliable and accurate PR system would be immensely useful. Different definitions of PR are available in the literature and some of them are given below:

- *The act of taking in raw data and taking an action based on the category of the pattern* [2].

- *The scientific discipline whose goal is the classification of objects into a number of categories* [3].

Generally speaking, the goal of any PR system is to group the objects into a number of categories. The objects can be signal waveforms, images or some other type of measurements. The grouping of objects is an important task which may be performed using statistical patterns, artificial intelligence and many other methods. Recently PR has gained much attention in the field of industrial application due to increase in automation in industrial production. In this study PR has been used for classification task, however, it is also frequently used for regression, clustering and description tasks.

A typical PR system illustrating different stages is shown in Figure 2.1. A sensor or transducer is used at the input which yields a measurement, later used for feature extraction. The sensor can be a thermometer, camera, microphone, etc. For the modulation classification problem the sensor would be a receiving antenna. Features are extracted from this measurement which represent a distinct quality or characteristic. An ideal feature (for classification task) is the one whose

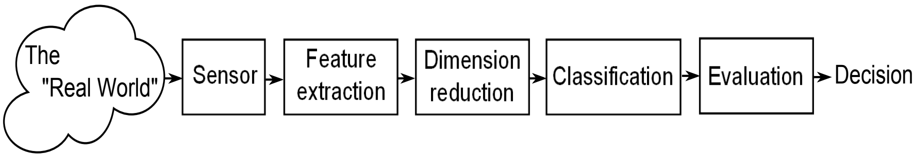


Figure 2.1: Basic version of a pattern recognition system.

values are very similar for the objects in the same category and different for the objects in different categories. If the number of extracted features is very large, a dimension reduction method is used to remove redundant or irrelevant features. After dimension reduction, the features are passed to a classifier. The task of classifier is to use these features and assign a class label to each pattern by making decision boundaries in feature space. Finally the efficiency of the trained classifier is evaluated using test data. The design cycle of a PR system is shown in Figure 2.2. In the coming sections each step of the design cycle will be explained in detail one by one.

2.1 Data Collection

Generally there are three types of data that we encounter in our daily life; categorical, ordinal and numerical data. Categorical data contains non-numerical data without any class ordering, e.g., a categorical variable could be description of the type of job a person does, e.g., medical, engineering, financial, etc. Categorical data with class ordering is called ordinal data, e.g., small, medium, large. Numerical data contains real numbers, e.g., temperature, height, blood pressure, etc. Since we ultimately want to perform some computation on the data, the categorical and ordinal data must be transformed into the numerical data.

The data collection is usually performed using some kind of transducer or sensor. This data contains the original signal with added noise. For modulation classification problem the receiving antenna acts as a sensor and collects (receives) the data (signal with noise).

2.2 Feature Extraction

Any object that can be classified, possesses some discriminative characteristics which differentiate between him and other objects. These characteristics are called features in PR problems. In some PR problems the data collected repre-

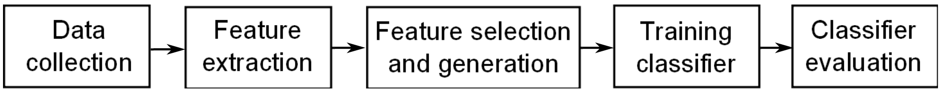


Figure 2.2: The design cycle of a pattern recognition system

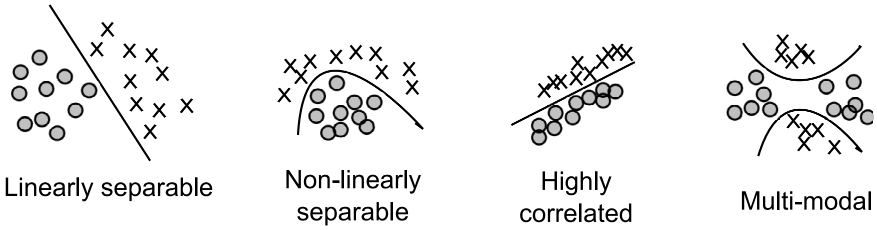


Figure 2.3: Examples of some ideal features

sents the actual features and there is no need to extract features explicitly. Features can be divided into three main categories, physical, structural and mathematical features. Physical features include colour of an object, its odour, its material, etc. Structural features represent the structural properties of an object, e.g., height, weight, width, etc. The mathematical features are the most important in the current context and represent mathematical properties of an object, e.g., mean, variance, eigenvectors, covariance matrix, principal components, etc. The process of feature extraction is also sometimes referred as feature generation in the literature but in this study the transformation of initially extracted features to new dimensions is called feature generation.

The choice of features is a critical design step and needs careful consideration. Ideally we would like a feature that is simple to extract, invariant to irrelevant transformations, insensitive to noise, and useful for discrimination between objects of different categories. Since the extracted features will be later used by the classifier for classification, an ideal feature should yield an information that makes the job of the classifier trivial which means that examples from the same class should have similar feature values and examples from different class should have different feature values. Some examples of the ideal features are given in Figure 2.3.

The choice and the number of features required is problem dependent. Usually for real world problems one feature is not enough to differentiate between objects of different categories. Multiple features representing the same object are organised into feature vectors and a set of all features is called feature space. Each occurrence of the features is termed as instance in this study, e.g., if in a matrix each column represents a feature then the row will be called an instance.

2.3 Dimension Reduction

Dimension reduction is the process of reducing the number of original features under consideration. It is a concept originated from the domain of statistics where large amount of data results in information overload, often known as *the curse of dimensionality* [4–6] in literature. Traditional algorithms used in machine learning and PR are susceptible to this problem where the performance of an algorithm decreases as the number of features increase. To deal with this problem dimension reduction methods are often applied as a pre-processing step to reduce the complexity [7].

High dimensional data poses many challenges to data analysis. One of the main problem with high dimensional data is that, not all the features are *important* to understand the underlying problem of interest. Some of the features might be correlated resulting in redundancy. Some features might be irrelevant and might contribute noise to the intrinsic (true) features. Noisy or irrelevant features will have a negative impact on the final classification accuracy. Moreover, the high dimensional data is computationally expensive. For many machine learning algorithms the training time will increase directly (sometimes exponentially) with increase in the number of the features.

Mathematically the problem can be stated as follows: given the m-dimensional dataset $x = (x_1, x_2, ..., x_m)^T$, find a lower dimensional representation of this dataset $y = (y_1, y_2, ..., y_n)^T$, such that $(n < m)$. This lower dimensional representation should capture the important contents of the original data according to some criterion.

Dimension reduction methods can be broadly divided into two main categories. In the first category, a subset of the existing dataset is selected (feature selection) according to some criterion and in the second category the existing features are transformed into a new reduced set of features (feature generation) capturing maximum information from the original features.

2.4 Feature Selection

The aim of feature selection process is to select a subset of the original dataset such that maximum information is retained regarding the discrimination between different categories (classes). The main function of any feature selection method is to identify the importance of different features with regard to a particular problem. In terms of classification, important (rich) features are the features having large inter-class variance and small intra-class variance. Feature selection is a crucial step in the design and the selection of poor features would make the job tough for the subsequent classifier. On the other hand, the selection of rich features would simplify the job of the classifier [8].

There are two main approaches for feature selection, wrapper approach and filter approach. In wrapper-type approach a classifier is used for selection and the results of the classifier (classification accuracy) are used to evaluate the feature. This approach usually results in better performance compared to the filter approach since there is a specific interaction between classifier and features. However, it is computationally expensive and can result in over fitting. Moreover, the solution presented by this approach lacks generality and the selected feature set will be specific to the classifier under consideration. In Filter-type approach, a feature is evaluated using the intrinsic characteristics of the data, e.g., correlation, mutation, etc. Filter-type approach is relatively simple and demands less computation for evaluation. Moreover, its solution is more general and exhibits good performance for a large family of classifiers since it is not tied to the bias of a specific classifier. The only disadvantage is that because of its monotonic nature, it has a tendency to select a large subset. Both of these approaches have been used in this thesis.

2.4.1 Hypothesis Testing

The first step in feature selection is to test the capability of each feature in discriminating the given classes and a statistical hypothesis is used for this purpose. Hypothesis testing tends to refute a claim about a population based on the sample data. Hypothesis testing is conducted using a pair of hypothesis, namely, the null hypothesis and the alternative hypothesis.

- The null hypothesis H_0: To accept a hypothesis we do not need to prove that it is true, we only need to prove that the other hypothesis is false. A null hypothesis is made to be rejected in order to prove that the alternative hypothesis is right. If we do not have enough evidence to reject the null hypothesis, the alternative hypothesis cannot be accepted.

- The alternative hypothesis H_1: This is the hypothesis that we want to prove and it is opposite to the null hypothesis.

The two hypothesis are set to be mutually exclusive and exhaustive which means if one of them is refuted the other must be true and vice versa. A common format for conducting a hypothesis test is given below:

- H_0: A statement of the null hypothesis is given, e.g., two populations come from the same distribution.
- H_1: A statement of the alternative hypothesis is given such that both hypothesis are mutually exclusive, e.g., two populations come from different distributions.
- The test method and a significance level α is chosen. The test method is used to analyse the data and α is used to accept or reject the null hypothesis. Generally, the test method involves some kind of test statistic, e.g., difference between the means, difference between the probability density functions, etc.
- The test method is conducted and the test score is calculated. Given the test statistic, the probability of obtaining a sample static atleast as extreme as the test statistic by assuming the null hypothesis is true, is called the P-value.
- Based on the test score, the P-value and α, the null hypothesis is either accepted or rejected. The P-value is compared with α to reject or accept the null hypothesis. If it is less than α, the null hypothesis is rejected otherwise accepted. The value of α is set at the start and typical values of α are 0.01, 0.05 and 0.1.

2.4.2 Student's t-test

In 1908 William Sealy Gosset, an Englishman and a chemist with a pseudonym ”student” developed a hypothesis called student's t-test (also called t-test). There are many variations of t-test and some of them are given below

- one-sample t-test.
- slope of regression line.
- Two-sample t-test with
  - equal sample size and equal variance.
  - unequal sample size and equal variance.
  - unequal sample size and unequal variance.

In this thesis only two-sample t-test with unequal sample size and unequal variance is explained and details about the other variations are available in [9]. The t-test tests whether the mean of two groups are statistically different from each other. The following assumptions are made for the t-test.

- Each group is considered to be sampled from a distinct population.
- The response in each group is independent of the response in the other group.
- Each group is drawn from a normal population.
- The population size is atleast ten times larger than the sample size.

Let x_i, $i = 1, 2, ..., n_1$, be the sample values for class C_1 with mean and variance μ_1, v_1 respectively and y_i, $i = 1, 2, ..., n_2$, be the sample values for class C_2 with mean and variance μ_2, v_2 respectively. The steps of the t-test are given below,

(a) State the hypotheses

  - The null hypothesis : H_0: $\mu_1 = \mu_2$
  - The alternative hypothesis: H_1: $\mu_1 \neq \mu_2$

(b) Calculate the t-value using the formula given below

$$t = \frac{|\mu_1 - \mu_2|}{SE} \tag{2.1}$$

where SE is the standard error and can be calculate as

$$SE = \sqrt{\frac{v_1}{n_1} + \frac{v_2}{n_2}} \tag{2.2}$$

(c) The P-value can be calculated from the t-distribution which is the probability distribution function of the test value by random sampling and is given as

$$f(t, v) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi}\Gamma(\frac{v}{2})} \left(1 + \frac{t^2}{v}\right)^{-\frac{v+1}{2}} \tag{2.3}$$

where Γ is defined as

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \tag{2.4}$$

Symbol v represents the number of degrees of freedom and is given as

$$v = \frac{(v_1/n_1 + v_2/n_2)^2}{(v_1/n_1)^2/(n_1-1) + (v_2/n_2)^2/(n_2-1)} \tag{2.5}$$

The symbol v must be rounded off to the nearest integer. If the two means are equal then the probability of difference of the two sample means, being as large as t is the cumulative of the t-distribution functions over the two tails. This probability is the P-value, and can be calculated as

$$p = \int_{-\infty}^{-t} f(t,v)dt + \int_{t}^{\infty} f(t,v)dt \tag{2.6}$$

The P-value can also be calculated from t-distribution table, available on the internet and in most statistics books.

2.4.3 Kolmogorov-Smirnov Test

Kolmogorov-Smirnov (K-S) test is also a hypothesis test [10], used to decide if the data samples come from a population with a specific distribution. It is a non-parametric test and makes no assumption about the distribution of the data. It can be categorised into one-sample and two-sample K-S tests. In a one sample K-S test, distribution of the test sample is compared with a reference distribution, while in a two sample K-S test, distributions of two samples are compared with each other. Let x_i, $i = 1, 2, ..., n_1$, be the sample values for class C_1 with cumulative distribution function (CDF) $E_1(x)$ and y_i, $i = 1, 2, ..., n_2$, be the sample values for class C_2 with CDF $E_2(x)$. The steps of the two sample K-S test are given below:

(a) State the hypothesis

  - The null hypothesis: the two samples come from the same distribution, $H_0 : E_1(x) = E_2(x)$.
  - The alternative hypothesis: the two samples belong to different distributions, H_1: $E_1(x) \neq E_2(x)$

(b) The K-S statistic D, the maximum difference between the two CDFs, is calculated using the following formula,

$$D = \max_{-\infty<x<\infty} |E_1(X) - E_2(X)| \tag{2.7}$$

(c) The significance level of the test statistics D, is calculated. The null hypothesis is rejected if the significance level is greater than a critical value, obtained from the table. More details about how to calculate the significance level are available in [9].

An important feature of the K-S test is that the distribution of the K-S test statistic does not depend upon the underlying distributions being tested. While the t-test tests only the significance of the difference of means, the K-S test is sensitive to difference throughout the entire scale. Despite the advantages, the K-S test has certain limitations. It can only be applied to continuous distributions and tends to be more sensitive near the centre of distribution than at the tails.

2.4.4 Kullback-Leibler Divergence

Kullback-Leibler divergence (KLD) (also known as relative entropy) is a method which measures the difference between two probability distributions. For two probability distributions p and q, KLD can be calculated as

$$KLD = \sum_{x} p(x)\; log \frac{p(x)}{q(x)} \tag{2.8}$$

KLD is asymmetric and non negative. The KLD value tells us how well the two distributions are separated and a bigger KLD value means a better feature. A value of 0 means p is equal to q.

2.4.5 F-Score Selection

F-score is a simple feature selection method which quantifies the ability of a feature in discriminating two classes. Given an ith feature vector $x_{k,i}$ where $k = 1, 2, ..., n$. If the number of instances of the positive class and negative class are n^+ and n^-, then the F-score value of this ith feature can be calculated as

$$F(i) = \frac{(\bar{x_i}^{(+)} - \bar{x_i})^2 - (\bar{x_i}^{(-)} - \bar{x_i})^2}{\frac{1}{n^+-1}\sum_{k=1}^{n^+}(x_{k,i}^{(+)} - \bar{x}^{(+)})^2 + \frac{1}{n^--1}\sum_{k=1}^{n^-}(x_{k,i}^{(-)} - \bar{x}^{(-)})^2} \tag{2.9}$$

In equation (2.9) the numerator shows the discrimination between the positive and the negative class while the denominator shows the variation within each class. Here $\bar{x_i}^{(+)}$ is the mean of the positive class for feature i, $\bar{x_i}^{(-)}$ is the mean of the negative class for the same feature, $\bar{x}_i$ is the mean of the i^{th} feature (including both the positive and the negative class), $x_{k,i}^{(+)}$ is the k^{th} value of the i^{th} feature for the positive class and $x_{k,i}^{(-)}$ is the corresponding value for the negative class.

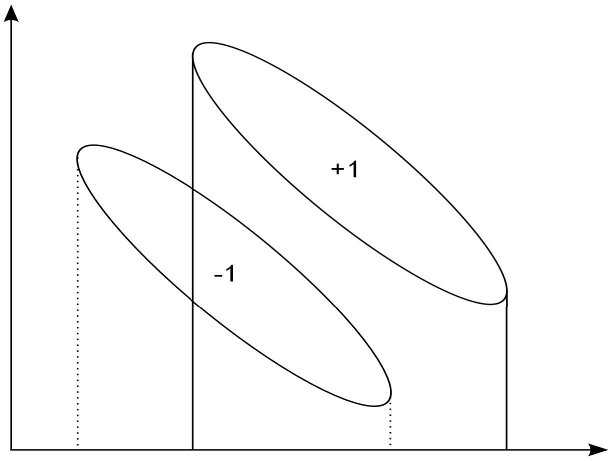


Figure 2.4: Two independent classes having low F-score value

F-score value of each feature is calculated using equation (2.9) and the features having higher F-score value are considered to be more effective in discriminating the positive and the negative class.

A disadvantage of the F-score selection is its inability to take into account the mutual information between two classes [11]. Figure 2.4 shows a simple example demonstrating this inability of F-score selection. In the Figure, although the two classes are independent, the F-score value is still low. This is because the denominator in equation (2.9) is much larger than the numerator, resulting in a low F-score value. Despite this disadvantage, the F-score method is simple, computationally inexpensive and generally quite effective. The steps of the feature selection, using F-score value can be described as:

- Calculate F-score value of each feature using equation (2.9).
- Calculate a threshold value by taking the average of F-score values of all the features.
- The features with F-score value greater than the threshold value are selected and the features with F-score value less than the threshold value are removed from the feature space.

2.4.6 Sequential Forward Selection

This is one of the simplest greedy search algorithms. It starts with an empty set and sequentially adds the feature x^+ such that the objective function $F(Y_k + x^+)$ is maximised. The choice of the objective function depends on the nature of the problem and in this thesis this function is the classification accuracy. The best feature (often recommended by feature selection methods) is added first and the performance is evaluated using a classifier. The second best feature is added next and the performance is evaluated again. This process continues until there is no further improvement in the performance. The steps of this process are given below

(1) Start with an empty set $Y_0 = \phi$.

(2) Add the next best feature x^+ such that $\max_{x \notin Y_k}[F(Y_k + x^+)]$.

(3) Update the dataset, $Y_{k+1} = Y_k + x^+, k = k + 1$.

(4) Go back to step 2.

The sequential forward selection method performs best when the optimal feature set is small. The main drawback of this method is its inability to remove features which become obsolete after the addition of new features. Sequential stepwise forward selection is a method designed to overcome this drawback. In this method, variables once entered may be dropped if they are no longer significant once new variables are added. In this study only sequential forward selection method is used.

2.4.7 Sequential Backward Elimination

This method works in the opposite direction of sequential forward selection. It starts with the full dataset and sequentially removes the feature x^- such that there is smallest decrease in the objective function $F(Y_k - x^-)$. The removal of a feature may actually result in an increase in the objective function. The steps of this algorithm are given below

(1) Start with full dataset $Y_0 = X$.

(2) Remove the worst feature x^- such that $\max_{x \in Y_k}[F(Y_k - x^-)]$.

(3) Update the dataset, $Y_{k+1} = Y_k - x^-, k = k + 1$.

(4) Go back to step 2.

The sequential backward elimination method performs best when the optimal feature set is large. The drawback of this method is its inability to reselect a feature once it has been removed. Sequential stepwise backward elimination is designed to overcome this drawback. In this method a removed variable may be added back to dataset if later it appears to be significant.

2.5 Feature Generation

Feature generation is a powerful term in the domain of PR and sometimes it can also include the concept of feature selection. The process of transforming the original set of features to a new reduced set of features (by making combination of the original features) such that the underlying classes are more separable is called feature generation.

In the domain of image processing and signal processing the term feature generation is sometimes used slightly differently, where it is used to describe the phenomenon of extracting features from the original image or signal. Instead of using the full size input, features which are reduced representation of the actual input are extracted from the input. These features are expected to contain enough relevant information to perform the desired task. This process is similar to the other definition in the sense that both of these methods are trying to find a reduced representation of input. The ultimate objective of the feature generation is same in all domains i.e. to reduce the amount of resources required to perform the desired task. The increase in the amount of resources required, with an increase in input dimensions has been described previously as *the curse of dimensionality*. Although, there has been an exponential growth in the computing power in recent years, the benefits offered by feature generation in terms of reducing computational complexity cannot be ignored. Moreover, the improvement in performance offered by feature generation is invaluable.

There are quite a few feature generation methods available in the literature. They can be categorised into different groups using different criteria. For example, they can be categorised as linear or non-linear based on the transformation function used, depending on whether the class information is available or not, they can be categorised as supervised or unsupervised, etc. Linear feature generation methods are relatively simple and provide analytical solutions for most of the problems. A linear solution, if possible, is always preferred over non-linear solution because of its simplicity. In this thesis both linear and non-linear feature generation methods are used for transformation of original features.

2.5.1 Principal Component Analysis

Principal component analysis (PCA) is one of the simplest and most popular feature generation method invented by Pearson in 1901 [12]. The purpose of PCA is to linearly transform the original set of possibly correlated features to a new set of linearly uncorrelated features, such that the new features are orthogonal to each other. The new set of features are called the principal components. The assumption made in PCA is that most of the information is along the dimensions where variance is the largest. The first principal component covers most of the variance, followed by the second principal component and so on. PCA does not take into account the class label of the features and is essentially an unsupervised technique. Since this study is not the theoretical study of statistics, only the methodology of PCA will be presented and interested readers can consult [1] for more details.

Given a m by n data matrix X, where each column represents an observation and each row represents a feature, the steps for calculating principal components are given below

1. **Subtract the mean:** For PCA to work properly, each feature should be zero mean which can be achieved by by subtracting mean of each feature from each feature element.

2. **Calculate the covariance matrix:** The symmetric covariance matrix of size m by m, can be calculated from the data matrix X, using the following equation

$$C_{m\times m} = \sum_{i=0}^{n-1} (x_i - \mu_i)(x_i - \mu_i)^T \tag{2.10}$$

   where x_i is the i^th feature vector and μ_i is the mean of ith vector.

3. **Calculation of Eigenvalues and Eigenvectors:** The eigenvalues and eigenvectors are calculated from the covariance matrix, calculated at the second step.

4. **Sorting:** The eigenvectors are sorted using the order of eigenvalues (in a descending order).

5. **Transformation of original matrix:** The original matrix is transformed into new matrix by multiplying eigenvector or a set of eigenvectors with

original matrix. The first eigenvector (corresponding to highest eigenvalue) will cover the maximum variance in the new feature space.

6. **Selection of principal components:** Since most of the data variance will be covered by first few principal components, last few principal components (transformed feature vectors) can be discarded. A famous criterion to decide, which principal components to discard, is the eigenvalue-one criterion [13]. Each feature contributes one unit of variance to the total variance of the dataset. Any feature having an eigenvalue greater than one covers more variance than the variance of one feature. Such a feature is considered more valuable than others and is worthy of being retained. On the other hand a feature with eigenvalue less than one, covers less variance than the variance of one feature. Such a feature is considered trivial and is removed from the new transformed features.

PCA is a non-parametric dimensionality reduction method since it does not require any priori knowledge about the data probability distribution. This is considered a strength as well as weakness of PCA. It requires less information about the problem in question (a strength) but on the other hand it can result in over fitting the problem when limited number of data samples are available (a weakness). It is important to point out that PCA is based on the following assumptions:

1. The dimensionality of the data can be efficiently reduced by linear transformation.

2. Most of the information is contained in the directions where data variation is maximum.

These two assumption are by no means always met. For example, if the data points are placed on the surface of hypersphere, linear transformation (for dimension reduction) cannot cope with it. Similarly, the direction having maximum variation, do not necessarily contain maximum information. This is particularly true when signal-to-noise ratio (SNR) is low, where noise is also viewed as useful variance. Moreover, for classification problems, PCA does not take into account class labels and the direction of maximum variation may not help in class discrimination.

Nevertheless, PCA is a basic and popular method for feature generation. It is fast and effective when certain conditions are met.

2.5.2 Genetic Programming

Genetic programming (GP) is an evolutionary computation method based on Darwinian model of natural evolution. In this thesis it has been used for feature generation. Since the main work of this thesis is based on GP and a full chapter (Chapter 3) is devoted to GP, it will not be discussed in this chapter.

2.6 Classification Methods

The assignment of class label to a set of unclassified objects, described by a set of features is called classification. Based on the methodology, classifiers can be divided into three main categories [14],

- The similarity approach is one of the simplest approaches. In this approach the patterns similar to each other are assigned the same class, e.g., template matching, nearest neighbour rule, etc.
- The probabilistic approach, e.g., Bayesian decision rule, maximum likelihood approach, k-nearest neighbour rule, Parzen classifier, etc.
- The decision boundary approach, e.g., Fisher's linear discriminant analysis, the decision tree approach, Support vector machines (SVMs), etc.

The aim of any classifier is to differentiate between different classes by finding the similarities and differences in the underlying patterns. Depending on the type of classifier it might need a learning or training phase. If a classifier needs training, the parameters of the classifier are adjusted during the training phase in order to facilitate the discrimination between different classes. The characteristics of different classes are used to adjust these parameters. Inter-class variance and intra-class variance are two important characteristics used to adjust the parameters of classifier.

The performance of the classifier heavily depends on the input features fed into the classifier. In any classification problem, either the feature generation process maximises the discriminatory information and presents the transformed feature to the classifier, or the original feature are directly fed into the classifier and classifier searches for the optimal boundary between the classes. Feature generation makes the job easier for the classifier since the generated features are more rich in discriminatory information. If the classifiers directly takes the original data, the final performance depends on the discriminatory information present in the original features. If the discriminatory information provided by the original features is poor, employing a powerful classifier will not improve the performance significantly. As a general rule, the more discrimination information present in the input features, the less powerful classifier required and vice versa.

Different classifiers used in this study including k-nearest neighbour, SVM and ANN are described in the next section. The aim is to provide an overview of all these classifiers, describing their strengths and weaknesses.

2.6.1 K-Nearest Neighbour Rule

In 1951, Fix and Hodges introduced a non-parametric method for pattern classification, later known as the K-nearest neighbour (KNN) rule [15]. It is one of the simplest and fundamental classification methods, requiring little or no prior information about the distribution of the data.

It assigns class label to a test sample based on the majority vote of the closest k neighbours, with known class labels. The nearest neighbours are determined using a distance measure (commonly Euclidean distance) between the test sample and the training samples. This method does not need any training phase and training examples are directly used at the run time. Since the training examples are required at the testing phase, they need to be stored in the memory at run time.

Let x_i be the training samples with d features $(x_{i1}, x_{i2}, ..., x_{id})$, with total number of samples n_1 $(i = 1, 2, 3, ..., n_1)$. Let x_t be the test samples $(x_{t1}, x_{t2}, ..., x_{td})$ with total number of samples n_2 $(t = 1, 2, 3, ..., n_2)$. Let ω be the true class of each sample and $\bar{\omega}$ be the predicted class of any test sample with $(\omega, \bar{\omega} = 1, 2, 3, ..., \Omega)$, where Ω is the total number of classes. The process of predicting the class of a test sample can be described as,

1. Calculate the distance of test sample from each training sample using the following formula,

   $distance(x_t, x_i) = \sqrt{(x_{t1} - x_{i1})^2 + (x_{t2} - x_{i2})^2 +, ..., +(x_{td} - x_{id})^2}$

2. Find out the k neighbouring training samples using the distances calculated at step 1.

3. The predicted class $(\bar{\omega})$ of the test sample is set equal to the most frequent true class (ω), among k nearest neighbour training samples.

The computational complexity of this method is mainly in the testing phase and it is of order $O(dn_1)$, where d is the total number of features and n_1 is the total number of training samples.

The value of k plays a vital role in the final classification accuracy. Generally, choice of k depends on the dataset used. A large value of k reduces the effect of noise but makes the boundaries between different classes less distinct and increases the computational complexity. For a binary classification problem k is chosen to be odd, and in general k is chosen not to be a multiple of the number of classes Ω. A special case where $k = 1$ is known as *nearest neighbour rule*, where the class of the test sample is simply equal to the class of nearest neighbour.

The main disadvantage of KNN is the possible bias towards one class when the number of training samples in the underlying classes are not same. A test sample is more likely to be classified as a class with larger number of training samples. The above issue will not arise if the classes are well separated which is not the case in real world problems. A large training dataset also poses problems for KNN classifier. Not only a large memory would be required to store such a dataset, the computational complexity would increase as well since the distance of test sample needs to be calculated from all the training samples.

2.6.2 Support Vector Machines

The concept of Support Vector Machines (SVMs) was first introduced by Vapnik in 1968 [16]. SVMs are a group of supervised learning methods that can be applied for classification or regression. In order to separate two classes, SVM constructs a hyper-plane in n-dimensional space which has the largest distance to the nearest point of two classes. The nearest points from both the classes are called *support vectors* since the hyperplane is determined by these points only, instead of the full dataset. The hyperplane is created in such a way that the distance between the two classes is maximised.

Let x_i, be the training samples with D dimensions, $(x_{i1}, x_{i2}, ..., x_{iD})$, with two underlying classes $y_i = \{1, -1\}$. Assuming that the data is linearly separable, a line can be drawn to separate two classes when $D = 2$, and a hyperplane can be used to separate classes when $D > 2$. A simple example for $D = 2$ is shown in Figure 2.5 where grey circles and black squares represent two classes..

The hyperplane can be described by the following equation $wx + b = 0$, where w is a vector normal to the hyperplane, b is a scalar threshold and $\frac{b}{\|w\|}$ is the distance from hyperplane to the origin. The aim of SVM is to select w and b such

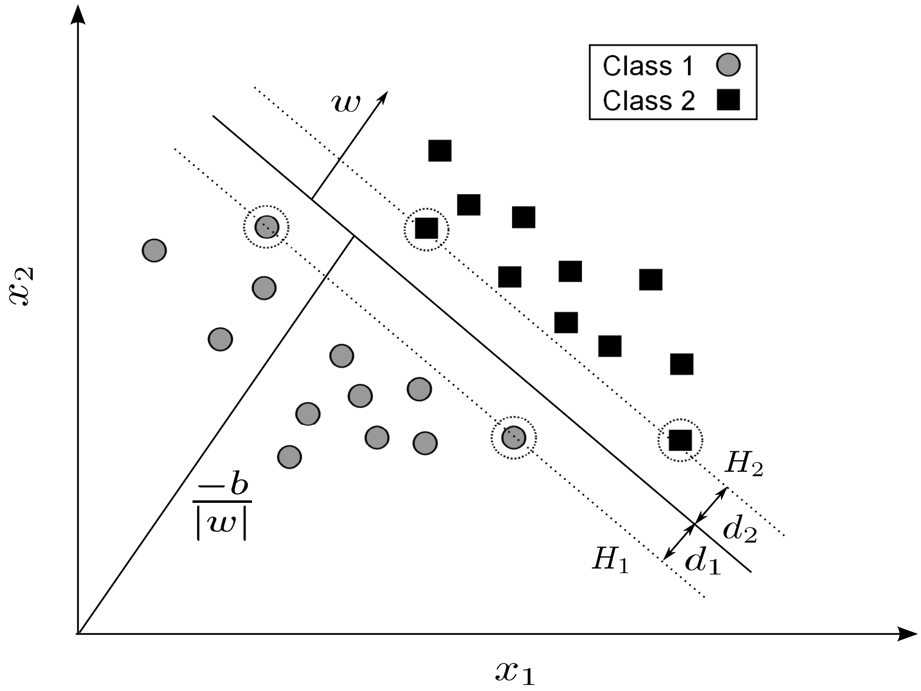


Figure 2.5: Two linearly separable classes.

that,

$$wx + b \geq 1 \quad for \quad y_i = +1 \tag{2.11}$$

and

$$wx + b \leq -1 \quad for \quad y_i = -1 \tag{2.12}$$

The points close to the separating hyperplane (shown inside dotted circles in Figure 2.5) are called the support vectors and the two hyperplanes (H_1 and H_2) on which these points lie can be described as,

$$x_i.w + b = +1 \quad for \quad H_1 \tag{2.13}$$

$$x_i.w + b = -1 \quad for \quad H_2 \tag{2.14}$$

The role of SVM is to find a linear boundary that maximises d_1 (distance of H_1 form the hyperplane) and d_2 (distance of H_2 form the hyperplane). The quantities d_1 and d_2 are also known as *SVM margins.*

The above mentioned idea is suitable for linearly separable classes, and for non-linear problems the condition of the hyperplane can be relaxed to allow some data points to cross the boundary. In the relaxed environment, where some of the data points are allowed to cross the hyperplane, the *SVM margins* are known as *soft margin*. Kernel method can also be used for non-linear problems, to find an optimal surface maximising the margin to each class samples.

The SVM covered above is suitable only for binary classification problems. Multi-class classification can be achieved using SVM in two ways: using a multi-class SVM, where a single output can be used to give different output levels for different classes [17], or by dividing a multi-class problem into several binary classification problems, each of which can be handled by a binary classifier. In this study SVM has been used only for binary classification problems with Kernel trick.

2.6.3 Artificial Neural Networks

Artificial Neural Networks (ANNs), also known as Neural Networks (NNs) are inspired by a mathematical model of biological neural networks [18, 19]. ANN is inspired by the design and functioning of brain. The concept of ANN was first introduced in mid-40s and it is widely used today in PR, blind source separation, filtering, image processing, medical diagnosis, process control, etc. ANN is able to solve complex non-linear input-output relationships that are difficult for other techniques. It is an adaptive system that learns and changes its structure as the input is fed to it.

The basic layout of ANN consists of many neurons and these are linked together according to a specific network structure. There are different architectures of ANN depending on the number of layers and the flow of information. Based on the learning paradigm it can be divided into supervised and unsupervised ANN and on the basis of network structure it could be single layer, multi layer. A supervised ANN is used when the desired output is already known while an unsupervised ANN is used when there are no target outputs. A supervised ANN model will be used in this study since the training data is available.

Generally, the operation of ANN consists of a training and a testing phase. Sometimes, depending on the training methodology used, a validation phase is also required. During the training phase, the training data is presented to the network and a desired response is set at the output. A training error is calculated using the difference of the desired response and the actual response. This error is fed back to the network and different parameters of the network are adjusted adaptively (learning rule) until the desired output is acceptable. If the validation phase is present, a separate set of validation data is used to check the generality of the classifier at each training iteration. A validation error is calculated using the validation data.

If the validation error continues to increase for a pre-defined period, even though the training error is decreasing, the network is said to be over fitting the test data. If this situation arises, the training is process is interrupted and the parameters of the network are reverted to the values that gave the smallest validation error (also called generalisation via early stopping). After the completion of training and validation phase, the performance of the trained network is tested using the test data. A well trained classifier should perform well for all three datasets (training, validation and testing).

Depending on how an ANN partitions the data into different classes it can be divided into three main types: Multi Layer Perceptron (MLP), Radial Basis Function Network (RBF) and Probabilistic Neural Network (PNN). In this study only the first type is discussed and interested readers can read details about other types in [18].

The Multi Layer Perceptron

One of the most common realisations of the neural network is the Multi Layer Perceptron (MLP). A typical multi-layer network consists of computing units called neurons, which make a network consisting of an input layer, one or more hidden layers of computation nodes and an output layer. Inputs propagate through the network, layer by layer, and a non-linear mapping of the inputs is produced by MLP at the output layer. Figure 2.6 shows a basic structure of MLP network. The general formula for any neuron can be written as,

$$y_j = \varphi(v(\mathbf{x})) = \varphi\left(\sum_{i=1}^{N}(w_{ji}x_i + b_j)\right) \tag{2.15}$$

where

$$v(\mathbf{x}) = \sum_{i=1}^{N}(w_{ji}x_i + b_j) \tag{2.16}$$

and N is the number of inputs of neuron, w_{ji} is the weight of neuron connecting the output of neuron i to neuron j , b_j is the bias attributed to neuron j, $\phi(.)$ is the activation function describing input output relationship and y_j is the output of the neuron j. In Figure 2.6 there is one hidden layer of neurons and one output layer. The activation function for both of these layer are different and can be written as,

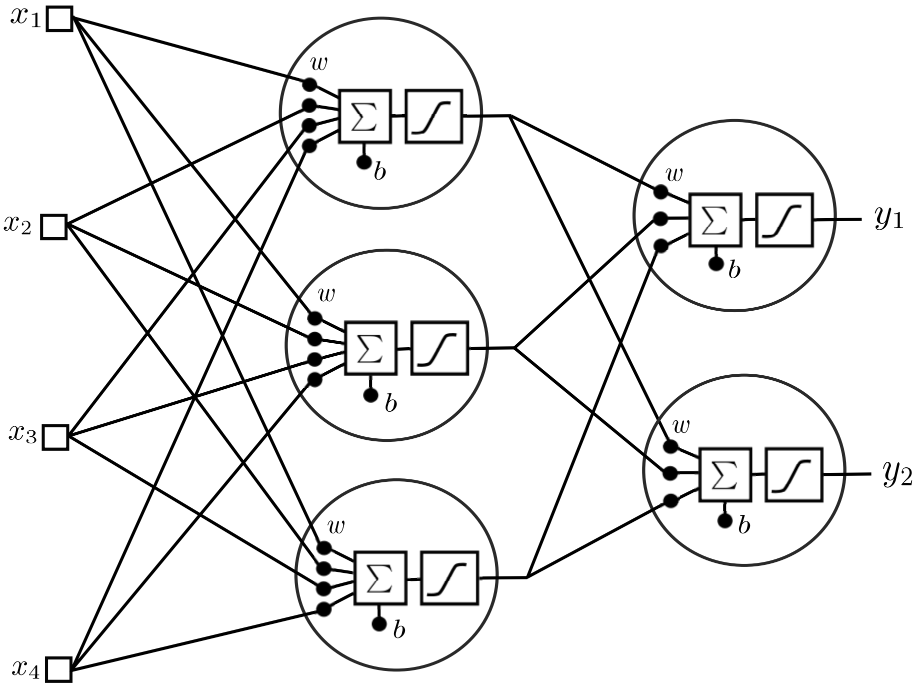


Figure 2.6: A simple MLP network.

$$\varphi(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}} \tag{2.17}$$

and

$$\varphi(v) = v \tag{2.18}$$

where a Tan-sigmoid activation function is used for the hidden layer (equation (2.17)) and a linear activation function is used for the output layer (equation (2.18)). The size of the output layer is equal to the number of outputs while the size of the hidden layer is specified by the user. Choosing a right size for the hidden layer is very important in designing any MLP network, since it has a big impact on the classification performance and the generality of the network. Hidden layer with a large size can result in over training (starts remembering the training dataset), while a size too small will result in poor classification performance.

Training Algorithm for Multi Layer Perceptron

A back-propagation (BP) algorithm has been used for training MLP, which uses the difference between the actual and the desired output (*the error function*) as a cost function, to adjust the weights of neurons [20]. The error signal at output node j for iteration n can be written as,

$$e_j(n) = d_j(n) - y_j(n) \tag{2.19}$$

where $d_j(n)$ is the desired output value and $y_j(n)$ is the actual output value. If the instantaneous energy for neuron j is $\frac{1}{2}e_j^2(n)$, the total instantaneous energy can be calculated as,

$$E(n) = \frac{1}{2}\sum_{j=1}^{J} e_j^2(n) \tag{2.20}$$

A gradient descent algorithm is used in this study during back-propagation to minimise the cost function. The weights of the neurons can be adjusted using the following steps,

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial \varphi_j(n)} \frac{\partial \varphi_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.21}$$

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{2.22}$$

$$\frac{\partial e_j(n)}{\partial \varphi_j(n)} = -1 \tag{2.23}$$

$$\frac{\partial \varphi_j(n)}{\partial v_j(n)} = \acute{\varphi}(v_j(n)) \tag{2.24}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_j(n) \tag{2.25}$$

Putting equations (2.22) to (2.25) in equation (2.21), yields

$$\begin{aligned} \frac{\partial E(n)}{\partial w_{ji}(n)} &= -e_j(n)\acute{\varphi}(v_j(n))y_j(n) \\ &= -\delta_j(n)y_j(n) \end{aligned} \tag{2.26}$$

The weight update (correction) of neuron can be written as,

$$\Delta w_{ji} = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{2.27}$$

$$\Delta w_{ji} = \eta \delta_j(n) y_j(n) \tag{2.28}$$

where

$$\delta_j(n) = e_j(n)\acute{\varphi}(v_j(n)) \tag{2.29}$$

Here η is the learning rate parameter of BP algorithm and $\delta_j(n)$ is the local gradient for neuron j. Finding a minimum in space using BP algorithm is quite fast but does not always result in actual global minimum. Sometimes it gets stuck in local minimum. Several strategies are present in the literature to reduce the likelihood of getting stuck in local minimum and interested readers can consult [21–24].

2.7 Summary

This chapter gives an overview of structural organisation of a general PR system. The basic building blocks of any PR system with approaches to design these blocks is presented. Feature selection and generation (key components of any PR system) techniques used in this thesis are discussed in detail. An introduction to various statistical feature selection tests including Student's t-test, Kolmogorov-Smirnov test, Kullback-Leibler divergence test and F-score selection is presented. Two popular methods Sequential Forward Selection and Sequential Backward Selection used to assist the dimension reduction process are also discussed. An introduction of the PCA, a well known feature generation method is presented. Finally the basic principles of different machine learning classifiers, KNN, SVM and ANN are outlined. In the following chapters these classifiers will be used to evaluate the selected or generated features.

Chapter 3

Genetic Programming

In the previous chapter the basic components of any PR system were discussed. One of the key component was feature generation used for dimension reduction. In this study genetic programming (GP), a machine learning methodology has been used for feature generation. GP offers feature selection and transformation both at a single platform which makes it superior to other methods. Since the work in this thesis is based on using GP for feature generation, a full chapter is devoted to GP. This chapter presents a general overview of GP algorithm.

The chapter is organised as follows: the layout of the GP algorithm is presented in the beginning showing different stages of the algorithm and different parameters of the algorithm are discussed in detail one by one. GP has been used in this research to generate features for classification tasks so the focus has been on the classification ability of GP. Different parameters of GP and their possible values and designs are discussed with respect to classification tasks. Creation of *super-feature* by GP and using it for classification purpose has been discussed. Finally, this chapter is concluded with a simple classification example, presented to test the effectiveness of *super-feature*.

3.1 Evolutionary Computation

Evolutionary Computation (EC) is a general term used for several computational techniques which are based on Darwinian model of natural evolution. EC started with use of genetic algorithms (GA), evolutionary strategies (ES) and evolutionary programming (EP). All EC methods work by setting a goal at the start of evolution and then use this goal to compare the quality of candidate solutions during evolution. If this process is successful, EC results in optimal or near optimal solution after a number of generations. In 1992 Koza [25] introduced the idea of GP in which he reported that first experiments in the field of GP were performed in 1980. GP follows the same basic principles of other EC methods.

The main thrust of all EC methods is the idea of evolution. The idea is to write a program which will evolve towards a certain goal where the goal could be as complex as solving an engineering problem or as easy as winning a toy game. All the EC techniques represent a variety of solutions which can be applied to a diverse range of problem domains.

3.2 Genetic Programming

GP belongs to class of EC methodologies in which computer programs are evolved based on Darwinian model of natural evolution. These computer programs compete with each other during the process of evolution and the best program wins this competition. The ultimate goal of GP is to find a computer program capable of solving a predefined problem. The best program is the actual solution of the predefined problem.

Since the dawn of computer age computer scientists have been trying to find a way to train the computers so that they can automatically find solutions to problems i.e. to make such computers which can do something without telling them explicitly how to do it. But the simple reality is if we want the computer to explicitly do anything without being told, the structure we need are computer programs. One of the difference between GP and other methodologies is that except GP all other methodologies seek solution in some other form instead of computer programs. They use some specialized structures, e.g., decision trees, clusters, chromosome strings and weight vectors, etc. These structure facilitate in finding the solution of a problem but they do not have the flexibility of programming computers as computers are not programmed in the form of weight vectors, decision trees, etc.

Representation of solutions in the form of computer programs offers many advantages. Computer programs can perform operations in a hierarchical way, can find alternative solutions based on the outcome of intermediate calculations, can perform recursions, iterations and can take variables of different types as inputs. In addition, the complexity or size of GP solutions is decided by GP according to the demand of a problem during evolution, and we do not have to define it in advance as done in some other methodologies. Due to this flexibility of GP, the search space of possible solutions becomes too vast and we need to search in an intelligent and adaptive way to reduce training time. Some advantages that GP

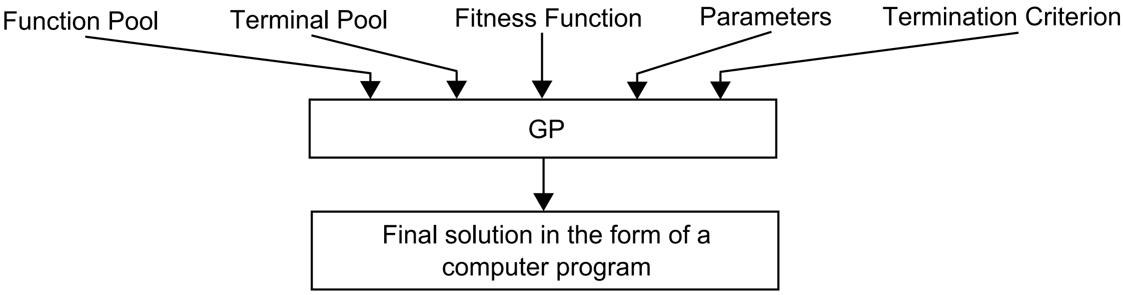


Figure 3.1: Basic version of GP

have on other machine learning methods are listed below.
(a) No prior knowledge about the statistical distribution of data is needed.
(b) Pre-processing of data is not required and data can be used directly by GP in its original form.
(c) GP returns mathematical function as output which can be used directly in application environment.
(d) GP has the inherent capability to select useful features and ignore others.

3.2.1 The Algorithm

The process of GP in its basic version is shown in Figure 3.1. One can see in the Figure that process of GP depends mainly on function set, terminal pool, fitness function, control parameters and termination criterion. Function set contains the functions that will be used by GP solutions, terminal pool contains all the input features, fitness function measures the ability of individual to solve the given problem, control parameters include population size, probabilities of genetic operators, maximum depth of solutions, etc., and termination criterion decides how would the GP training process end. All of these parameters of GP are explained in a later section. A typical GP process can be explained by the following equation.

$$g_{t+1} = g_o(f(g_t)) \tag{3.1}$$

where g_{t+1} is the new generation being created, g_t is the current generation, function f chooses the fittest individuals from the current generation and the function g_o applies genetic operators on the current generation to create the next generation. Typically GP implementation follows the following steps:
(a) GP starts with a randomly generated initial population (generation 0) of user

defined size. This population of individuals (computer programs) is composed of available functions and terminals.
(b) Each individual is assigned a fitness value which represents the strength of the individual to solve the given problem. The fitness value is calculated using user defined fitness function.
(c) New individuals are created by applying genetic operators (crossover, mutation or reproduction) on current generation.
(d) Fitness value for new individuals is calculated and the individuals having better fitness values make their way to the next generation.
(e) If the termination criterion is satisfied then the best individual in the population is returned as solution, otherwise steps (c) and (d) are repeated until termination criterion is satisfied.

Figure 3.2 gives a flow chart showing important steps of GP process. In order to generalize the performance of GP, a GP process is repeated many times to get an average performance. The variable Run used at the start of GP process in Figure 3.2 is used for this purpose. The process starts from initial random generation (Gen = 0) of individuals composed of terminals and functions. The initial random individuals are of different size (have different number of functions and terminals). The variable Gen represents generation number. Termination criterion is checked after creation of each generation. Termination criterion is satisfied if we reach maximum number of generations set by the user or a solution is found which completely solves the problem. After checking termination criterion fitness values for individuals are calculated which tell us how well an individual performs the given task. Some individuals in the population are more fitter than others and this difference in fitness is exploited by GP. The fitter individuals get transferred to the nest generation while weaker individuals are discarded.

Population of new individuals (offspring) is created by applying genetic operators on some current individuals that are selected based on fitness value. After creation of new generation termination criterion is checked again and this process of creating offspring is repeated until termination criteria is satisfied. Once the termination criterion is satisfied, the best individual for that run is saved. The GP process is repeated for next run until we reach N runs set by user and eventually we get N best individuals, one for each run. The training process of GP is finished at this time. In order to find the quality of learning that has taken place, the best individuals must be tested using test data. A test data set is comprised of inputs and output from the same domain, the system got trained. Although from the same domain, the test data contains different examples than

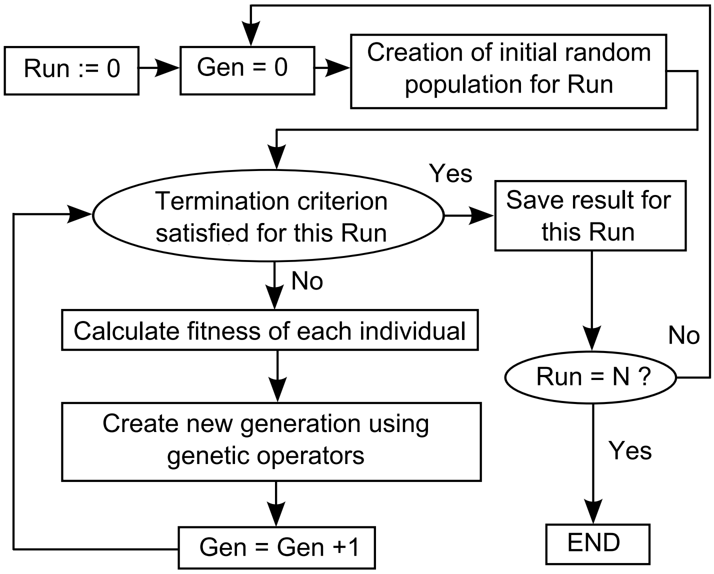

Run := 0
Gen = 0
Creation of initial random population for Run
Termination criterion satisfied for this Run
Yes
Save result for this Run
No
Calculate fitness of each individual
No
Run = N ?
Create new generation using genetic operators
Yes
Gen = Gen +1
END

Figure 3.2: Flow chart of a simple GP

the training data. The ability of system to correctly predict the test data is very important to generalize its ability as it might have just memorized the training data. The performance of the best individual for each run is tested using test data. These performances are averaged to get the mean test performance of GP.

3.2.2 Terminal Set

Terminal set contains all the input variables, constants or functions with an *arity* of zero. They are also called leafs as they terminate a branch of a tree in tree-based GP. The terminal set is also called training set as this is the data using which GP learns. In some problem domains it can include instructions instead of values, e.g., artificial ant problem where inputs are instructions for ant movement. It can include constants which do not change during the run. It can also include function with zero arity, e.g., rand() function which returns a random number. The terminal set is just one of the primitives GP uses, for building solutions and it does not have any fixed place like other machine learning methodologies. At times GP can ignore an input altogether.

3.2.3 Function Set

Function set contains all the functions, operators available to GP. The functions to be included in the function set depend purely on the nature of problem. For example for logical problems, basic logical operations AND, OR, etc., are required for meaningful solution and for non-linear problems non-linear functions are preferred. Some of the functions typically used in GP are given below.

Logical Functions

- *and(x,y): returns 1 if both inputs are one, else 0,*
- *or(x,y): returns 1 if either inputs is 1, else 0,*
- *greater(x,y): returns 1 if* $x > y$*, else 0,*
- *smaller(x,y): returns 1 if* $x < y$*, else 0,*
- *eq(x,y): returns 1 if* $x = y$*, else 0,*
- *neq(x,y): returns 1 if* $x \neq y$*, else 0.*

Arithmetic Functions

- *plus(a,b): adds a and b,*
- *minus(a,b): subtracts b from a,*
- *times(a,b): multiplies a with b,*
- *divide(a,b): divides a by b, returns 0 if b is zero.*

Non-Linear Functions

- *sin(a), cos(a): returns sine and cosine of a,*
- *asin(a), acos(a): returns* $sine^{-1}$ *and* $cosine^{-1}$ *of a,*
- *square(a): returns* a^2*,*
- *sqrt(a): returns* $\sqrt{a}$*,*
- *log(a): returns* $log_e(a)$*,*
- *log10(a): returns* $log_{10}(a)$*.*

Miscellaneous Functions

- *abs(a): converts a to absolute value,*
- *ceil(a): returns the integer above a,*
- *floor(a): returns the integer below a,*
- *round(a): returns the nearest integer to a.*

The range of functions is considerably broader and only some of them are mentioned here. Any function that a programmer can dream of can be part of function set. Although there is a wide range of choice for functions but choosing a function set too large will increase the search space and can make it hard for GP to search for a solution. It is often encouraged to start with smaller simpler function set and increase it later if it does not work well. All these functions can have different number of inputs but their output is always single.

The freedom in choosing a function set often reduces the need to pre-process input data. As the result of one function is to be used by other functions, the functions should be protected against returning an invalid solutions. A typical example is *log* function which can return $-\infty$ if given an input of zero or returns a complex number if given a negative input. This result may not be acceptable by other functions so the original *log* function can be replaced by a protected (*mylog*) fnction as

$$mylog(x) = \begin{cases} 0 & \text{if } x = 0 \\ log(abs(x)) & \text{otherwise} \end{cases} \tag{3.2}$$

Although this protection system can effect the final solution but it is inevitable in some cases.

3.2.4 Program Structures

The best individual returned by GP training process is the solution of the given problem. Normal computer programs are syntactically complex and a little variation from available syntax makes them useless. Combining ASCII characters randomly may generate a working program but is highly unlikely. The program structure in GP should be such that it allows genetic operators to produce meaningful, working programs. Three main structures used for GP are tree, linear and graph structures. Most common of these is the tree structure and the same structure has been used here. A tree has leaf nodes, internal nodes and output node (root). Leaf nodes are the terminals (inputs), internal nodes represent the functions operating on inputs while the output node gives the output of the tree. An example of a tree structure is given in Figure 3.3. This tree implements the following expression $(a + b) - (b \text{ x } c)$. Normally the tree structure is stored in the form of string expression and the tree in Figure 3.3 will be stored as

$$minus(plus(a, b), times(b, c))$$

The tree structure is decomposed into small expressions with each expression working independently. Each expression performs the operation on the data passed to it, guaranteeing closure and passes output to next expression. This structure is ideal for GP as it can produce new generation ensuring it is syntactically viable.

3.2.5 Initializing Population

The initial population in GP is randomly created using functions (function set) and terminals (terminal set). The maximum depth of these initially created individuals is set by the user. There are three famous methods for initialization and they are explained below.

- **Grow Method**
  This method selects nodes from functions and terminals randomly except the root which is selected from functions only. Once a branch contains a terminal it ends there, even if the maximum depth is not reached and this results in unbalanced trees.

- **Full Method**
  Instead of choosing nodes randomly from functions and terminals, full method chooses only from functions until it reaches maximum depth where it chooses from terminals. Every branch of tree goes to full maximum depth which results in a balanced tree.

- **Ramped Half and Half**
  Above methods could result in individuals having uniform structure which means loss of diversity. In order to promote diversity ramped half and half method is used. For each depth size (until the maximum depth) starting from 2, half of the individuals are created using grow method and other half

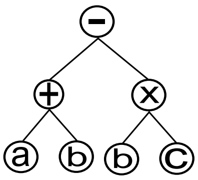


Figure 3.3: A simple tree structure

using full method. The population resulting from this method is diverse, having balanced and unbalanced trees.

3.2.6 Tree Depth

Tree depth needs to be controlled in order to avoid large trees which would increase processing time. There are many ways of doing it. An *initial maximum depth* is the maximum depth allowed to trees in initial population and is normally very small. During the evolution, trees have to follow *dynamic maximum depth* which could be violated only if the individual violating it, is the best individual in the population. *Strict maximum depth* cannot be violated by any individual at any point and if violated, the individual doing this violation is dropped from population.

3.2.7 The Fitness Function

The fundamental premise in GP is ”the fittest individual survives”, so the fitness function is very important parameter of an individual to judge its position in population. **Fitness** *is the measure used by GP during training evolution and tells how well an individual has learned to perform the given task.* The goal of fitness function is to help GP find, which individuals should be given chance to reproduce and multiply and which individuals should be removed from the population. Normally, GP is given a set of predefined inputs and corresponding outputs, also called training examples. GP is asked to find a function that could produce those outputs from given inputs. A typical fitness function is to calculate the sum of the squared differences between the generated and desired output for a number of training examples (squared error). This fitness function is shown below

$$f_p = \sum_{i=1}^{n} (p_i - o_i)^2 \tag{3.3}$$

Table 3.1: A simple training set

Case	Input	output
1	1	2
2	4	20
3	7	56
4	10	110
5	13	182

where f_p is the fitness of individual p, p_i is the output generated by Individual p for example i, o_i is the desired output for example i and n is the size of training set . **Standardized fitness** *is a fitness function in which an individual having a fitness value of zero is considered to be the fittest individual.* The above fitness function is an example of standardized fitness. In this fitness function the lower the fitness value is, the better is the individual. Let us look at a simple example to elaborate it further.

Table 3.1 shows a simple training dataset and we want to find the function satisfying the cases in the Table. GP is to find a program that can predict the output value knowing only the input values. The fitness function of equation (3.3) could be used as a fitness measure for such a problem. It is a fairly simple problem for GP and it easily finds that the solution is $f(x) = x^2 + x$. The process of learning when learning domain comprises of input and outputs is called symbolic regression. The above mentioned fitness function works well for symbolic regression problems but may not work for other problems. Fitness function is problem dependent and needs to be modified according to the requirement.

3.2.8 Selection for Reproduction

After calculating the fitness of the individuals, we need to decide which individuals will take part in genetic operation to create new individuals. There are quite a few methods in literature for selection and some of them are mentioned below.

- **Roulette Wheel Selection**
  Roulette wheel selection, also known as "Fitness Proportionate Selection" associates fitness of individual to its probability of selection. This works like a roulette wheel where individuals having better fitness occupy more proportion of wheel. This is the most common method of selection and the

work in this research uses this method.

- **Tournament Selection**
  A random subset of individuals is chosen (irrespective of their fitness) from the population and the best individuals (in terms of fitness) from the subset are selected.

- **Rank Selection**
  This is similar to roulette selection but instead of fitness as selection parameter, a rank is given to each individual which is used for selection. If only few individual contribute to major part of total fitness then the probability of their selection would dominate in roulette method but this probability is reduced in rank selection.

3.2.9 Genetic Operators

The evolution of any EC methodology proceeds by transforming the initial population using genetic operators.There are many genetic operators used in GP but three principal genetic operators are discussed here.

Crossover

Crossover takes two parents as input and produces two new individuals by swapping a part of one parent with a part of the other. Figure 3.4 shows an example of crossover. A crossover follows the following steps.

- Two parent trees are selected randomly for crossover from a pool of candidates where the pool is selected through a selection process explained in Section 3.2.8.

- After selecting two parent trees, a sub-tree is randomly chosen on each parent tree. The two selected sub-trees are enclosed by dotted circles in Figure 3.4.

- The selected sub-trees are swapped with each other to make two offspring (children).

Mutation

Mutation operates on only one parent and alters the parent in a random way to produce child. A sub-tree is chosen randomly on parent tree and is replaced

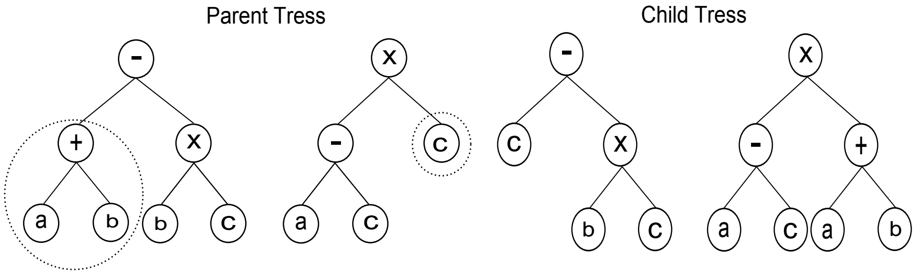


Figure 3.4: An example of crossover

by a randomly generated tree. This random tree is generated in the same way and subject to the same constraints (depth, size, etc.) as trees in the initial population.

Reproduction

Reproduction is a cloning process in which an individual is copied into the next generation without any change. Reproduction is seldom used in GP as survival strategy (explained in Section 3.2.10) may copy a parent without need of reproduction.

3.2.10 Survival Strategy

After producing a new generation of individuals, GP has to find which individuals (among children and parents) will make it to the next generation. This depends upon the survival strategy or the elitism parameter of GP. The elitism parameter can have three levels of elitism.

- **Replace**
  All the parents are replaced by their children even if they are worse than their parents. The elitist individual is not guaranteed to make it to the next generation in this option. Some of the good solutions may be destroyed in this case.
- **Keepbest**
  The best individuals from both parents and children are selected for the next generation. The drawback is that evolutionary process may loose diversity in this case and it might also make the learning process slow.
- **Halfelitism**
  The best half of individuals (according to fitness values) from both parents

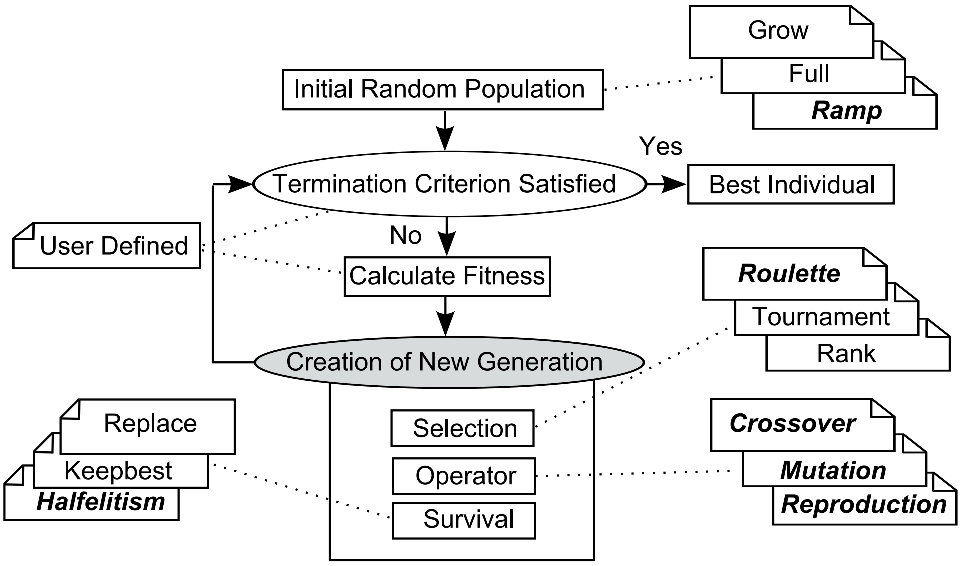


Figure 3.5: Full GP showing different parameters

and children make it to the next generation. In this case there is almost equal representation from both parents and children and this also makes sure the elitist individual remains in the population. This option of elitism has been used in this research.

3.2.11 Termination Criteria

This criteria decides when to terminate a GP run. Generally, a GP process is terminated if it reaches a predefined number of maximum generations or it finds a perfect solution which solves the problem completely. It can also be terminated if a certain threshold fitness is achieved (0 in case of ideal solution for standardized fitness).

Figure 3.5 shows full GP process showing all the stages of GP and also showing the different parameters used in this research during GP evolution. The parameters in bold have been used in this research.

3.3 Some Applications of Genetic Programming

There are a lot of relevant practical applications of GP and some of them are discussed below. Early work in GP was limited to some simple logic problems for demonstrating the method and these problems were used as benchmarking problems for comparing GP with other state of the art systems. The main reason for this limitation was the computational resources required for GP. The improvement in computational resources over the years has enabled us to apply GP for real world problems. The tree structure used for GP solutions was discussed earlier but there are many other structures used to represent solutions including graphs and lists. The choice of program structure depends upon the problem and it affects the execution order and use of genetic operators. These structures (trees, graphs, etc.) can represent a complex structure (e.g., circuit, controller, antenna, chemical reaction, etc.) and can be tuned for any particular application (e.g., control, data mining, financial, image processing, PR, etc.).

Antenna design which is a structural design problem can be solved by carefully creating geometry and size of antenna that satisfies design requirements using LISP structures. In this problem each individual will represent an antenna and the performance of this antenna will be tested by antenna simulator, used as fitness for that antenna. Recent works have proved that it can produce antennas which are human competitive [26]. It can also be applied to sequence problems where information is transferred in sequence. For example reading this line is a sequential process as text is transferred into a spatio-temporal form which our brain can process. The way to solve it in GP is to analyse certain patterns in sequence and compare different sequences for similarity [27].

An Impressive GP-supported image processing algorithm was presented by Daida, Begey, Ross and Vesecky [28]. Design of electrical circuits using GP was presented by Koza [25] where GP was successfully evolved for a large number of circuits with impressive results. Resistors, capacitors, inductors and functions for making parallel or series connections were used in the function pool. Many human competitive solutions have been synthesized using this system [29]. Some other popular examples are speech processing, communications in general, DNA pattern recognition, weather prediction, etc.

Above discussion shows that GP has shown its worth in a broad spectrum of real world problems with remarkable flexibility as a machine learning method. The interpret-ability of GP solution generates new insight into the solved problem. The designs produced by GP in some cases [29] matches human expert's performance both in time and quality of solution. Trying to achieve the same results with other methods will require high level of human input to adapt the method to meet design requirements.

3.4 Issues in GP

While GP literature often cites the important contributions made by GP and demonstrates that it is a powerful methodology, it does have some inherent problems. This section is about the problems in GP and different methods to tackle those.

3.4.1 Code Growth

The application of GP to a difficult problem can result in an increasingly big solution in size which may not be correlated with improvement in performance. This increase in size without corresponding increase in fitness is a big worry for GP practitioners. During the evolution some nodes are added in the solution which have no effect on the output and are known as *introns*. Introns not only increase the computational complexity by increasing the size of solution but also result in inefficient solutions. Many strategies have been developed in literature to limit the code growth, each having its own advantages and disadvantages. Some common methods for controlling code growth are described below.

Parsimony Pressure

In this strategy the solution with bigger size are penalised by adding a component in fitness function which depends on size. Assuming the standard fitness function (where lower fitness is better) the fitness with parsimony pressure can be calculated as given below

$$f_{new} = f_{old} + kn \tag{3.4}$$

where f_{new} is the new fitness after adding penalty for size, f_{old} is the old fitness calculated through original fitness function (pre-parsimony pressure), k is the weight given to parsimony and n is the number of nodes in the solution. The value of k depends upon requirement and a small value could be used if computation is not a problem. On the other hand, a large value of k would make sure that size of the final solution is small and evolution process is fast.

Maximum Depth of Solution

This is the most widely used method for restricting code growth in GP solution. Two parameters are used for controlling the depth of a solution, *strict maximum*

depth and*dynamic maximum depth*. The strict maximum depth of GP trees (solution) is defined at the start of evolution and any individual (off-spring) created as a result of genetic operation, violating the strict maximum depth condition is rejected. Dynamic maximum depth is not fixed and can change during evolution but can never exceed strict maximum depth. For each new produced individual there could be three possible scenarios:
1) If the new individual does not exceed the dynamic maximum depth, it can be used as it does not violate the condition.
2) If the depth of individual is greater than dynamic maximum depth but less than strict maximum depth, the fitness of individual is calculated. If the fitness of the individual is the best among all the individuals produced so far, it is allowed to stay and dynamic maximum depth is increased to the depth of this individual else individual is rejected and one of its parents takes its place.
3) The depth of the individual is greater than strict maximum depth, individual is rejected and one of the parents takes its place.

Dynamic maximum depth is initially set with a lower value and a value of 6 has been used in this study. The value of strict maximum depth used in this study is 28.

3.5 Super Feature Generation Using GP

GP has been used in this research for generating features with the intention of solving classification problems. Many classification techniques have been used in the past but the technique introduced here uses the concept of *super feature*, which makes classification problem easy for any independent classifier. This technique can also be considered as dimension reduction, in which the given problem dimensions always reduce to a single dimension (super feature), such that the required classes are independent in super feature. GP does not solve classification problems directly but generates effective super feature which makes the classification an easy task.

In this section we will discuss different types of classifiers and the use of GP for classification tasks. A simple example will be used to demonstrate the effectiveness of GP for classification.

3.5.1 Classification Background

Classification is a problem in which a test sample is assigned to a particular class, on the basis of training samples for which class membership is known. Classification plays a vital role in many fields such as PR, fault detection, image processing, biomedical data processing, data mining, etc. This subject has been of major interest in recent literature and different approaches have been presented for classification problems.

Classifier Types

Any algorithm which maps data into corresponding categories is called a *classifier*. There are two types of classifier algorithms, *supervised and unsupervised.* In *supervised* algorithm a training dataset with known outputs is available and algorithm's task is to find patterns in this training data. Once the training is complete, algorithm is tested with test data to quantify its learning ability. In *unsupervised* algorithms there is no known output and algorithm is asked to group similar items based on some inherent similarities. It is also known as *clustering*, and in some cases (k-means clustering) the number of clusters (groups/classes) are known and algorithm is asked to make specific number of clusters. It is apparent that unsupervised algorithms have much less to start with and is more complex than its counterpart. That is the reason most of the past methods have considered supervised class of problems and the same has been used in this work.

Two types of approaches are used while designing a classifier, *feature space division approach* and *function estimation approach.* In feature (input features) space division approach, each feature represents one dimension and this feature space is divided into different class labelled sections. The test sample is classified according to which labelled section it falls in. A linear classifier can create a hyper plane in feature space directed by linear classifier for classification. Although this approach works only if classes are linearly separable but it provides fast classification and is particularly useful in situations where speed is a major concern. A non linear hyper plane could be drawn for problems not linearly separable using SVMs. SVMs are relatively complex but have gained much importance in recent years. A distance based classification for feature space division is achieved using KNN. This classifier plots the training (reference) samples in feature space and labels their respective classes. The distance of test sample is calculated from all the reference samples. The class of k nearest neighbours is found and the class having maximum number of neighbours is considered to be the class of this test sample. KNN belongs to machine learning algorithms and is one of the simplest machine learning algorithms. Another simple approach is decision tree classifier (rule based classifier), which makes decision using IF-ELSE logic on feature values. It works well for simple classification problems but making simple decision rules for complex problems is not always possible.

In function estimation approach the aim is to find a function of input variables which can separate classes. ANNs have been used in this field widely. ANNs consist of input layer of neurons connecting input features to hidden layer of neurons which is then connected to output neurons. ANN is trained to give specific outputs for specific inputs by changing weights of different neurons. Although ANN is capable of classification where linear separation of classes is not possible, its computational requirement is always a major concern.

Feature Selection

Features are the most important part of any classification process. Ideally a feature set is required which have enough information to classify the data easily but it does not necessarily mean that a large feature set is required. A large feature set having irrelevant features would not only add to complexity but would result in decrease in performance. Large feature sets require preprocessing of the data removing redundant and less important features.

Feature selection can be divided into two main approaches *filter approach* and *wrapper approach.* In filter approach feature selection can be seen as preprocessing step independent of classification algorithm used later while in wrapper approach it is taken as part of classification algorithm. Filter approach is much more simpler and computationally efficient. A subset of original features is chosen based on the predictive significance of features. An ideal subset should contain features, correlated to class membership but uncorrelated to each other. Normally statistical tests are carried to check the above mentioned criteria. Wrapper methods on the other hand are more complex, have more computational requirement but give better results compared to its counterpart.

Feature generation is also performed at times instead of feature selection. It transforms a set of features from original d_o dimensions to new set of dimensions d_n such that $d_n < d_o$, which results in smaller, richer set of features. The popular methods for feature generation are PCA, ICA, etc. Fortunately, GP has an inherent feature selection process and implicit feature selection is performed during evolution.

3.5.2 Classification Using GP

Advantages

GP offers many advantages compared to other classifiers when used for classification. First of all the input data can be given to GP in raw form and there is no need to perform the preprocessing in most of the cases. This is not the case for most of the other classifiers, e.g., ANN frequently requires preprocessing of data. Flexibility is another advantage offered by GP. It offers a solution which could be a combination of conditional, non-linear and many other functions, by including all these functions in the function pool.

Interpretation of the result is another factor that makes GP important. The final solution consists of combination of input terminals with available functions which is available for interpretation. Some of the other classifiers are considered to be black box as they focus on improvement in performance but are difficult to interpret, e.g., ANN, KNN and SVM have been traditionally considered as block box. GP belongs to class of white box methods which produce interpretable solutions. Not only these solutions are more interpretable but could be made more comprehensive by controlling the size of final solution (at the cost of reduction in performance). For example by specifying the maximum size of tree or by penalising the big trees through fitness function.

The most important advantage offered by GP is its inherent capability of feature selection. GP automatically selects the useful features from terminals and ignores less important features. In most of the other classifiers this process must be done before training as a part of preprocessing.

History As a classifier

GP has been used in the past for classification tasks. A full survey is available in [30]. Most of the times GP has been used for binary classification problems. In [31], a weighted error in classification accuracy for two classes was used as fitness function. A weight was associated with each instance of training data and the instances harder to classify were given a higher weight during evolution. In [32], a solution was proposed to deal with classification of imbalanced data. A data is imbalanced when one class heavily dominates the data and normally the minority class is the class in which we are interested. An evolutionary algorithm has a tendency in an imbalanced dataset to overlook the minority class data and pay more attention to the majority class data. In this paper they tried to solve the problem by generating classification rules only for the class we are interested in, and the other one is taken as default class.

GP was used for multi-class classification in [33]. This paper divided the classification process into two stages. At the first stage a Bayesian network was trained using evolutionary programming, representing relationships between different attributes. This Bayesian network defined the rules to be used by GP at the second stage. In [34], a penalty term was introduced in the fitness function for controlling size of GP trees. A method for solving n class classification problem using GP was proposed by Kishore, Patnaik, Mani and Agrawal [35], and Muni, Pal and Das [36]. They divided the n-class classification into n 2-class problems to realize multi-class classification with GP. This method inherited the simplicity of 2-class GP classifiers. However, as the whole process involves n number of 2-class classifiers, the total resources required are n times the one needed for each 2-class classifier. As the number of classes increase, the complexity of the classifier increases as well. This problem makes this classifier suitable only for small number of classes.

Zhang and Nandi [37] and Guo, Jack, and Nandi [38] used GP for feature selection as well as feature generation for fault classification in rotating machines. They demonstrated through experiments that conventional feature selection process is not needed for GP because of its built in feature selection mechanism. Guo and Nandi [39] used a modified Fisher criterion for optimising GP in a way that the ratio of within-class scatter over between class scatter is minimised. They tested their method for breast cancer detection. Zhang, Jack and Nandi [40] used KNN in combination with GP for multi-class classification. The rotating machine data was used for testing GP solution and the final classifier used was KNN. Another method to use GP for multi-class classification was developed by Zhang, Ciesielski and Andreae [41]. They used multiple thresholds for different classes and used GP to generate outputs for different classes to fit in spaces between these thresholds. When applying this method, as the thresholds are problem-dependent, users need to set the thresholds manually for every problem. Before using GP to optimize the output used for classification, if one wants to achieve better performance from the GP classifier, a large number of tests are needed to estimate the optimum order of the thresholds and the spacing between different thresholds. This is a time consuming process and it is still difficult to achieve the optimum performance.

Super-Feature Generation

There are many methods available in literature for dimension reduction either by transforming original features into new feature space or by removing less useful features. All of these methods are helpful in reducing the computational cost by reducing the number of features but it is rarely possible to reduce dimensions to a single feature due to the nature of the process. GP has the ability to combine input features in a diverse way and give result in the form of single feature. A *super-feature* is a GP generated feature (single dimension) which combines the input features in linear or non-linear fashion (depending on the function pool) such that it has all the information required for classification. After getting the *super-feature* from GP training process, the classification can be achieved using simple decision making process based on the value of *super-feature*.

Fitness function plays an important role in quantifying the ability of *super-feature* for classification. Since fitness function quantifies the usefulness of *super-feature*, it is necessary to explain the term "useful feature" in terms of classification. Before choosing fitness function, the first question one should ask is what kind of information we want a *super-feature* to produce? In terms of classification an ideal feature should return a value based on class membership and same is expected from *super-feature*. In addition to class membership, a *super-feature* should be consistent in assigning a class label to different inputs of the same class (known as robustness). The variance of *super-feature* is an important parameter to check the robustness of *super-feature*. A small output variance for inputs of the same class is the quality of a good feature. Ideally we want the output for the inputs of same class to be clustered close to each other so that variance is minimum. Also we want the outputs of two different classes to be as much away from each other as possible. In short, a good feature should have high inter-class variance and low intra-class variance.

Figure 3.6 shows a classification example using *super-feature* where two classes are perfectly separate (an ideal scenario). It highlights different parameters that will be used in fitness evaluation. In the Figure m_1, m_2 are the means of two classes, M is the difference between the mean of two classes and v_1, v_2 are the variance of two classes. According to our previous statement we want to maximize M (inter-class variance) and minimize v_1 and v_2 (intra-class variances). This could be achieved using *Fisher criterion* as a fitness function which tries to maximize the difference between means of two classes while minimizing their variances. Fisher criterion is shown in the form of equation below

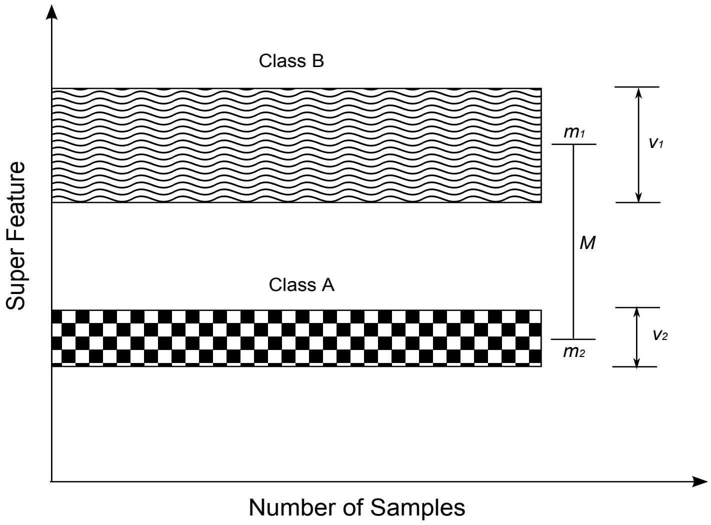


Figure 3.6: An example of a super feature

$$fitness = \frac{|m_1 - m_2| + p}{\sqrt{(v_1 + v_2)} + p} \tag{3.5}$$

where p is a constant used to guard against division by zero and multiplication by zero. In order to standardize the fitness function where lower fitness is considered better, an inverse of the above equation was used throughout this research as fitness function and is shown below

$$fitness = \frac{(\sqrt{v_1 + v_2}) + p}{|m_1 - m_2| + p} \tag{3.6}$$

From now onwards a lower fitness would mean a fitter individual (as in standard fitness). The above fitness function is useful if the number of classes are two but it has limitations if the classes are more than two. Loog and Haeb-Umbach [42] pointed out that Fisher criterion favours the outlier classes in multi-class classification. In this study Fisher criterion has been used only for binary classification and details about how to use it for multi-class classification are available in [38].

A Binary Classification Problem

A simple binary classification example is presented in this section to elaborate the effectiveness of *super-feature.* Modulation classification of QPSK and 16QAM has been used as an example. Noisy data with an SNR of 5 dB is received which uses the above mentioned two modulations. Some features need to be collected

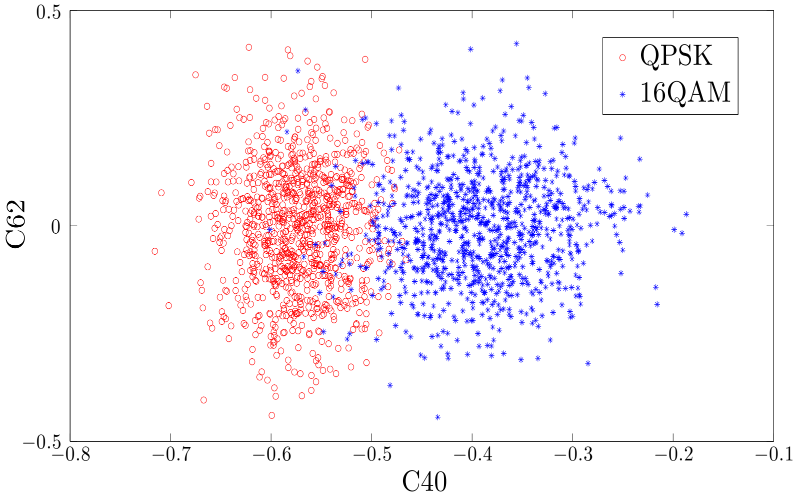


Figure 3.7: Cumulant values for QPSK and 16QAM samples using 1024 number of samples and at an SNR of 5 dB

from this data which will be supplied to GP for making a *super-feature* from these input features. Cumulants which will be discussed in the next chapter are collected as features from original data samples. Two cumulants C_{40} and C_{42} are used for this purpose.

Figure 3.7 shows the two cumulant features plotted against each other clearly showing the overlap between two modulations. These feature were fed as input to GP and GP was trained to produce a solution separating the two classes. The number of generations used for training were 25 with a population size of 25. The role of *super-feature* is to reduce the dimensions to a single dimension (by using combinations of original dimensions) such that the two classes are separate from each other. Figure 3.8 shows the solution offered by *super-feature* for this problem where two classes are completely apart from each other and the dimensionality is also reduced from two to one. This example demonstrates the ability of GP to solve simple binary classification problems. In upcoming chapters GP will be applied to more difficult binary classification problems and the extension of GP for multi-class classification will also be presented.

After the generation of *super-feature* the classification accuracy of this feature has to be tested by a classifier. In the above example two classes were completely separate from each other in super feature space so classification can be performed by setting a simple threshold value between the two classes. This is not the case in real world classification problems where the overlap between different classes is much greater compared to the overlap in Figure 3.7. GP tries to minimise this

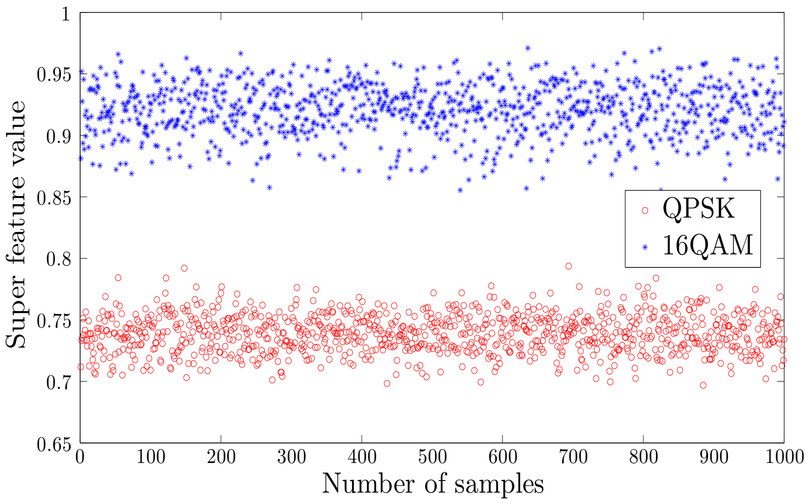


Figure 3.8: Super feature values for QPSK and 16QAM samples using 1024 number of samples and at an SNR of 5 dB

overlap but even after generating *super-feature* there is still some overlap between the classes and it is not preferred to calculate classification accuracy using simple threshold decision. Any machine learning classifier can be used for this purpose and this study uses KNN, ANN and SVM as a final classifier for calculating the accuracy of *super-feature*. The use of particular classifier depends on the problem but KNN is the preferred choice because of its simplicity and low computational complexity compared to the other two.

3.6 Summary

This chapter presented an overview of GP. GP is highly adaptable and has the potential to be used in a wide variety of problem domains. GP belongs to class of evolutionary algorithms in which computer programs (solutions) are evolved. The fact that it evolves actual solutions of the problem makes it different to all other machine learning methods as most other methods train a network which is later used as a classifier. It also has the built in ability of feature selection so feature selection as a pre-processing step is not required. Different parameters of GP are user-defined and can be adjusted to target the problem. For example the function pool helps the evolution process as functions can be added or removed according to the nature of the problem. Fitness function is another important parameter which can be designed to guide the training process according to the requirement of problem.

GP has been increasingly used in the recent literature for classification problems and the introduction of *super-feature* makes it more suitable for real world classification problems. The example provided in this chapter shows the potential of GP for classification problems. The algorithm is not problem dependent and can be easily applied to any classification problem with slight modifications. The primary aim of the remainder of this thesis is to modify the existing structure of GP to make it more suitable for classification tasks. As a secondary aim, attempts are made to provide solutions for improvement of classification accuracy of real world problems.

Chapter 4

The Experimental Datasets and Preliminaries

This chapter explains the case study problems and the datasets used in this thesis. Two types of PR problems have been used as case studies in this thesis. Although both of these problems fall under the category of PR, they are from two completely different fields. The first problem is related to the detection of diabetes, a medical diagnosis problem. The detection is performed using a set of features obtained from diabetes patients. A well known publicly available dataset is used for this purpose. The second PR problem is related to the communication systems. Automatic digital modulation classification without prior knowledge about the modulation has been used as a case study problem.

The chapter is organised as follows: a short history of the diabetes is presented initially, explaining different types of the disease, its causes and effects. The advantage of doing automatic diabetes detection is also discussed. Brief history of the Pima Indians and the Pima Indian diabetes dataset used in this study is presented next. A short review of communication systems, the advantages of modulation and comparison between digital and analog modulations is presented. Automatic modulation classification and its applications are discussed. Finally, some statistical and probabilistic definitions, and measures are presented which will be used as features for modulation classification problem.

4.1 The Diabetes

Diabetes mellitus, often simply called diabetes, *is a metabolic disorder in which a person has high blood sugar level either due to lack of insulin or due to ineffectiveness of the insulin.* The word *diabetes* is a Greek word which means *to pass* *through*, and mellitus is a Latin word which means *honey-sweet*. The first part of the name is for the reason that diabetes causes frequent urination and in fact it was also known as *the pissing evil* in 17th century. The second part of the name is because this disease results in an increase in sugar in blood.

Most of the food that we eat is turned into glucose or sugar. Glucose is passed to the bloodstream after digestion, where it is used by the cells for growth and energy. Insulin is a hormone produced by pancreas for moving glucose from blood to cells. In diabetes, either the pancreas produces little insulin or the cells do not use the produced insulin properly. This results in an increase of glucose in the blood, which passes out of the body through urine and ultimately results in loss of fuel (glucose) for the body, although, it is present in large amount in the blood. In simple words diabetes can be described as; *a metabolic condition in which the blood sugar level is higher than normal*. Symptoms of diabetes include; increase in hunger, increase in urination, increase in thirst, weight loss, weakness, tiredness, vomiting, etc. Diabetes is one of the most common hormonal problems and leads to many other diseases including heart disease, high blood pressure, nerve damage, numbness in hands or feet, diabetic retinopathy and diabetic nephropathy.

There are two main types of diabetes, type 1 and type 2. In type 1 the beta cells in pancreas, responsible for producing insulin are destroyed and as a result pancreas produces little or no insulin. Type 1 mostly occurs in children or young adults but can affect at any age. People suffering from this type have to take insulin injections regularly to stay alive. Type 2 is the most common type of diabetes covering at least 90% of all the diabetes cases. In this type body becomes resistant to insulin and does not effectively use the insulin being produced. This type mostly occurs in the elders after the age of forty years but can be found in youth as well and can be treated by following a healthy diet plan, doing exercise regularly and/or taking tablets. In some extreme cases insulin injections may also be required. However, diabetes still contributes to heart disease even if it is under control.

In recent years diabetes has become an alarming problem across the globe. There are more than 260 million diabetes affected people in this world. Each year 65,000 new cases of diabetes are reported. Diabetes has been increasing at a rapid rate and if it continues to increase at the current rate, there would be demand of a large number of physicians in near future. In order to cope with this problem, the use of the classifier system in medical diagnosis has increased in recent times. Given some details about the patient, a classifier system can automatically figure out if a patient has diabetes, without the need of a physician. If the decisions made by physicians on previous patients having similar conditions are saved in a list along with patient conditions, a classifier system could be designed which makes use of the conditions and classifies that list according to the decisions made by physicians. No doubt, data taken from the patient and expert's opinion about the data are the most important in diagnosis but a classifier system can also help physicians a great deal.

4.1.1 The Pima Indian Diabetes Dataset

Pima Indians are the American Indians living close to southern Arizona for about 2,000 years and their ancestors came to America about 30,000 years ago. They have volunteered themselves for research studies since last 50 years and according to doctors this will help to understand the root cause and diagnose different diseases [43]. According to Dr peter Bennett, Chief of the Phoenix Epidemiology and Clinical Research Branch of the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), "*The Pima Indians are giving a great gift to the world by continuing to volunteer for research studies. Their generosity contributes to better health for all people, and we are all in their debt*". The research on Pima Indians started in 1963, after finding a high rate of diabetes among them.

Pima Indians have lived near Gila River for generations and they often marry other Pimas, and this can help scientist to trace the root causes of different diseases through several generation of families. Studying the progress of disease through generations and generations helps to find out how different diseases affect people and gives an opportunity to develop a cure for the disease. Most of the today's findings about diabetes, the reasons and the remedies have come out of this research.

The dataset used in this study was prepared by NIDDK in 1990 and is available publicly from UCI Repository of Machine Learning Databases [44]. All the patients who took part in this study were females of Pima Indian heritage and at least 21 years old. There were total 768 cases, out of which 500 (65.1%) cases had no diabetes (class 0) and 268 cases (34.9%) had diabetes (class 1). Each of these cases had eight attributes and details of these attributes are given in Table 4.1.

An important point to mention is that there are a lot of missing values in this dataset. Although, the donor of the dataset did not mention any missing values but there are some values in the dataset that cannot be true, e.g., there are a lot of zeros for triceps thickness (feature 4) and hour serum insulin (feature 5) which cannot be true. The number of zeros for feature 1-8 are 111, 5, 35, 227, 374, 11, 0 and 0 respectively. The presence of zeros can be true only for feature 1 which is number of times a woman was pregnant but all other zeros cannot be true. If we remove the rows with zero values for feature 2-6, only 392 samples out of 768 would be left which means almost 49% of the data would be removed if missing values were removed. Since most of other researchers have used the data in its original form (considering zeros as true), in this study also the original data available at [44] is used without any change to have a fair comparison with other methods present in literature.

Normalisation

In any data analysis problem, data must be prepared before it can be used in an algorithm. The term *preparing the data* means transforming the data such that the job of the algorithm becomes easier. The feature values of a practical dataset may lie within different dynamic ranges and the features with larger values (where large values do not necessarily represent more significant values) may have greater influence on the final outcome of an algorithm. Normalisation is used in such a situation where original features values are transformed such that the new feature values lie within similar ranges. A simple technique to normalise data involves mean and variance of features. The process of normalisation can be demonstrated using the diabetes data. Since the means of different diabetes features are quite far from each other, the data is normalized according to the following equation

$$X_{ik,new} = \frac{(X_{ik,old} - \bar{X}_k)}{\sigma_{X_k}} \tag{4.1}$$

Here $X_{ik,new}$ is the new feature value after normalization, $X_{ik,old}$ is the original feature value, $\bar{X}_k$ is the mean of the feature X_k and σ_{X_k} is the standard deviation of the feature X_k. The definitions of mean and standard deviation are given below.

$$\bar{X}_k = \frac{1}{N}\sum_{i=1}^{N} X_{ik} \tag{4.2}$$

$$\sigma_{X_k} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_{ik} - \bar{X}_k)^2} \tag{4.3}$$

Table 4.1: Brief analysis of the diabetes dataset

No.	Attribute	Mean	Standard Deviation	Min/Max
1	Number of times pregnant	3.8	3.4	0/17
2	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	120.9	32.0	0/199
3	Diastolic blood pressure (mm Hg)	69.1	19.4	0/122
4	Triceps skin fold thickness	20.5	16.0	0/99
5	Hour serum insulin (mu U/ml)	79.8	115.2	0/846
6	Body mass Index	32.0	7.9	0/67.1
7	Diabetes pedigree function	0.5	0.3	0.078/2.42
8	Age (years)	33.2	11.8	21/81

4.2 Automatic Modulation Classification

4.2.1 Communication System

Communication has become an integral part of everybody's life nowadays. We often use communication systems without realising the process behind it, e.g., telephones, computers, television, internet, etc., that we use in our daily life are all examples of different communication systems. Although, the study of communication system involves many areas of science, broadly, the study of any communication system can be divided into two main areas under the following questions:

1. How any communication system works?
2. How does it perform under the influence of noise?

Knowledge of signal processing techniques is required to understand and study the first area. Study of the second area involves basic understanding of the probability theory and stochastic processes. It is necessary to have a good understanding of both parts in order to understand a communication system completely [45]. A complete communication system showing different parts is shown in Figure 4.1. Brief description of each part is given below.

The *input signal* such as voice, a television signal, a text message, etc., is modulated (will be explained shortly) by the modulator and sent by the transmitter. The message could be digital or analog. Digital message have a finite number of symbol levels while an anlalog message can have infinite symbol levels. Digital signals have taken over from analog signals because of several advantages they

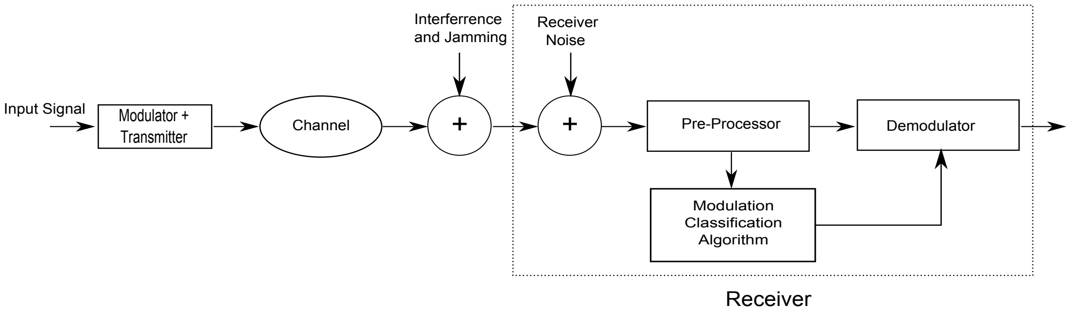


Figure 4.1: An example of a communication system.

offer, especially the larger noise limits. Generally, analog signals are converted to digital signals for efficient communication and once received, they are converted back to their original form.

The *channel* is a medium to transfer the input signal from sender to receiver, e.g., wire, optic fibre, coaxial cable, wave guide, wireless link, etc. There will be noise and interference added to the original transmitted signal, as the signal passes through the channel. There are two types of noise added to the signal during transmission, internal and external. The internal noise is due to the thermal motion, diffusion recombination of charge carriers, faulty contact switches, etc. External noise could be due to lightening, storm, interference from other nearby signals, etc. Although noise can be reduced with proper care, it is difficult to eliminate completely. It is one of the major factors that put a limit on the maximum achievable communication rate. Once the modulation type of received signal is found, demodulator is used to reconstruct the original signal.

The *receiver* receives the modulated signal and demodulates it. If the modulation type of incoming signal is not known, a modulation classification algorithm is used which tries to find the modulation type of received signal without having any information about the modulation used at the sender.

4.2.2 Modulation

The original signal or message (also called the baseband signal) is not always suitable for transmission over the channel for various reasons. The baseband signals is modified in order to facilitate the process of transmission. A carrier signal which is a sinusoid of high frequency is used for this purpose. Modulation can be defined as: *the process of modifying one or more properties of a carrier signal by a baseband signal is known as modulation.* The properties which are modified by the baseband signal could be amplitude, frequency, phase, etc., and the corresponding modulations are called amplitude modulation (AM), frequency modulation (FM), and phase modulation (PM). The modulated signal, once received at the receiver must be demodulated before it can be used. There are various advantages offered by modulation and some of them can be described as:

1. For an efficient transmission, the size of the radiating antenna should be one tenth of the wavelength of the radiated signal. The frequency of most of the baseband signals is quite low resulting in impracticable large dimensions for the radiating antenna. A high frequency carrier makes sure that size of antenna is within practical limits.
2. Baseband signals, transmitted at the same time, with similar frequencies would interfere with each other. Using carrier signals with frequencies far apart from each other for different baseband signals, will translate each baseband signal to a different frequency range, thus avoiding interference. This method of translating signals to different frequency bands is called frequency division multiplexing (FDM).

4.2.3 Digital vs Analog Modulations

Today digital modulations are taking over from analog modulations rapidly, e.g., satellite televisions, digital radio, WiMAX, CDMA, etc., are all examples of digital communication systems. The popularity of internet is another evidence which also uses digital signals. Moreover, in mobile communication the transfer from 2G networks to 3G [46–48] and 4G [49, 50] is an example of popularity and effectiveness of digitally modulated signals. Digital signals offer many advantages over analog signals, e.g., they are more fast, power efficient, provide higher data rate, more immune to noise, provide better encryption, etc. Since there has been more focus on digital signals, only digital modulations are considered in this thesis.

Some common digital modulations are amplitude shift keying (ASK), frequency shift Keying (FSK) and phase shift keying (PSK). A more popular digital modulation scheme is quadrature amplitude modulation (QAM) which varies both the amplitude and phase of a carrier signal. Essentially, it is a combination of ASK and PSK, and both ASK and PSK can be regarded as special cases of QAM. A variety of QAM signals are available depending on the number of bits per sample used and some of the most common are 8QAM, 16QAM, 32QAM, 64QAM, 128QAM, etc. In this study only four digital modulations are used as a case study problem, including BPSK, QPSK, 16QAM and 64QAM. The reason for choosing these modulations is that there has been a lot of focus on these modulations recently because of the advantages they offer. The similar constellation diagrams of these modulations make the classification difficult. These four modulations are included in quite a few IEEE standards [51], e.g., 802.11a, 802.11b, 802.11g, 802.11n, etc.

4.2.4 Automatic Modulation Classification and its Applications

Automatic modulation classification(AMC) has received much attention in the recent past and is a rapidly growing area. A modulated signal once received at the receiver needs to be demodulated in order to get the actual message. The process of demodulating a signal without having any information about the modulation used at the sender is called AMC. In the past human operators were used for finding modulations, after analysing different parameters of received signals. Typical parameters used were instantaneous amplitude, instantaneous phase, signal spectrum, etc. In another scheme a series of demodulators was used to find the actual modulation type. This scheme also required a human operator who used to decide about the modulation, based on the outputs of the demodulators.

Each modulation has its own unique characteristics which can be considered as a pattern. Finding the underlying modulations in a received signal can be regarded as a PR problem. In this study it has been treated as a PR problem and machine learning methods (GP, ANN, SVM) have been used for finding the patterns. The goal of the present study is to just classify different underlying modulations, without actually demodulating them.

Applications

Any communication intelligence (COMINT) system has three main parts: receiver front end, modulation detector and output [52]. The first part (receiver front end) could be a heterodyne receiver, homodyne receiver, channelised receiver, etc. The second part is modulation detector which tells us about the modulation of the received signal. The last part consists of demodulators, deciphers, information extractors, etc. Since the focus of this research is modulation classification, we are concerned about the modulation detector only and discussion about other parts is not presented. The performance of the output stage depends on the information passed by modulation detector so it plays an important role in COMINT. COMINT has a wide range of military and civilian applications and some of them are given below:

1. Civilian Applications
    - No Human operator required
    - Spectrum management
    - Interference identification
    - Transmission control and monitoring
    - Software defined radio
    - Cognitive radio
2. Military Applications
    - Electronic Warfare
    - Target acquisition
    - Surveillance and threat analysis
    - Jamming
    - Homing

In the past COMINT relied on human operators who used to decide the modulation type of an incoming signal after observing some properties of signal. Human operators are not required any more since AMC system can be used instead of humans. The frequency spectrum in any country is divided between different licensed users. COMINT can be used to find out if any user is within or outside its assigned spectrum (spectrum management). A user which goes outside its allotted spectrum will cause interference to other users and this interference can be identified using COMINT. The spectrum management and interference identification ability of COMINT can be used to control and monitor the transmission of signals.

Software defined radio (SDR) is an emerging application of AMC. Recently, there has been a lot of focus on digitising RF signals to replace analogue components with Digital Signal Processors. Minimization or replacement of hardware in communication systems has resulted in the birth of SDR [53–58]. SDR is a flexible communication device that can handle multiple modulation schemes. With the recent developments in SDR, AMC has gained more attention than ever. AMC can be used at the front end of SDR to classify the modulation type of an incoming signal. In this way a single SDR can handle multiple modulations.

Cognitive radio is also an emerging technology for dynamic spectrum access [59–61]. A cognitive radio is a SDR that intelligently senses its environment, tracks changes, plans the most appropriate response and reacts accordingly, so it can be called a refined SDR. Main idea in cognitive radio is to allow a secondary user to reuse or share a radio spectrum originally allocated to primary user if it is under utilized by primary user. In military and public safety applications, there is no information available about the modulation type of signals. Hence, in this case cognitive radio needs to have an accurate information about the modulation type of signal used by primary user to demodulate it. For this reason, the performance of cognitive radio can be greatly enhanced by using a reliable modulation classification scheme for such applications.

COMINT can be used in electronic warfare to find out the signal used by the enemy in a particular spectrum and to take appropriate action to negate it (target acquisition). Based on the energy of the signals used by the enemy, one can identify the threat attached to it. COMINT can provide detailed information about the enemy signals which can be used for jamming. Another use of COMINT is intercepting radar signals and learning their locations (homing).

4.2.5 Mathematical Preliminaries

Let S be a sample space consisting of all the possible outcomes of a random experiment. Let A and B be two mutually exclusive (disjoint) events and are subsets of S. If N trials of this random experiment are conducted and the number of occurrences of event A is $N(A)$, the probability of the event A denoted as $P(A)$ (also sometimes called the relative frequency of A) can be written as,

$$P(A) = \lim_{N\to\infty} \frac{N(A)}{N}$$

such that

- $0 \leq P(A) \leq 1$
- $P(S) = 1$

Since A and B are mutually exclusive,

$$\begin{aligned} P(A \cup B) &= \lim_{N \to \infty} \frac{N(A) + N(B)}{N} \\ &= P(A) + P(B) \quad if \quad AB = \phi \end{aligned}$$

If A and B are not disjoint,

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

From mathematical point of view it is desirable to have a real number as outcome of an experiment. Since the outcome of a random experiment is not always a real number, a random variable is used. A random variable is a function that associates a real number with each outcome of any random experiment. Let us consider a random variable X and $x_1, x_2, ..., x_n$ are the values it takes. The cumulative distribution function (CDF) for this random variable can be written as,

$$F_X(x) = P(X \leq x)$$

such that

- $F_X(x) \geq 0,$
- $F_X(\infty) = 1,$
- $F_X(-\infty) = 0,$
- $F_X(x_1) \leq F_X(x_2) \quad \text{for} \quad x_1 \leq x_2$

The derivative of CDF is called the probability density function (PDF) and can be written as,

$$f_X(x) = \frac{\mathrm{d}F_X}{\mathrm{d}x}$$

such that

- $0 \leq f_X(x)$
- $\int_{-\infty}^{\infty} f_X(x)\,\mathrm{d}x = 1$
- $F_X(x) = \int_{-\infty}^{x} f_X(\xi)\,\mathrm{d}\xi$
- $P(x_1 \leq x_2) = \int_{x_1}^{x_2} f_X(x)\,\mathrm{d}x$

Expectation, Moments and Cumulants

Expectation of any random variable is the mean value or the average value of the random variable and can be written as [62],

$$E(X) = \int_{-\infty}^{\infty} x f_X(x)\,\mathrm{d}x \tag{4.4}$$

Let $g(.)$ be a real function of X, the expected value of this function is given as,

$$E(g(X)) = \int_{-\infty}^{\infty} g(X) f_X(x)\,\mathrm{d}x \tag{4.5}$$

Moments are a set of expectations with two main types, moments about the origin and moments about the mean. If the function $g(X) = X^n$ where $n = 0, 1, 2, ...,$ the n^{th} moment about the origin can be written as,

$$m_n = E(X^n) = \int_{-\infty}^{\infty} X^n f_X(x)\,\mathrm{d}x \tag{4.6}$$

Here $m_0 = 1$ is the area of $f_X(x)$, and $m_1 = \bar{X}$ is the mean or expected value of X. The moments about the mean are also known as central moments. If the function $g(X) = (X - \bar{X})^n$, where $n = 0, 1, 2, ...,$ the n^{th} central moment can be written as,

$$\mu_n = E[(X - \bar{X})^n] = \int_{-\infty}^{\infty} (X - \bar{X})^n f_X(x)\,\mathrm{d}x \tag{4.7}$$

Here $\mu_0 = 1$ and $\mu_1 = 0$. The second order central moment μ_2 is called variance and is represented as σ_X^2

$$\sigma_X^2 = E[(X - \bar{X})^2] = \int_{-\infty}^{\infty} (x - \bar{X})^2 f_X(x)\,\mathrm{d}x \tag{4.8}$$

The square root of variance is called the standard deviation (σ_X) and is a measure of the dispersion in the density function $f_X(x)$ about the mean. The third central moment μ_3 is a measure of the asymmetry of the density function about the mean ($\bar{X}$) and is also known as skewness. For a symmetrical density function about mean, the skewness is zero.

Moments are a set of constants of a distribution, measuring its properties. There is another set of features called cumulants having properties more useful than moments. They are related to moments and provide an alternative to moments. Cumulants can be calculated from the moment generating function defined below,

$$M(\xi) = E(e^{\xi x}) = \int_{-\infty}^{\infty} exp(\xi x) f_X(x)\, \mathrm{d}x \tag{4.9}$$

and

$$C(\xi) = \log M(\xi) = \sum_r \frac{C_r \xi^r}{r!} \tag{4.10}$$

Since $\mu_1 = 0$, this implies $C_1 = 0$. The relationship between first few moments and cumulants for a zero mean variable x is given below,

$$\begin{aligned} C_1 &= \mu_1 \\ C_2 &= \mu_2 \\ C_3 &= \mu_3 \\ C_4 &= \mu_4 - 3\mu_2^2 \\ C_5 &= \mu_5 - 10\mu_3\mu_2 \\ C_6 &= \mu_6 - 15\mu_4\mu_2 - 10\mu_3^2 + 30\mu_2^3 \end{aligned} \tag{4.11}$$

Here C_4 is called the kurtosis and is a measure of length of the tail of $f_X(x)$. A normalised version of kurtosis is more often used, given as,

$$\hat{C}_4 = \frac{\mu_4}{\mu_2^2} - 3 \tag{4.12}$$

If the variance is unity the above equation reduces to,

$$\hat{C}_4 = \mu_4 - 3 \tag{4.13}$$

which is simply the fourth order central moment with a bias. The above formulae can be extended to the multivariate case. For example, C_{31} can be calculated from C_4 (equation (4.11)) by following the procedure given below,

$$C(r^4) = \mu(r^4) - 3(\mu(r^2))^2$$

If it is treated as a function of r, then operating it by $s\frac{\partial}{\partial r}$ gives,

$$4C(r^3 s) = 4\mu(r^3 s) - 12\mu(r^2)\mu(rs)$$

Dividing by 4 and replacing r and s by first and second variate gives,

$$C_{31} = \mu_{31} - 3\mu_{20}\mu_{11}$$

Following the same procedure other variates can also be calculated (for details please refer to [63]). The cumulants used in this study for modulation classification are given below,

$$\begin{aligned}
C_{40} &= \mu_{40} - 3\mu_{20}^2\\
C_{41} &= \mu_{41} - 3\mu_{20}\mu_{21}\\
C_{42} &= \mu_{42} - |\mu_{20}|^2 - 2\mu_{21}^2\\
C_{60} &= \mu_{60} - 15\mu_{20}\mu_{40} + 3\mu_{20}^3\\
C_{61} &= \mu_{61} - 5\mu_{21}\mu_{40} - 10\mu_{20}\mu_{41} + 30\mu_{20}^2\mu_{21}\\
C_{62} &= \mu_{62} - 6\mu_{20}\mu_{42} - 8\mu_{21}\mu_{41} - \mu_{22}\mu_{40} + 6\mu_{20}^2\mu_{22} + 24\mu_{21}^2\mu_{20}\\
C_{63} &= \mu_{63} - 9\mu_{21}\mu_{42} + 12\mu_{21}^3 - 3\mu_{20}\mu_{43} - 3\mu_{22}\mu_{41} + 18\mu_{20}\mu_{21}\mu_{22}
\end{aligned}$$

For modulation classification problem the above cumulants are extracted from the received signal and used for classification.

Covariance and Correlation

Covariance is the measure of strength of relationship between two random variables. If x and y are two random variables, the covariance between these two variables can be calculated as,

$$\begin{aligned}
cov(x,y) &= E[(x-\mu_x)(y-\mu_y)]\\
&= \overline{(x-\bar{x})(y-\bar{y})} \qquad (4.14)
\end{aligned}$$

where $\mu_x = \bar{x}$ and $\mu_y = \bar{y}$ are the respective means of two variables. In a special case when $x = y$, the above relation reduces to variance,

$$\begin{aligned}
cov(x,x) &= \overline{(x-\bar{x})(x-\bar{x})}\\
&= \sigma_x^2 \qquad (4.15)
\end{aligned}$$

This motivates the use of symbol σ_{xy} for $cov(x,y)$, because $\sigma_{xx} = \sigma_x^2$, where σ_x is the standard deviation. From equation (4.14),

$$\begin{aligned}
\sigma_{xy} &= \overline{xy} - \overline{\bar{x}y} - \overline{x\bar{y}} + \overline{\bar{x}\bar{y}}\\
&= \overline{xy} - \bar{x}\bar{y} - \bar{x}\bar{y} + \bar{x}\bar{y}\\
&= \overline{xy} - \bar{x}\bar{y}
\end{aligned}$$

If the two variables are uncorrelated i.e., $\sigma_{xy} = 0$, then $\overline{xy} = \bar{x}\bar{y}$, which is a property of independent variables. Hence, two independent variables are uncorrelated but the converse is not necessarily true. Correlation between two variables can be derived from the covariance,

$$\begin{aligned} cor(x, y) &= \frac{cov(x, y)}{\sigma_x \sigma_y} \\ &= \frac{\sigma_{xy}}{\sqrt{\sigma_{xx}\sigma_{yy}}} \end{aligned} \tag{4.16}$$

which is the statistical correlation between variables x and y.

4.3 Summary

This chapter gives an overview of two different case study problems used for evaluating the methods presented in this thesis. The problems belong to two different fields of science, one deals with diabetes data (a medical field problem) and the other deals with modulation data (a communication systems problem). The importance of both the problems is highlighted by discussing their impact in the respective fields and their applications. Some basic concepts of probability theory and statistics are presented, leading into the definitions of some statistical measures which will be later used for AMC problem.

Chapter 5

Feature Generation Using Genetic Programming for a Binary Classification Problem

This chapter is aimed at designing a GP-based feature generation framework for binary classification problems. In this chapter a modified version of GP, designed to improve the efficiency of standard GP is used. The modified GP introduces a variation in the genetic operator crossover and is known as *comparative partner selection* (CPS) [64]. This strategy introduces another criterion for parents selection in crossover in addition to fitness value. The goal is to maintain a diverse population throughout evolution, capable of solving all the training cases equally and this goal is achieved by allowing more diverse individuals to participate in crossover. Diabetes detection, a well known binary classification problem has been used as a case study problem. Diabetes detection is a binary classification problem where a decision is to be made whether a person has diabetes or not. In the proposed design, a GP-based system is used to generate a new single feature by using a combination of the original diabetes features. The system makes use of the simplicity of Fisher criterion for evaluating the fitness function. Although the system is used for the detection of diabetes, it can be used for any binary classification problem.

The chapter is organised as follows: The proposed GP-based system is presented initially. The basic principles of CPS method and its design for classification problems is presented. The efficiency of the inherent feature selection process of GP is validated by comparing the feature selection results of GP with various other feature selection methods. A comparison of the classification results achieved using GP generated features with original diabetes features is presented.

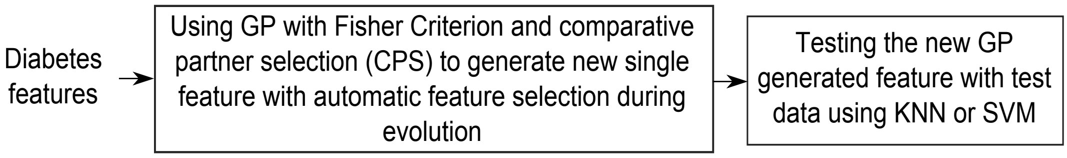

Diabetes features
Using GP with Fisher Criterion and comparative partner selection (CPS) to generate new single feature with automatic feature selection during evolution
Testing the new GP generated feature with test data using KNN or SVM

Figure 5.1: Block diagram of the proposed system

Classification results are also compared with other methods present in the literature, showing the superiority of GP based method. Finally, a conclusion based on the experiments and results is presented.

5.1 GP-Based Classification System

The block diagram of the proposed GP-based system for diabetes detection is shown in figure 5.1. The eight diabetes features (explained in Section 4.1.1) are given as input to the GP algorithm. GP makes combinations of these features during evolution such that the resultant feature is always one. At the end of generations GP returns the best combination, in terms of separation between the two classes. The task during evolution is to maximise the between class scatter while minimising the within class scatter. During evolution, the ability of new generated feature to separate the two classes is tested by Fisher criterion, a method which evaluates the statistical distribution by testing the between class scatter over within class scatter. The Fisher criterion is incorporated into GP by employing it for the task of fitness measure. A modification of standard crossover operation, called comparative partner selection (CPS) is introduced to guide the process towards optimum solution. This process helps GP to come out of any local maxima and proceed towards global maxima. Since the GP generated feature is always one dimensional, all the functions within GP (Fisher criterion, CPS, KNN, etc.) need to deal with one dimensional feature space which simplifies the design and improves the computation speed. At the end of evolution, the efficiency of GP generated feature in terms of separating the two classes is tested using either KNN or SVM classifier.

5.2 Modified Crossover in GP

There are two ways to look at the genetic make up of a GP individual, genotype and phenotype. Genotype describes the structure of an individual while phenotype represents the behaviour of an individual. The structure of an individual is described by different functions and terminals used by the individual, and the fitness value is used to represent the behaviour of individual. The main aim of the upcoming discussion is to improve the performance using behaviour of the individual so we will talk about the phenotype of the individual ignoring the genotype. In standard crossover operation, the parents are normally chosen using only the fitness values and this might be limiting as only one aspect is considered. Many partner selection strategies have been proposed in the literature, introducing other criteria for selecting parents in addition to fitness but the aim of most of those strategies has been to reduce the code growth. Burke, Gustafson, Kendall and Krasnogor [65] presented a method in which crossover is discouraged if two individuals share similar genetic material but the method proved to be good only for one problem domain. Ryan [66] introduced Ryan's Pygmy Algorithm in which first individual for crossover is selected based on the fitness value while the second one is chosen having similar fitness but smaller size. This method was reported to overcome the problem of code growth.

There are many other methods available in literature to reduce the code growth but most of these methods come at considerable computational cost with little or no improvement in performance. Some of the methods showed improvement in performance but only in a particular problem domain and as a result no method has been widely adopted. The method used here tries to explore the behaviour of an individual ignoring the structure of individual. It differs from other methods in a sense that instead of reducing the code growth it tries to increase performance by treating all training cases equally.

5.2.1 Binary String Fitness Characterisation

The ability of an individual to solve all the training cases is represented by one single parameter, fitness function. As pointed out before representing the ability of an individual to solve a full training dataset in terms of a single value might be limiting. A typical fitness function is the sum of the errors made by an individual for different training cases. Imagine two individuals, one performing very well for certain training cases and other performing averagely for most of the training cases might get same overall final fitness value. In another case one individual performing well for half of the training examples and another individual performing well for the other half will get the same fitness value, although, behaviourally they are different. If a crossover takes place between two individuals sharing same weaknesses, their off-spring are likely to have same weaknesses because of the inheritance. In this way the behavioural diversity of the population will reduce and so will the ability of individuals to solve all training cases. As the population evolves, the behavioural diversity to solve all the training cases across the population will continue to decrease and in the process the population might loose the genetic material to solve some training cases. In order to to identify the strengths and weaknesses of individuals for particular training cases in addition to over all fitness value, Binary String Fitness Characterisation (BSFC) is introduced. It was shown in [64] that the problems converging sub-optimally performed well for only particular training cases and lacked the genetic material to solve other training cases. An ideal solution should perform well for all the training cases in addition to over all good fitness value and for this purpose BSFC is introduced.

In BSFC, a training case in which an individual performs well is considered a strength and a case an individual is unable to solve is considered a weakness. Consider a logical problem where the output is either true or false, a correct output by an individual would mean a strength and a wrong output would mean a weakness. A binary string is attached with each individual and the length of this binary string is equal to the length of training dataset. Each bit is attached with one training case and a one is placed in the binary string bit if corresponding training case is solved and a zero is placed in the bit if the individual fails to solve the training case. Assigning strengths and weaknesses to quantify individual's ability in the form of binary string is named as Binary String Fitness Characterisation. If all the training cases are solved, the binary string will have ones in all the bits.

Let us consider another problem where output is not binary but still we have a target value for each training case. The fitness function is the sum of the errors made by an individual for each training case and it is represented as $e(t)$. The binary string could be assigned during the process of fitness evaluation. While it is possible to assign a one to the binary string bit if the individual's output is within a small margin of the target output value, it will require to assume a predefined threshold which means that initial population of individuals might be dominated by zeros. A more generalised approach would be to relate strengths and weaknesses with fitness value. If the output error for a training case is less than the mean fitness value (fitness value is sum of the error for all test cases), it is considered a strength otherwise weakness. Figure 5.2 shows the graphical representation of the process. The calculation of different bits of binary string in the form of equation can be written as

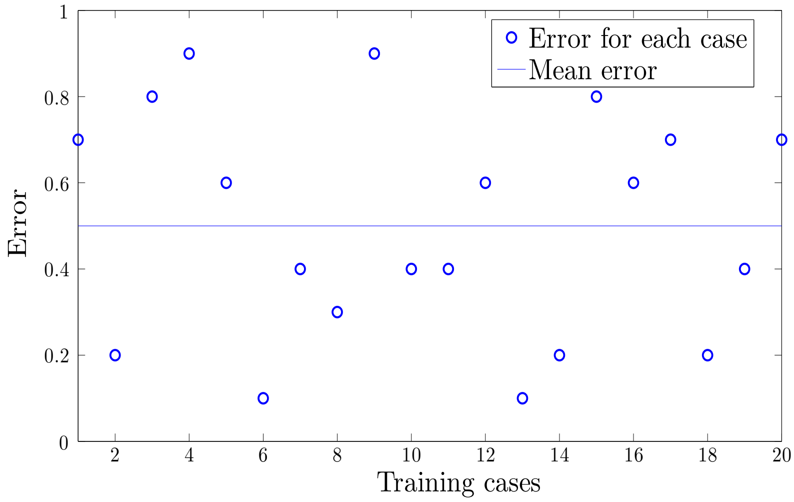


Figure 5.2: Each circle is the error made by an individual for each training case. The solid line represents mean fitness or mean error. All the circles below this line will get a one in the binary string and circles above the line will get zeros. The resultant binary string will be 01000111011011000110

$$b_i = \begin{cases} 1 & \text{if } e(t_i) \leq mean(fitness) \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where b_i is the binary bit of binary string for the training case i, $e(t_i)$ is the error for training case i and $fitness$ represents the overall error for all the cases. An ideal solution will have a binary string consisting of ones only. The above equation gives more deeper details about the behaviour of GP individual compared to the fitness value.

For classification problems the assignment of binary string bits is not straight forward as there are no target values. The target in such a problem is to separate the two classes and the output points which prove more effective in terms of classification should be rewarded. The binary string for diabetes detection is updated as follows: an output point close to the mean of output distribution of the relevant class is considered as strength and gets a one in the binary string. Similarly, a point away from the mean distribution is considered a weakness and gets a zero. The question arises how to decide which points are closer and which points are far? In this study mean and standard deviation of output distribution are used to define close and far points.

The mathematical formula for updating binary string of a single class in any classification problem can be written as

$$b_i = \begin{cases} 1 & \text{if } O(t_i) \lesseqgtr (mean(O(t)) \pm (k * std(O(t)))) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where b_i is the binary string bit for the training case i, $O(t_i)$ is the output of an individual for training case i, k is a constant which determines the distance within which an output point is considered strength, $*$ is the multiplication sign, *std* represents the standard deviation, $O(t)$ is the output vector for a particular class and *mean* is the function which takes the mean of a vector. According to this equation if the output point is within certain (k) standard deviations of the mean output, it is considered as strength otherwise weakness. The value of k is an important parameter as it determines the maximum distance, a point could be away from the mean and still classified as strength. The value of k is problem dependent and we will use a value that gives the best results for a particular training data after trying a few values.

5.2.2 Comparative Partner Selection

The previous section explains how to assign a binary string to an individual. After getting this binary string the next step is to use this binary string for selecting parents for crossover. A simple approach has been used here to accomplish this task in order to minimise the computational cost. In order to improve the population phenotype variance and ultimately improve the performance of individuals, the weaknesses of individuals must be eradicated. Now that we have a binary string explaining strengths and weaknesses of individuals, it could be used to decide which individuals should have a crossover. A crossover should be discouraged if two individuals share similar weaknesses and a crossover should be encouraged if their strengths are in different areas. The procedure for selecting the parents for crossover in CPS is the same as in standard GP. The two parents for crossover are chosen using any standard selection operator (Roulette wheel, Tournament, Rank selection) but an additional criterion for crossover is introduced in CPS and just because two individuals are selected for crossover does not guarantee they will definitely have a crossover. The additional CPS criteria tries to make sure that the crossover of two parents is beneficial.

Computationally inexpensive logic operations can be used for comparison between two individuals due to the binary nature of the binary string. The logic

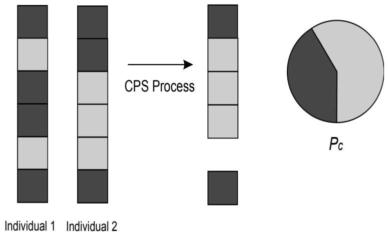


Figure 5.3: A graphical representation of the CPS process

operation should discourage crossover if two individual share similar weaknesses (a logical NOR operation, which gives a one if both bits are zero, otherwise zero) and should encourage crossover if one individual is strong in an area where the other is weak (a logical XOR operation, which gives a zero if two bits are same, otherwise one). The probability of crossover can be calculated using these two logic functions and a simple formula is given below

$$P_c(I_1, I_2) = \frac{\sum XOR(b_1, b_2)}{\sum XOR(b_1, b_2) + \sum NOR(b_1, b_2)} \tag{5.3}$$

where P_c is the probability of crossover for Individuals I_1 and I_2, b_1 and b_2 are the binary strings of the two parents chosen for crossover, and Σ represents the summation of each bit in the binary string. The denominator can be replaced by a single $NAND$ operation so the above equation can be written as

$$P_c(b_1, b_2) = \frac{\sum XOR(b_1, b_2)}{\sum NAND(b_1, b_2)} \tag{5.4}$$

The CPS process is a stochastic process so the individuals which are different in terms of CPS are more likely to have a crossover than the individuals similar to each other. The process of CPS is shown in graphical form in Figure 5.3 where light and grey block represent strength and weakness respectively.

The process of CPS can be described as: a single parent is chosen using standard selection criterion as primary partner and a second parent is chosen as secondary partner using the same criterion. The probability of crossover between the two parents is calculated using equation (5.4) and it is decided stochastically using this probability whether the crossover takes place or not. If the crossover takes place then the new individuals (offspring) are inserted in the new population and a new primary partner is selected. If the crossover does not take place, a new secondary partner is selected without changing the primary partner and the above process is repeated. If the CPS process is unable to find a suitable secondary partner after evaluating N/2 individuals (N is the total population size), a secondary parent is randomly selected using the standard selection method and a crossover takes place ignoring the CPS criteria. In order to penalise the CPS for not finding a suitable partner, the probability of crossover is decreased by 1/N and the probability of mutation is increased by 1/N in the current generation. At the beginning of the next generation, the probabilities of crossover and mutation revert back to the initial values used at the start of the evolution. A flow chart of the CPS process is shown in Figure 5.4.

5.3 Case Study: Detection of Diabetes

In this section all the experiments and results are presented. The use of GP in classification problems is not new and it has been used quite a lot in the past for classification problems and the details can be found in a survey presented by Espejo, Ventura and Herrera [30]. The CPS strategy was applied to some simple benchmarking regression problems in [64]. In this study this strategy has been applied to a real world diabetes classification problem [67]. The diabetes dataset and its attributes were explained in Chapter 4. Initially a comparison between GP and GP with CPS is presented. As this is a binary classification problem, the inverse of Fisher criterion (equation (3.6)) has been used as the fitness function because of its simplicity, especially for binary classification problems. A ploy to reduce the computational complexity by using a variable population size is presented next. The classification accuracy achieved and its comparison with other methods present in the literature is presented at the end.

5.3.1 The Binary String for the Diabetes Problem

The binary string associated with each individual for CPS is updated slightly differently for the diabetes problem. In equation (5.2) a strategy for assigning binary string was presented which favoured the output points within a certain standard deviation of the mean of the output distribution. The equation is useful for multi-class classification problem but it can be modified for a binary classification problem. As there are only two classes in a binary classification problem, the output point away from the mean of one class could be towards or away from the output distribution of the second class. If it is away from the second class it is considered a strength and gets a ’1’ in the binary string. If it is towards the second class, a second check is used to assign binary string value. If the point is within half standard deviation (k=0.5) from the mean it gets a ’1’ in the bi-

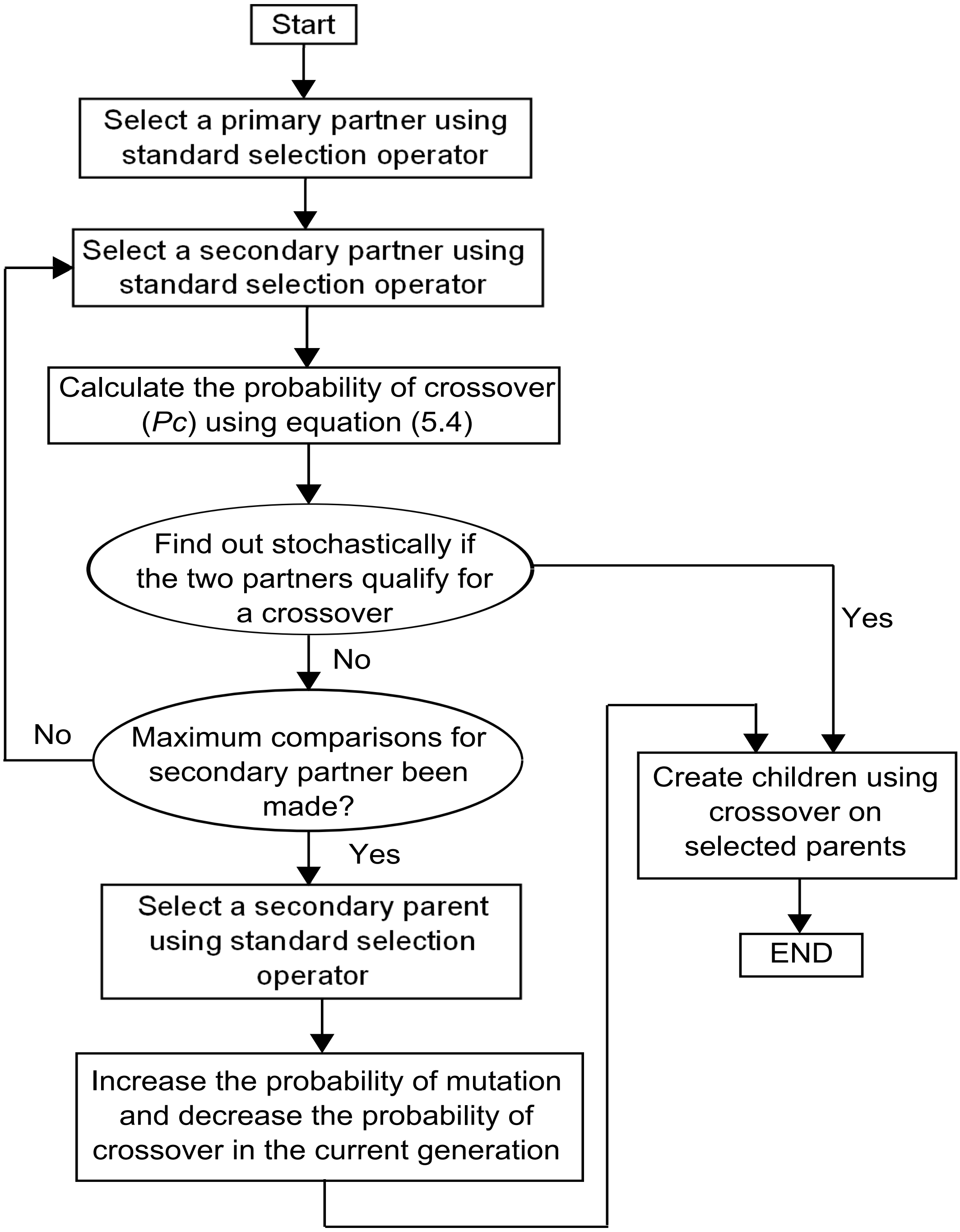

Start
Select a primary partner using standard selection operator
Select a secondary partner using standard selection operator
Calculate the probability of crossover (*Pc*) using equation (5.4)
Find out stochastically if the two partners qualify for a crossover
Yes
No
No
Maximum comparisons for secondary partner been made?
Yes
Select a secondary parent using standard selection operator
Increase the probability of mutation and decrease the probability of crossover in the current generation
Create children using crossover on selected parents
END

Figure 5.4: A flow chart of the CPS process

nary string otherwise '0'. The process is shown in graphical form in Figure 5.5. Two random Gaussian distributions are used to explain the process. The solid line distribution represent class 1 and dotted line distribution represents class 2. For class 1 if a point falls behind the vertical solid line (mean(class1) + k * std(class1)), it gets a '1' in the binary string otherwise '0'. Similarly for class 2 if a point falls behind the dotted line (mean(class2) - k * std(class2)), it gets a '0' in the binary string otherwise '1'. The value of k is chosen as 0.5 in this study. For diabetes problem, class 1 and class 2 distributions could be either diabetes data distribution or healthy data distribution. The actual position of the data distribution (ahead or behind) does not matter and the objective is to separate them.

5.3.2 Comparison Between GP and GP with CPS

Two types of experiments are performed for comparison, one with smaller number of generations, having bigger population size (Exp-1), and the other with relatively larger number of generations but with smaller population size (Exp-2). The number of generations for Exp-1 is 125 with a population of 100 individuals while the number of generations for Exp-2 is 200 with a population size of 25. Both of these experiments were conducted for diabetes classification using the full diabetes dataset. All the other parameters of GP for these experiments are given in Table 5.1.

The results of the Exp-1 are shown in Figure 5.6 which shows the comparison between GP and GP with CPS in terms of fitness value where a lower fitness means better classification. GP was evolved 40 times and each point on the curve is an average of 40. It is clear from the Figure that the fitness of GP with CPS is lower compared to the GP, although, the difference between the final fitness values is not huge. An important observation is that the fitness value of the standard GP is settling at the end of the generations while the fitness value of GP with CPS is still decreasing. In order to investigate if the fitness continues to decrease (which means better performance), if allowed to go over more generations, Exp-2 was performed.

Figure 5.7 shows the results of Exp-2. For the first 75 generations GP performs better compared to GP with CPS, however, post 75 generations GP with CPS performs better. The final fitness achieved by GP with CPS is better compared to GP. The improvement in fitness is marginal but we will see later that this marginal improvement will be significant in terms of classification accuracy. An

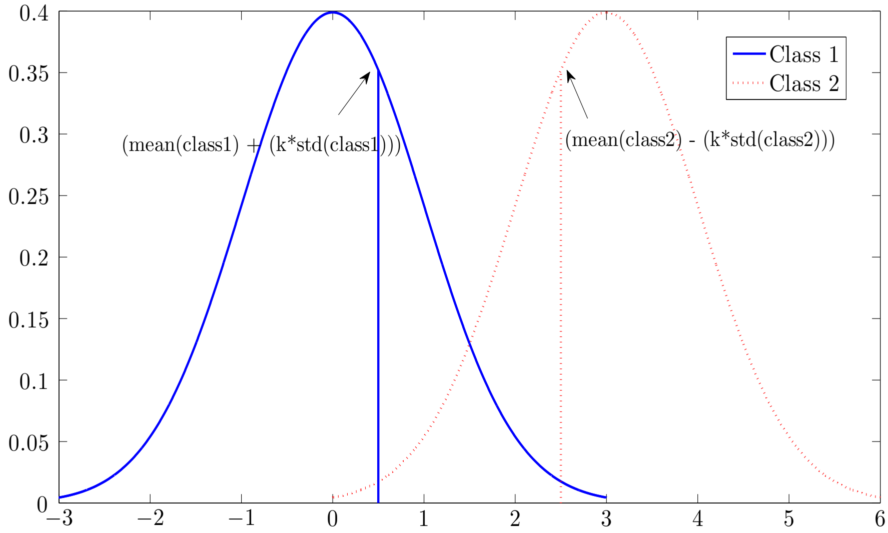


Figure 5.5: Binary string evaluation for diabetes problem

interesting point to note from this Figure is that after about 75 generations, the standard GP seems to have locked in and there is hardly any improvement in fitness upto 200th generation. It is believed that this lack of improvement after 75 generations is due to the reason that GP lacks the genetic material which can help to solve some of the training cases. In contrast GP with CPS finds a way to improve the fitness and does not get locked in due to the fact that it maintains a diverse population throughout the evolution, capable of solving all the training cases [64].

In order to demonstrate the effectiveness of GP with CPS in terms of classification accuracy, the solutions produced by GP with CPS and GP were tested using KNN classifier. The data was divided into 90% and 10% training and test data. Both GP variations were evolved 40 times for the training data and 40 solutions produced were tested with test data using KNN classifier. The training data was used by KNN classifier as reference samples. The classification accuracy is defined as

$$\text{Classification Accuracy (\%)} = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.5}$$

where TP is the number of patients (diabetes) detected as patient, FP is the number of healthy persons detected as patient, TN is the number of healthy persons detected as healthy and FN is the number of patients detected as healthy.

Table 5.1: Parameters used for the experimental work

Parameter	**Standard Value**
Number of Generations	125, 200 (Exp-1, Exp-2)
Population Size	100, 25 (Exp-1, Exp-2)
Function Pool	(plus, minus, times, mydivide, mysquare, mysqrt, reciprocal, sin, cos, myasin, myacos, tan, tanh, negator, abs, mylog)*
Terminal Pool	8 attributes
Genetic Operator	Crossover, Mutation
Operator Probability	0.6, 0.4
Tree Generation	Ramped half-and-half
Initial Maximum Depth	6
Maximum Depth	28
Selection Operator	Roulette
Elitism	Half-elitism

* All the functions starting with ”my” are protected functions.

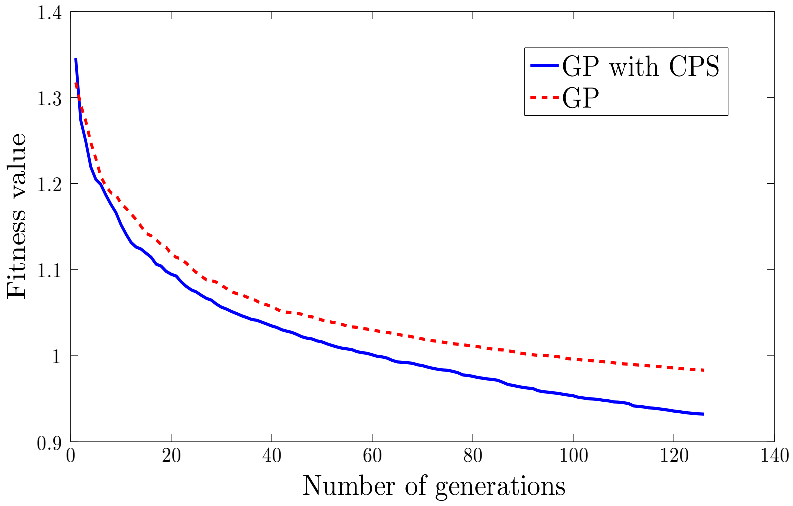


Figure 5.6: Average fitness of the best of generations for 125 generations averaged over 40 runs using a population of 100.

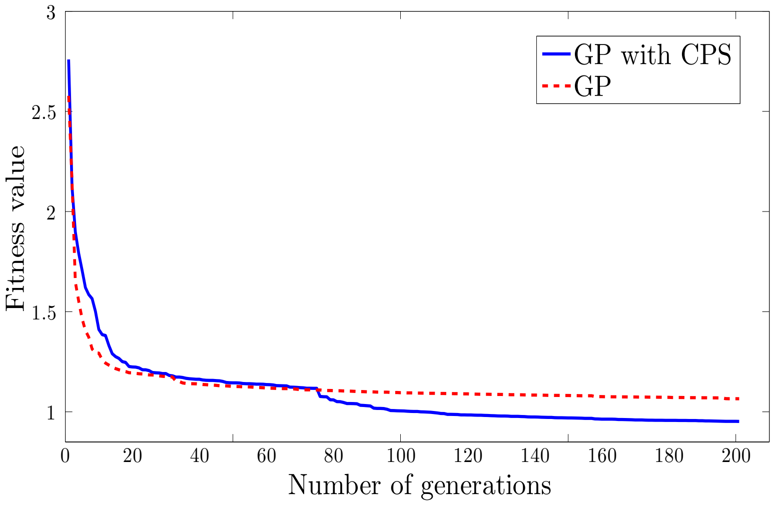


Figure 5.7: Average fitness of the best of generations for 200 generations averaged over 40 runs using a population of 25.

Table 5.2: Comparison of the classification accuracies using GP and GP with CPS

Method	**Accuracy (%) ± Std**
GP (Exp-1)	78.2 ± 2.5
GP (Exp-2)	78.4 ± 2.5
GP with CPS (Exp-1)	79.1 ± 2.1
GP with CPS (Exp-2)	79.7 ± 2.0

The results for Exp-1 and Exp-2 are summarised in Table 5.2. The classification accuracy (written as accuracy in the Table) as well as standard deviation (std) for each experiment is given in the Table. The classification accuracy for GP with CPS is better compared to GP in both the cases. Moreover, the standard deviation is also lower which makes GP with CPS more robust.

Complexity comparison between GP and GP with CPS

The additional computations added by CPS are the evaluation of the binary string and the calculation of P_c. Although, these two quantities add extra computations to GP, the binary nature of both of these quantities does not add much to the computational complexity when compared to other methods present in literature for evaluation of phenotype behaviour. There are two ways the additional complexity added by CPS can be compared with GP, i) additional memory required, and ii) additional time required.

Each individual in GP has some memory requirement for storing different parameters related to that individual, e.g., fitness value, number of nodes, origin of creation, string representation of tree, tree depth, etc. The additional memory required by CPS process is for storing binary string of each individual. The additional memory requirement (in bits) introduced by binary string is $m = NB$, where N is the population size and B is the length of the binary string of one individual. Considering the memory cost of all other parameters related to an individual, the additional cost introduced by the binary string is marginal.

The second cost added by CPS is the additional time required to evaluate the binary string and to find the right partner during CPS operation. For Exp-1 (in the previous section), the average time taken for training one GP run was about nine minutes for standard GP and about 11 minutes for GP with CPS. It is clear from this that the additional resources required for CPS are on average less than 20% in terms of computation. Most researchers agree that the biggest portion of time during evolution is spent during fitness evaluation of any generation (as demonstrated in the next section) and it is more significant for complex problems. Since diabetes detection is also a complex problem (as demonstrated by achieved classification accuracies), the additional resources required by CPS are negligible in comparison. In a nutshell, the additional complexity introduced by CPS process is marginal compared to other processes involved in GP algorithm.

5.3.3 Complexity Reduction Using Variable Size Population

The discussion in the previous section proved that the GP with CPS performs better than the GP. An interesting observation from the Figure 5.7 is that the final fitness value achieved in Figure 5.7 is a not much different compared to the value in Figure 5.6, although it had more generations. This highlights the advantage of having a higher number of individuals in maintaining a good diversity in the population (Exp-1 had 100 individuals while Exp-2 had only 25). If the population for Exp-2 is increased, the complexity of the algorithm will increase, resulting in a longer training time. In order to enjoy the advantage offered by a large population size while keeping the complexity in control, three more sets of experiments (Exp-A, Exp-B and Exp-C) were performed. The maximum number of generations for all these experiments is 200, with Exp-A and Exp-B having a population size of 100 and 25 respectively. The population size in Exp-C is variable: it starts with a larger population (to maintain diversity) of 100 individuals, and after 50 generations the number of individuals is reduced to 25 (to reduce the

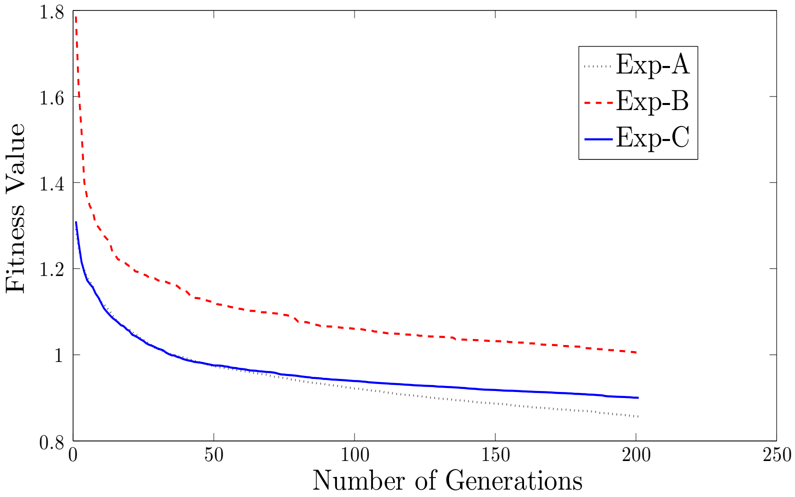


Figure 5.8: Average fitness of the best of generations for 200 generations averaged over 40 runs using different population sizes.

complexity), by picking 25 fittest individuals (based on the fitness value) from 100 individuals. This way GP is given a fair chance to explore a large search space for the first 50 generations.

Figure 5.8 shows a comparison between three experiments based on the fitness value. It is clear from the Figure that the fitness curves for Exp-A and Exp-C follow similar patterns and the difference in the final fitness value is only 0.04 whereas the fitness curve for Exp-B is quite different from the other two. Since the performance of Exp-C is very close to the performance of Exp-A, it can replace Exp-A if it has lower computational complexity. Two parameters are used for comparing the computational complexity of the experiments: (1) the mean time taken (seconds (s)) to create a new generation during the evolution, averaged over 25 runs (2) the mean time(s) to calculate the fitness of individuals in each generation during evolution, averaged over 25 runs. The value of parameter 1 for Exp-A, B and C is 9.4 s, 1.8 s and 4.7 s respectively, whereas the value of the parameter 2 is 69.8 s, 11.2 s and 27.1 s respectively. These two parameters indicate that the complexity of Exp-C is up to half or less compared to Exp-A. Above discussion clearly demonstrates that the performance of Exp-C is fairly close to Exp-A with considerably low computational complexity. On the basis of the results of this section all the experiments for diabetes classification are performed using Exp-C employing GP with CPS.

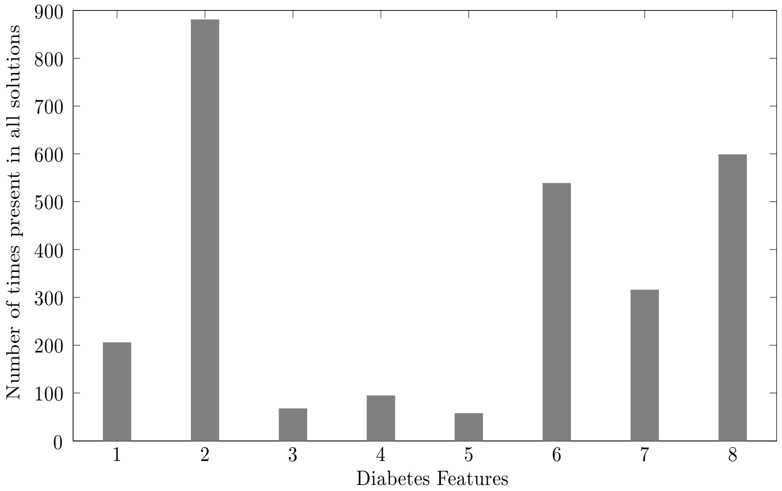


Figure 5.9: Occurrence of diabetes input features in GP solutions.

5.3.4 Feature Selection Results

The diabetes dataset was explained in detail in Chapter 4. There are eight diabetes features as mentioned in Section 4.1. Instead of using all the features for GP training, the effectiveness of different features was tested using the feature selection methods presented in Chapter 2 and features were used for GP training in decreasing order of importance using sequential forward selection method. In addition to other feature selection methods GP was also used in feature selection process. For feature selection GP was trained using all eight input features and we found out the number of times any feature was used in the final solution returned by GP. In the experiment, the number of generations used was 200 and the number of individuals in each generation was 25. The experiment was performed 40 times and total number of times a feature appeared in all the solution was counted. Figure 5.9 shows the appearance of different features in all the solutions. One can see that use of the feature 2 in final solutions is far more than any other feature. After feature 2, the second best feature used by GP was feature 8, followed by feature 6 and 7. Rest of the features were not used a lot in the final solutions. The results of all the feature selection methods are summarised in Table 5.3. The features for the detection of diabetes are arranged in decreasing order of importance. It is clear from the Table that all of the methods have resulted in approximately similar ordering with slight differences.

All of the above feature selection methods help in finding useful features from a group of diabetes features but they do not reduce dimensions by removing less

Table 5.3: Feature importance using different feature selection methods

Method	**GP**	**T-Test**	**K-S Test**	**F-Score**	**KLD Test**	**Finalised Order**
Ordered List	2	2	2	2	2	2
	8	6	8	6	6	6
	6	8	6	8	1	8
	7	1	1	1	5	1
	1	7	5	7	8	7
	4	5	7	5	7	5
	3	4	4	4	4	4
	5	3	3	3	3	3

useful features. The dimension reduction is left to the user who can decide to remove some features based on the results of feature selection methods. After getting the order of importance for diabetes features, the next step is to use them in training GP. We have followed sequential forward selection process for training. We start training GP with most useful feature and get the classification results. The next important feature is then added to the training data and GP is trained again. This process continues until we reach maximum number of features. In this research, features were used one by one in the following order; 2, 6, 8, 1, 7, 5, 4, 3. This order has been chosen using results of feature selection methods from Table 5.3.

In addition to the feature selection, feature generation was also performed using PCA. Since the first few principal components account for most of the variance of the original data, the first principal component was considered as the most valuable feature followed by the second principal component and so on. The PCA generated features were also used for training the GP using sequential forward selection method. The full process for training GP is shown in Figure 5.10.

5.3.5 Preparation of the Training and the Test Data

Two types of testing approaches have been used in the literature for diabetes classification. Some of them used k-fold cross validation method by dividing the training and test data into k mutually exclusive subsets while some of them used just one training and test data partition (conventional method). The performance achieved using conventional method could be quite different compared to $k\times$fc

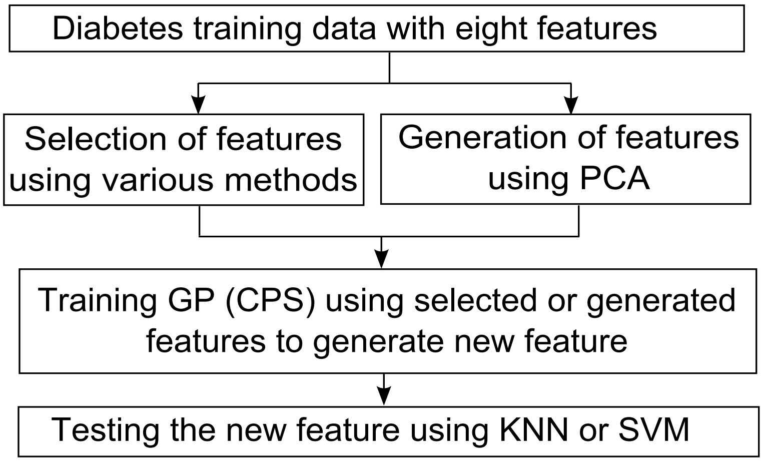

Diabetes training data with eight features
Selection of features using various methods
Generation of features using PCA
Training GP (CPS) using selected or generated features to generate new feature
Testing the new feature using KNN or SVM

Figure 5.10: System model for diabetes classification.

method and we will demonstrate it in this study.

In this study the dataset was divided into 10 different training and test datasets with 90% and 10% training and test data partition. This division was done to carry out a 10-fold cross validation test (10×fc), and each GP solution for a specific training dataset was tested with corresponding test dataset. Each training and test dataset was then divided into 8 subsets where in each subset one feature was added one by one (sequential forward selection) according to the order mentioned in previous section. GP was used to produce the solutions for all these subsets. The solutions produced by GP were later tested by a classifier. The training setup is shown in Table 5.4.

5.3.6 Classifiers Used for Testing

The classifier have been used in two ways in order to demonstrate the effectiveness of GP generated feature.

- In the first case, the original diabetes features have been used directly as inputs of the classifiers. The features are used one by one using the sequential forward selection method and the performance is calculated. In this case information extraction is the responsibility of the classifier and it is performed by favouring the discriminating features and rejecting the interfering features during the learning process.

- In the second case, a new feature is generated from the original features by making combination of the existing features using an evolutionary GP algorithm. GP is trained for each combination of input features using forward selection method and a new feature is generated for each combination. This

Table 5.4: Training setup

Training Datasets	**Subsets**	**GP Solutions**
10 (Each dataset contains 90% of diabetes data with 8 features)	8 (1 for each dataset and each subset adds diabetes features one by one)	400 (5 solutions for each subset and 40 solutions for each dataset)

new feature is then used as an input for the classifiers and the performance is calculated.

The classifier used are KNN and SVM.

K-Nearest Neighbor (KNN)

KNN was explained in Chapter 2 and for the diabetes problem the training data has been used as reference samples for KNN. Since the best choice of k for KNN classifier depends on the dataset used, an experiment was conducted to find out the best value of k for the diabetes dataset. Few odd values of k for a random diabetes test dataset were tested and the results are presented in Figure 5.11. Figure shows that the performance is maximum when k is 21 and does not improve after that so the same value of k was used for all the experiments.

Support Vector Machine

SVM was also explained in Chapter 2. After trying few kernel functions (linear, polynomial, multilayer perceptron, radial basis function), radial basis kernel function was used in SVM for making hyper plane for the diabetes problem.

5.3.7 Results and Discussions

This section explains all the experiments and results using the original diabetes features, GP generated features and the PCA generated features. Some of the best trees obtained for GP are included in Appendix A.1. Table 5.1 gives all the parameters used for GP during the experiments. Results are presented in terms of classification accuracy, sensitivity and specificity. The classification accuracy was defined in equation (5.5), sensitivity and specificity are defined below:

$$\text{Sensitivity}(\%) = \frac{TP}{TP + FN} \tag{5.6}$$

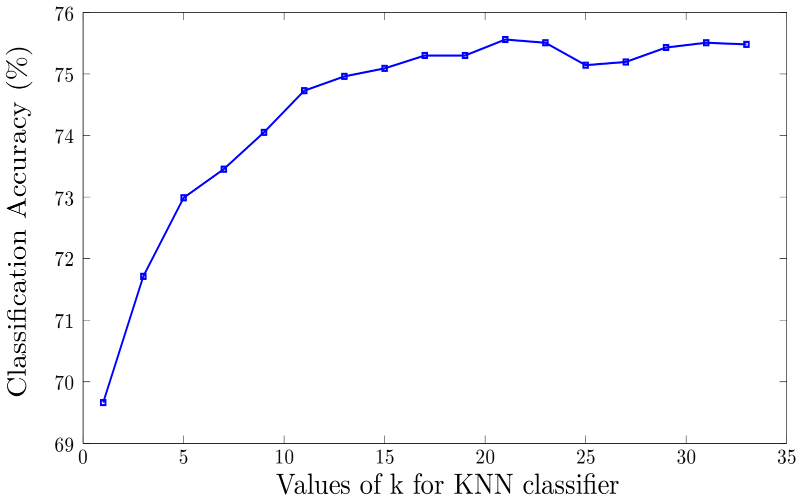


Figure 5.11: Classification accuracy for different values of k for KNN classifier.

$$\text{Specificty}(\%) = \frac{TN}{TN + FP} \tag{5.7}$$

where all the variables are the same as in equation (5.5). In order to compare the performance of GP generated feature with original diabetes features the performance of the original features tested by KNN and SVM is also presented in addition to the performance of GP feature tested by KNN (GP-KNN) and SVM (GP-SVM). Figure 5.12 shows that the maximum performance (80.5% GP-KNN) can be achieved using the first five ordered features and the performance decreases if more than five features are added for most of the methods (except KNN) which supports the opinion that addition of redundant features could add noise to useful features. This is interesting because it is always argued that the built in feature selection ability of GP should be sufficient and GP should automatically filter out noisy features. Although, it was shown previously that the most useful features are heavily utilised by GP for generating new features and less useful features are seldom used but the less useful features still had some contribution. Figure 5.12 demonstrates that the addition of last three features (feature 5, 4 and 3) causes a reduction in accuracy, demonstrating that the less useful features introduce noise. Nevertheless, the Figure shows that GP generated features still provide best classification results and outperform original diabetes features. The performance shown in Figure 5.12 is using 10×fc method where each classification accuracy is the mean of the classification accuracies of 10 datasets.

Classification accuracies for different datasets averaged over all features are shown in Figure 5.13 where one can see that the variation in classification accuracy is quite high for different datasets which proves that the performance achieved using conventional method (using one partition of training and test datasets) could be misleading. One of the possible reasons for such variation could be the missing values in the dataset. Although in all previous literature (except [68]) nobody else has mentioned missing values but we believe some of the zeros in the dataset represent missing values, e.g., there are quite a few zeros for blood pressure and triceps thickness which cannot be true. But according to the source of the dataset, donor did not mention any missing values. In order to compare the conventional performance with other methods, the conventional performance using dataset 3 is shown in Figure 5.14. The maximum classification accuracy achieved using conventional method (87% GP-SVM) is much higher than 10×fc method (80.5%).

Figure 5.15 shows the classification accuracy achieved using PCA where horizontal axis shows the number of principal components used. As maximum variation is within first few principal components, only first five principal components were used out of eight. The principal components were also divided into ten training and test dataset for evaluating 10×fc performance. The maximum classification accuracy achieved was 77% using all five principal components. One of the reasons for using PCA was the classification accuracy (89.5%) mentioned in [69] which is the best accuracy reported so far. However, this accuracy was questioned in [70] where exactly same method was used but the classification accuracy achieved was only 66.8%. Although a different method has been used in this study (PCA with GP) but the performance using PCA features is much lower than the performance mentioned in [69].

The possible reason for the poor performance of PCA could be the inability of PCA to deal with outliers especially in the presence of missing data. Principal components are sensitive to outliers and the presence of one bad outlier in the data can seriously affect the performance of PCA. In order to show the presence of outliers in diabetes data one of the test datasets among the 10×fc PCA datasets was chosen as a sample. Figure 5.16 shows the first two principal components where outliers can easily be seen.

The performance in terms of classification accuracy, sensitivity and specificity for 10×fc and conventional method is given in Table 5.5. In any classification system it is important that both sensitivity and specificity should be high. Since sensitivity represents how accurately a patient is classified as patient, it is more important in the present case. Although the sensitivity is reasonable in the

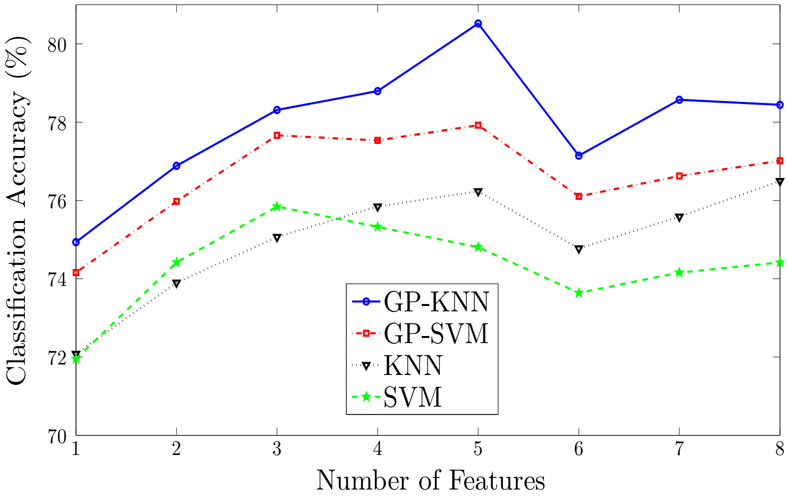


Figure 5.12: Classification accuracy (10×fc) using different number of features ordered as (2, 6, 8, 1, 7, 5, 4, 3).

conventional method (74.1%) but it is quite low for 10×fc method (55.4%) while the specificity is high in both the cases (94%). The reason behind this is that the dataset is imbalanced which means that the data samples of one of the class dominate the dataset. In diabetes dataset the number of samples of diabetes is only 268 compared to 500 healthy data samples. Because of the dominance of healthy samples in the data, the solutions of GP are biased towards healthy class as it results in high classification accuracy. This problem will be discussed in detail in the next chapter and different solutions for dealing with imbalanced data will be presented and evaluated.

5.3.8 Comparison with Existing Methods

Table 5.6 gives a comparison of the proposed method with existing methods and a complete list of all the proposed methods for diabetes classification is available in [71]. Polat, Gunes and Arslan [71] produced a classification accuracy of 79.2% (10×fc) using Generalized Discriminant Analysis (GDA) and Least Square SVM (LS-SVM). GDA was used at the first stage as a pre-processing step and LS-SVM was used at the second stage for classification. They reported a classification accuracy of 79.2% which is less compared to the accuracy achieved in this research (80.5%). Polat and Gunes [69] reported 89.5% accuracy, using PCA as a feature extractor, which has been questioned in [70] and is endorsed by the current investigation. Temurtas, Yumusak and Feyzullah [70] produced 79.6%

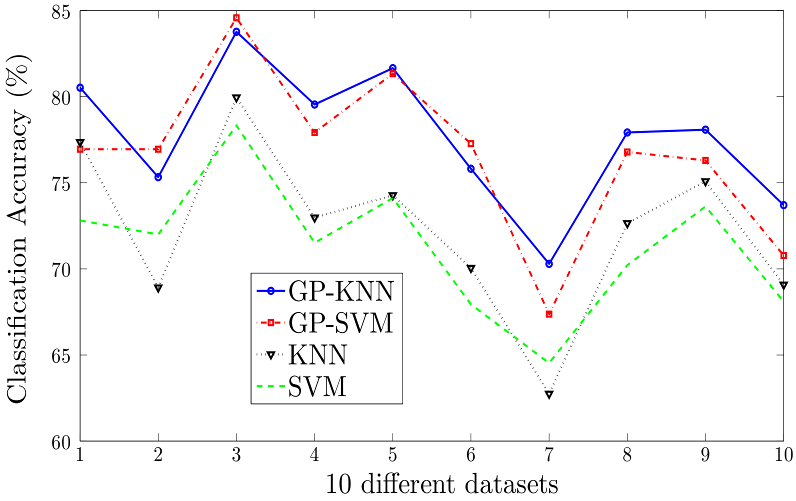


Figure 5.13: Classification accuracy using different datasets trained by GP and tested by KNN and SVM.

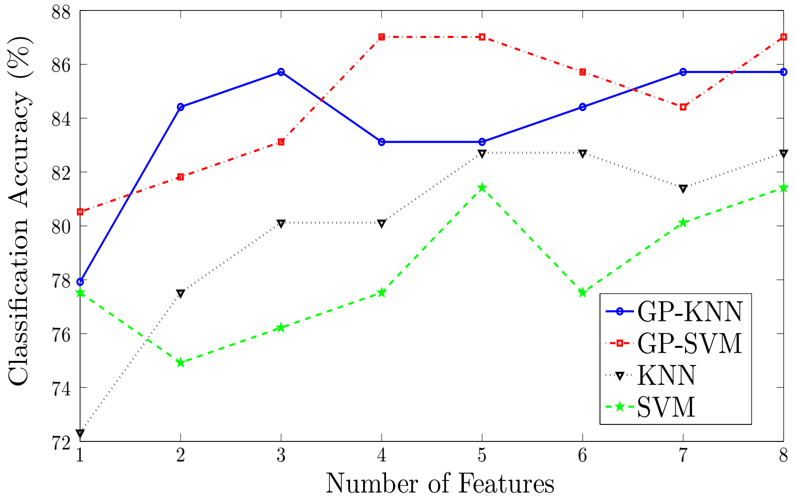


Figure 5.14: Classification accuracy (conventional) using different number of features ordered as (2, 6, 8, 1, 7, 5, 4, 3).

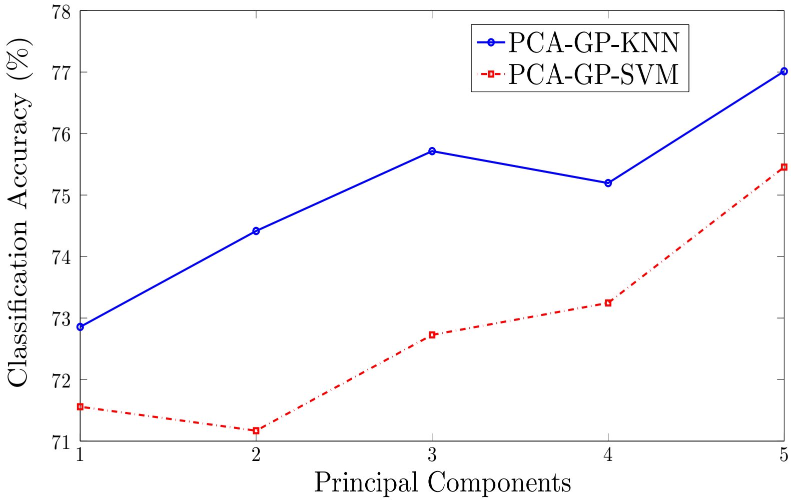


Figure 5.15: Classification accuracy (10×fc) using the first five principal components trained by GP and tested by KNN and SVM.

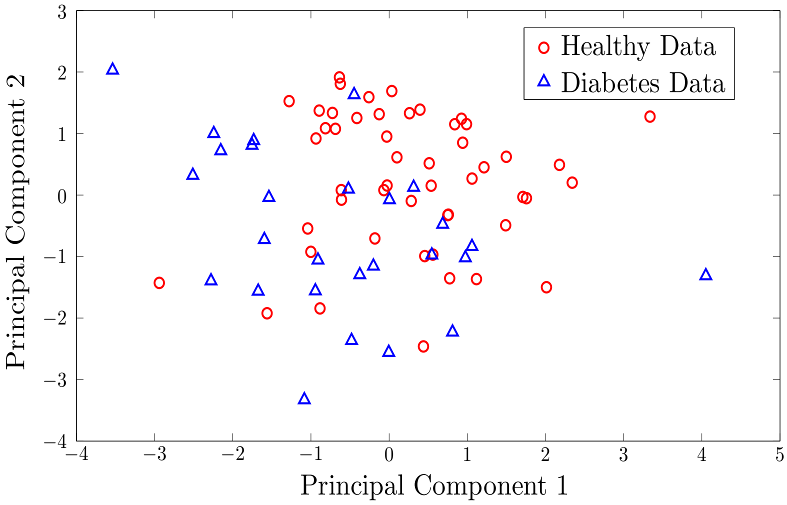


Figure 5.16: The data distribution of sample test set using the the first two principal components.

(10×fc) and 82.4% (conventional) accuracy using Multi Layer Neural Network (MLNN) with Levenberg-Marquardt (LM) training method which is inferior to the accuracy achieved here (80.5% and 87.0%). Gadaras and Mikhailov [72] reported classification accuracy of 92.3% using fuzzy rules but they used a different training and test data partition (50%/50%). Moreover, they used 2×fc validation which as demonstrated in the previous subsection could have a huge impact on the performance.

A layered GP based method was used in [73]. They used a reduced dataset, without mentioning how did they reduce it. Moreover, it was mentioned that the missing values are completed, without giving any details. Nevertheless, the best accuracy they reported was 73.54%, which is less compared to ours (80.5%). Brameier and Banzhaf [74] used linear GP for diabetes classification with main focus on reducing the training time and complexity of GP algorithm. They reported 81% best accuracy which is less compared to the best accuracy achieved in this study (87%). Lee and Wang [75] used fuzzy ontology based system for classification. They divided the data into different groups based on the age of patients (feature 8) and reported classification accuracies separately for each group. They reported a higher classification accuracy for people with higher age. However, the average of all the accuracies is 85% (conventional), which is less compared to ours (87%).

Kala, Vazirani, Khanvalkar and Bhattacharya [76] reported classification accuracy of 82.4% (conventional) using ANN which is less compared to our prposed method (87.0%). Lekkas and Mikhailov [77] produced 79.4% accuracy (conventional) using fuzzy classification and under the same conditions we produced 87.0%. In [78], Goncalves, Vellasco, Pacheco and de Souza used Inverted Hierarchical Neuro-Fuzzy System (IHNFS) and reported 78.6% classification accuracy (conventional). Balakrishnan, Narayanaswamy and Paramasivam [68] reported 98.9% accuracy but they did not use the full dataset. They reported missing values in the original dataset without mentioning the source of dataset and used a reduced dataset, ignoring the missing values without giving any details which values were ignored. Ignoring the missing values (which results in smaller training and test dataset) can seriously affect the performance as mentioned in the previous section. It is clear from the Table 5.6 and above discussion that the proposed method produces superior results when compared with others methods using the same training and test data partitions.

Table 5.5: Performance

Method	**Accuracy (%)**	**Sensitivity (%)**	**Specificity (%)**
10×fc	80.5	55.4	94
Conventional	87.0	74.1	94

Table 5.6: Comparison of the proposed method with existing methods

Author	**Method**	**Accuracy (%)**
K. Polat et al. [71] (10×fc)	GDA & LS-SVM	79.2
K. Polat et al. [69] (10×fc)	PCA & ANFIS	89.5*
Temurtas et al. [70] (10×fc)	MLNN & LM	79.6
Temurtas et al. [70] (conventional)	MLNN & LM	82.4
Gadaras et al. [72] (2xfc)	Fuzzy Rule	92.3**
Kala et al. [76] (conventional)	RBFN	82.4
Lekkas et al. [77] (conventional)	Fuzzy Classification	79.4
Goncalves et al. [78] (conventional)	IHNFS	78.6
This Study (10×fc)	**GP-KNN**	**80.5**
This Study (conventional)	**GP-SVM**	**87.0**

* Questioned by this research and [70] as it is not reproducible, ** Used a different training and test data partition (50%/50%)

5.4 Summary

A novel method of assigning strengths and weaknesses to GP individuals for classification problems has been introduced in this chapter. This measure (BSFC) gives meaningful insight into the behaviour and response of each individual for each training example. A unique way of assigning binary string for classification problems is introduced. The strengths and weaknesses are defined by considering the distance between a point of one class and the mean of the other class. The pair-wise strategy called CPS is introduced in crossover, utilising BSFC. The aim of CPS is to discourage a crossover between the individuals sharing similar weaknesses. A simple logical formula is used to find out the probability of crossover between individuals using the binary string. Experiments indicate that a standard GP method often gets locked and never improves after that because of the lack of genetic material capable of solving all the training cases. The CPS method on the other hand finds a way to come out of this situation since it utilises the individual's response for each training example.

The CPS method has been compared with standard method using a real world diabetes detection data. The results demonstrate that the CPS method shows continuous improvement over a large number of generations while the standard GP method reaches a saturation state after a certain number of generations and hardly shows any improvement after that. Since addition of CPS requires additional computations, it increases the overall computational complexity and the training time of GP. In order to counter that, a variable size population method has been introduced which starts with a higher number of individuals but after certain generations picks the best individuals and removes the rest to decrease the computational complexity. Experiments reveal that this way the computational complexity is reduced considerably at the cost of small decrease in performance.

Feature generation capability of GP has also been tested using diabetes data. GP is a powerful and efficient tool for performing feature selection and transformation at the same time. Diabetes features are evaluated and ranked using various feature selection methods before using in GP. GP is trained many times using sequential forward selection method and different new features are generated. PCA is also used to transform the original diabetes features into new dimensions, resulting into new features. PCA generated features are also used in GP to generate new PCA-GP features. The new features returned by GP are tested by KNN and SVM classifier. Experiments show that the performance of GP generated features from original features is superior compared to the performance achieved using original features and GP generated features from PCA. A comparison of the performance of GP generated features for diabetes detection with other methods present in the literature reveals that GP outperforms all of them. Based on the experimental results, it can be concluded that a classifier using features generated by GP (CPS) provides considerable improvement in classification performance compared with those classifiers using classically developed features.

Chapter 6

Classification Using Genetic Programming for Imbalanced Diabetes Data

6.1 Introduction

A dataset is said to be imbalanced if the number of examples from one class are very few compared to other classes. The class with few examples is called the minority class and the class dominating the dataset is called the majority class. In such a situation the traditional learning algorithms result in high classification accuracy for the majority class and low classification accuracy for the minority class. The reason is that most of the learning algorithms are designed to reduce the overall error rate and the misclassification error for both classes is treated same. The learning algorithm ignores the difference between two types of misclassification errors and results in a suboptimal classifier.

Quite often, especially in medical diagnosis problems, the class with fewer examples is the class we are interested in. The minority class is usually more important in the sense that it represents rare cases and the consequences of misclassifying a sample from minority class could be huge, e.g., in medical diagnosis usually the samples from the minority class represent a positive test for the disease and wrong classification of these samples could be costly.

In the last chapter we presented a GP based method for classification of diabetes data. It was demonstrated through experiments, results and comparison that this method was able to give highly efficient solutions in terms of classification accuracy. One drawback of the proposed method was the low sensitivity, although the overall accuracy was high. The reason behind the low sensitivity was the imbalanced nature of diabetes data. The number of healthy samples in the diabetes dataset were 500 while the diabetes samples were only 268. So the imbalance was roughly about 2:1. This imbalance caused a bias in the proposed training algorithm during evolution, favouring the class with more number of samples. In this chapter different fitness formulae for the training of GP are employed which help GP in handling imbalanced data. In addition, an improved CPS process is also introduced to increase the role of CPS selected individuals in evolution.

The chapter is organised as follows. Different approaches for handling the imbalanced datasets are presented at the start. The approach used in this thesis for enabling GP to handle imbalanced datasets is a two step approach. First step is the design of classification strategy and the next step is modification in fitness function to keep a balance between different classes. The classification strategy and the fitness functions used are discussed in detail. A modification of traditional CPS method to enhance its performance is presented next. Finally experimental results and analysis is presented at the end.

6.2 Approaches for Classification with Imbalanced Dataset

Generally, there are two approaches used to handle this problem. The first approach is called the *external approach* and it tackles the problem by re-sampling the original dataset without modifying the learning algorithm. The other approach is called the *the internal approach* and in this approach the parameters of the learning algorithm are modified without changing the original dataset [79]. Some approaches also use a hybrid approach which involves both re-sampling and internal parameter adjustment [80, 81]. Both approaches, their advantages and limitations are discussed in the next sections.

6.2.1 External Approach

In this approach the original dataset is re-sampled, either by over-sampling the minority class or by under-sampling the majority class such that the two classes are in equal proportions in the new dataset. In some cases both under-sampling and over-sampling are used. Since it is a pre-processing technique and the data presented to learning algorithm will be balanced, any kind of learning algorithm can be used in this approach. The new balanced data will make sure that the learning algorithm does not show any favouritism during training.

In under-sampling some of the samples of majority class are removed such that the two classes have equal number of instances. Different variations of under-sampling methods are available in literature but the most simple one is random under-sampling in which the samples to be removed are randomly selected. The main drawback of this method is that it can remove samples, potentially useful for the learning algorithm. Although, there are various techniques available in literature that could be adopted to find out the importance of each sample but that would require some a priori information and also would add a computational overhead.

Over-sampling on the other hand increases the size of the minority class to match the size of the majority class by duplicating some of the samples. The most common method of over-sampling is random over-sampling in which the samples to be duplicated are randomly selected. This method can result in over-fitting since exact copies of original samples are duplicated. More sophisticated methods are also available, e.g., Synthetic Minority Over-sampling Technique in which the new minority examples are created by interpolating several minority class examples [82].

6.2.2 Internal Approach

Internal approach (also called cost sensitive approach) takes into consideration the variable cost of misclassification of samples of different classes. During the learning process the misclassification of a sample from minority class is treated as more costly compared to the misclassification of majority class sample. Cost sensitive approach can be divided into two main types, fixed cost approach and variable cost approach.

In fixed cost approach, a fixed penalty is attached to the misclassification example of each class. Usually the cost of misclassifying a minority class sample is greater than the cost of misclassification of majority class sample and this cost is set at the start and does not change during the learning process. Different cost functions, penalising the sensitivity and specificity at different rate are compared using a rule based classifier in [83]. They demonstrate through experiments that large penalties for sensitivity improve the classification accuracy of the minority class. In another research [84], fixed cost penalties for misclassified samples were compared with overall accuracy for training classifiers for multi-classification problem. Although fixed cost approach is simple but the actual fixed cost or penalty varies from problem to problem and it is difficult to adopt a global fixed cost for all the problems.

In variable cost approach the penalties vary during the learning to achieve an optimum performance for both the classes. In [85], an adaptive weighting fitness function was introduced in GP to give more attention to the classes harder to classify. The results revealed that this adaptive method was able to produce better classification accuracies compared to GP with fixed fitness. Three fitness functions were used in [86] for multi-class classification of transmission control protocol data. The experiments revealed that the fitness function using adaptive weight for the minority class increased the accuracy of the minority class. In [87], different fitness functions were tried on two UCI benchmarking problems with increasing penalty for majority class. The experiments demonstrated that penalising majority class increased the accuracy of the minority class but at the expense of reduction in the accuracy of majority class which results in decrease in overall classification accuracy.

Many researchers have shown through comparisons that for most problem domains, internal approach performs better than external approach [80, 88] and for this reason internal approach has been used in this thesis. In this chapter different fixed cost penalties are introduced for fitness evaluation to increase the classification accuracy of the minority class. The sensitivity and specificity have been used in the fitness function to favour the more important class.

6.3 GP for Classification of Imbalanced Data

In the previous chapter, GP was used for the classification of diabetes data using Fisher criterion as fitness function. In this chapter new fitness functions employing sensitivity and specificity are used. Since the Fisher criterion tries to increase the distance between classes while decreasing the variance, it does not give any information about the sensitivity and specificity. Normally in GP, a threshold based method is used to find the true positives (TP) and true negatives (TN). Threshold based method can be divided into two categories, static threshold based method and non-static threshold based method. Both of these methods are discussed below.

6.3.1 Static Threshold Based Classification

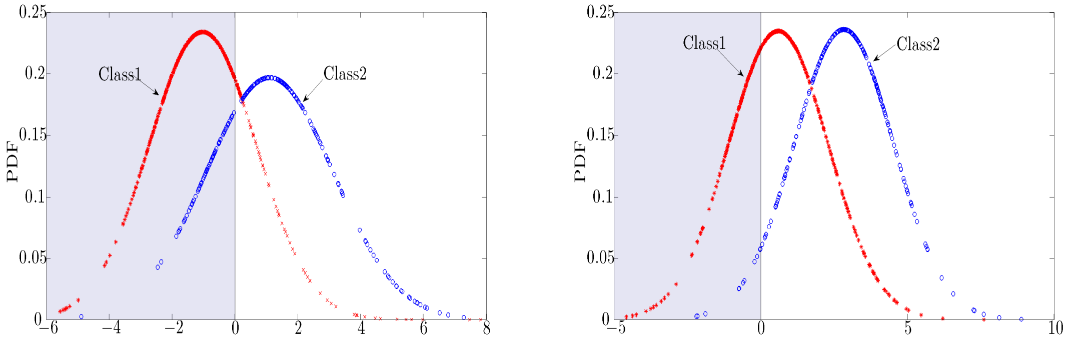
Each GP individual (a mathematical expression) returns a single output for each input instance. This output is then mapped to a set of class labels. For a binary class, the normal way is to define zero as threshold and positive and negative numbers are used as labels for two classes. If the output of the GP is negative it belongs to one class otherwise the opposite class. For multi-class problem different thresholds are used for different classes and each GP individual is trained to return an output to fit between those thresholds for corresponding classes.

Use of threshold puts some serious limitations on evolution of GP. This is because the class boundaries are not linked to the output of GP and remain constant throughout the evolution. In other words GP individual not only have to separate the two classes but also have to make sure that it returns a positive output for one class and negative output for the other class.

This fact is demonstrated using an example. Figure 6.1 shows probability distributions of two GP individuals trained on diabetes data. The horizontal axis represents the output of GP for different instances of training data and height of each distribution shows the relative frequency of output values. The points within shaded area belong to class 1 and points in non shaded area belong to class 2. In Figure 6.1(a) the optimum threshold point is close to zero so choosing zero as threshold is justified and it will result in almost maximum separation of two classes. Figure 6.1(b) shows a different scenario where the optimum threshold point is away from zero. In this case choosing zero as threshold will result in a lot of samples of class 1 classified as class 2 and as a result the final classification accuracy will be poor. Most of the output points will be mapped to wrong class labels which could be mapped correctly by just using a different threshold. The above discussion makes clear that a fixed threshold based decision strategy might be neglecting potentially better performing solutions.

6.3.2 Non-Static Threshold Based Classification

In order to find the best possible GP solution for binary class (in terms of separation), the thresholds need to be shifted for each GP solution, maximising the classification accuracy. Recently many techniques have been proposed to dynamically determine the optimum boundary for each GP solution during evolution. In this research a technique proposed in [89] has been used. In this technique the actual value of probability distribution of underlying classes has been used to find the optimum threshold point.



(a) (b)

Figure 6.1: Probability distributions of two GP individuals

PDF Based Optimum Threshold Detection

In this technique the PDF of each class is used to find the class label of each input instance. The PDF of each class is modelled as Gaussian distribution using the following equation

$$\phi(\mu_c, \sigma_c, x) = \frac{1}{\sigma_c\sqrt{2\pi}} exp\left(\frac{-(x-\mu_c)^2}{2{\sigma_c}^2}\right) \qquad (6.1)$$

where μ_c is the mean and σ_c is the standard deviation of the class c. The variable x is the output of GP individual when evaluated on any input instance. Using the ϕ values of two classes, the samples falling on the wrong side of distribution can be found. The procedure for finding the class labels of input samples during training is given below.

(a) Calculate the output of GP individual for all the training instances for both classes

(b) Calculate the mean and standard deviation of each class from GP individual's output

(c) For each sample calculate the two ϕ values, one for each class using their corresponding mean and standard deviation

(d) The class having lager value of ϕ will be the class of this sample

This way, threshold is the point at which the overlap between two distribution starts. Figure 6.2 shows an example where the point of threshold is found using PDF. The dotted vertical line is the threshold point between two classes. Any

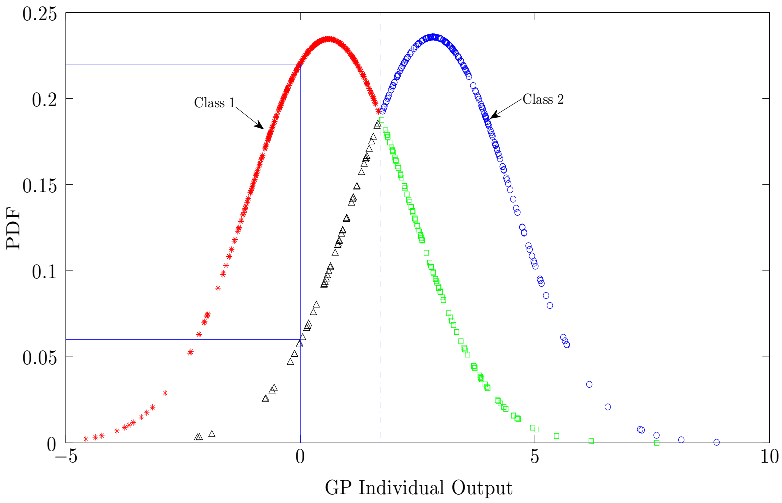


Figure 6.2: Threshold detection using PDF

sample on the left side of this line belongs to class 1 and any sample on the right side of this line belongs to class 2. The squares and triangles are the samples of class 1 and 2 ending on the wrong side of the boundary respectively.

As an example consider a sample having a value of zero (output of GP individual). Looking at the graph it can have two values of ϕ, one for each distribution. Since the value of ϕ for class 1 would be higher for this sample, it will belong to class 1 and it is shown in the Figure. Any sample falling on the left side of threshold line will have larger value of ϕ for the class 1 and will always be labelled as class 1. Similarly any sample on the right side of threshold line will have higher value of ϕ for class 2 and will belong to class 2. An output sample falling on the threshold line will belong to the minority class (the diabetes class in this case).

A similar procedure can be applied to label any unseen sample from the test data. Two value of ϕ can be calculated for the test sample using the mean and standard deviation from the training data. The class with larger ϕ will be the class of the test sample.

This fitness function has the built in ability to handle the imbalanced datasets. Since the value of each ϕ is calculated relative to each mean and standard deviation, all the samples are equally treated irrespective of which class they belong to. In terms of computational complexity it only adds a little to zero threshold technique and is quite fast.

6.3.3 Fitness Functions

The technique discussed above finds the optimum threshold between classes and tells us how to assign class labels to different input samples. Using those class labels, the fitness function of the GP individual can be calculated which tells us how accurately it labels the samples. The true positives (TPs) and true negatives (TNs) are calculated using PDF based threshold detection and are used for the evaluation of different fitness functions. The sensitivity and specificity defined in the previous chapter are also used in evaluation of fitness functions. Five fitness functions are used in this study and they are explained in the coming sections one by one.

Accuracy Fitness

Accuracy fitness (AF) takes the overall classification accuracy as fitness function of GP individual. The formula for AF is same as classification accuracy given in equation (5.5). It is a ratio of total number of samples correctly classified to the total number of samples available in the dataset. Since it treats all misclassified samples equally, irrespective of which class they belongs to, there is a possibility that the majority class may influence the training process and may result in biased solutions.

Figure 6.3(a) shows the contribution of sensitivity and specificity in AF, where sensitivity is the accuracy of the minority class and specificity is the accuracy of the majority class. The Figure has been plotted by using diabetes training dataset as example so the number of majority class samples was 450 and the number of minority class samples was 241. Figure shows that if the specificity is 1 and the sensitivity is 0, the fitness will be 0.65 while in the other case where specificity is 0 and sensitivity is 1, fitness will be only 0.35. It shows that the classifier could be biased towards specificity as it results in high overall fitness.

Equal Weighted Fitness

The equal weighted fitness (EWF) takes average of sensitivity and specificity to favour the individuals performing well on both majority and minority class. The equation for calculating EWF is given below,

$$\text{Equal Weighted Fitness (EWF)} = 0.5 * (Sensitivity + Specificity) \tag{6.2}$$

Figure 6.3(b) shows the graph of EWF. It shows that if either sensitivity or specificity is 0, the maximum weighted fitness an individual can achieve is 0.5. In order to have a high overall weighted fitness value, an individual needs to perform well on both majority and minority class samples.

Weighted Square Fitness

The weighted square fitness (WSF) is average of square of sensitivity and specificity. It is similar to weighted fitness but varies in the rate at which fitness changes with change in sensitivity and specificity.

$$\text{Weighted Square Fitness (WSF)} = 0.5 * (Sensitivity^2 + Specificity^2) \tag{6.3}$$

The variation in WSF with respect to sensitivity and specificity is shown in Figure 6.3(c).

Geometric Mean Based Fitness

Geometric mean based fitness (GF) also gives equal weight to sensitivity and specificity.

$$\text{Geometric Mean Based Fitness (GF)} = \sqrt{Sensitivity * Specificity} \tag{6.4}$$

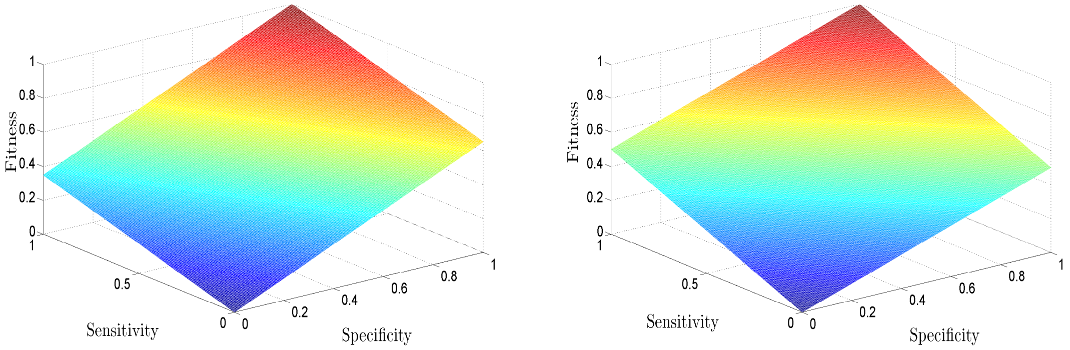
Figure 6.3(d) shows a plot of GF against sensitivity and specificity. One major difference between GF and other fitnesses is that if either sensitivity or specificity is zero, the fitness will be zero. Fitness increases only if there is an increase in both sensitivity and specificity at the same time.

Variable Weighted Fitness

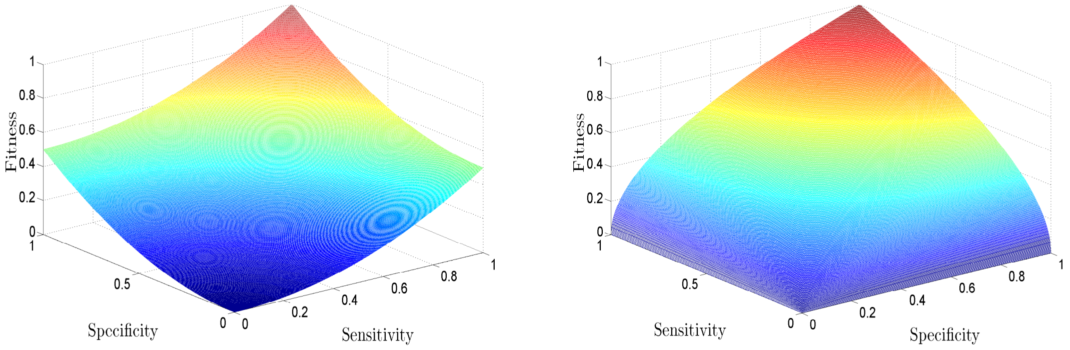
In variable weighted fitness (VWF) the total fitness is weighted average of sensitivity and specificity.

$$VWF = W * Sensitivity + (1 - W) * Specificity \tag{6.5}$$

where W is the weight factor controlling the contribution of any class in the fitness function. The value of W varies between 0 and 1. When $W = 0.5$, the VWF reduces to EWF where the accuracy of both the classes is considered equally important. When $W > 0.5$, the minority class contributes more to the fitness and when $W < 0.5$, the main contribution towards fitness comes from majority class.



(a) AF (b) WF



(c) WSF (d) GF

Figure 6.3: Fitness functions

Choosing the right value of W is a hard task and often varies from problem to problem depending on the class imbalance ratio. In this thesis the value of W is varied from 0.1 to 0.9 with an increment of 0.1 and its effect on the classification accuracy is investigated. The investigation is based on receiver operating characteristics (ROC) explained below.

Receiver Operating Characteristics

Receiver operating characteristics (ROCs) are graphs commonly used in medical diagnosis. Recently ROC graphs have been widely adopted in machine learning because of their ability to give more accurate measure of accuracy for imbalanced datasets. An ROC graph is a two dimensional graph with TP rate along y-axis and FP rate along x-axis.

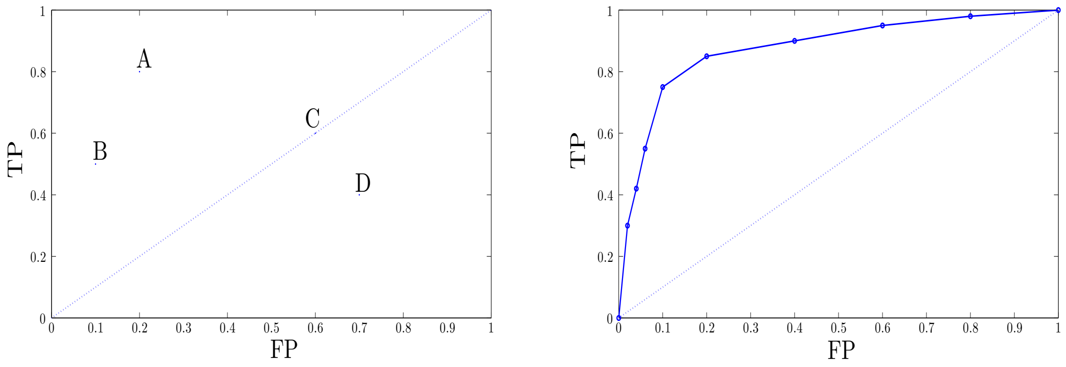
Figure 6.4(a) shows an ROC graph with outputs of four discrete classifiers. There are some important points and regions in this graph that need special attention. The point (0,0) never makes a false positive mistake but never finds a true positive as well. On the other hand the point (1,1) unconditionally labels all the points in the dataset as true positives. The point (0,1) represents perfect classification with 100% TP and 0% FP rate.

The dotted line (y=x) in the middle represents a random guess. For example the classifier C in the Figure is a random classifier which makes a right guess for TP 60% of time but also commits FP mistakes 60% of time. The performance of random classifier always moves on this line and if TP rate increases so does the FP rate and vice versa.

Any classifier in the lower left half is said to be conservative. It labels a sample as TP only with strong evidence making very few FP errors but also has low TP rate. A classifier in the upper right corner is said to be liberal because it labels samples as TP with weak evidence, labelling nearly all the samples as TPs resulting in high FP rate. In the Figure, the classifier B is more conservative than A. Since in many real life problems the TPs are more important than TNs, any classifier working the in upper left half of the graph is more interesting.

A classifier performing below the dotted line is worse than random guessing but can still be useful if negated. With negation its TPs become FNs and FPs become TNs and it can perform in the upper right plane. Such a classifier is said to have useful information but mainly for TNs.

Normally a classifier produces one single point in ROC graph and is called a discrete classifier. For binary classifiers there is a way to get a set of operating



(a) (b)

Figure 6.4: (a) ROC graph showing four classifiers (b) ROC curve showing performance of a classifier at different thresholds

points on the ROC graph and make a curve by varying the threshold. Static threshold technique was discussed in previous section and same technique is normally used for obtaining a set of points. The static threshold can be varied in a way to start from low TP ratio and move to a high TP ratio gradually. A set of operating points will be obtained which can be used to draw a curve. One such curve with a set of ten hypothetical operating points is shown in Figure 6.4(b). Probably a threshold between point 5 (0.1,0.75) and point 6 (0.2,0.85) would be a good compromise between TP and FP rate.

Since we are using PDF based threshold detection which gives us the optimal threshold point between two classes maximising both TPs and TNs, a set of points for plotting ROC cannot be collected using threshold technique. The variation in W for VWF which gives a set of operating points is used in this research to plot the ROC curve. Although this is different to traditional way but gives a good picture for evaluating the optimum value of W.

6.4 Improved CPS with Multiple Offspring

The crossover phenomenon was explained in Chapter 3 in which the two sub trees of parents are swapped with each other. Generally the point of crossover between two parents is chosen randomly. The steps of general crossover process can be written as

1. Randomly pick two parents from the population.
2. Randomly chose a point of crossover on each parent and swap sub trees to create children.

The traditional crossover operation (discussed above) could be constructive or destructive depending on the point of crossover. A simple example is used to describe its destructive effect.

Consider the tree shown in Figure 6.5. Assume that the sub tree with dark grey nodes (4-6) is a good tree. A tree is a good tree if it can contribute to the final optimum solution. There are 15 crossover points available in this tree. If the crossover point is chosen randomly then the probability that this good sub tree will be disrupted is 2/15 or 13.33%. Assume that a GP crossover takes place and it finds a new good sub tree. The new good sub tree consists of the dark grey and light grey nodes (2-9). The probability that this new good sub tree will be disrupted by randomly chosen crossover point is 7/15 or 46.67%.

It is clear from the discussion above that as GP crossover finds new good sub trees by assembling small good sub trees, the chances of the new larger good tree being broken by a crossover increase. This discussion clearly highlights the self destructive nature of crossover. In fact it can damage the optimum solution in more than one ways, e.g., even if the good block survives in next crossover there is a possibility that a new sub tree is added in this crossover which nullifies the effect of good sub tree.

The conclusion of the above discussion is that crossover is both constructive and destructive. Although many strategies have been proposed to control the point of crossover, none has been widely adopted. In [90], a cultural algorithm was used to select the crossover points for crossover. It was reported that the new crossover improved the performance but only for symbolic regression problems. D'haeseleer [91] proposed strong context preserving crossover (SCPC) which only permitted the crossover between two parents at nodes which were at the same position. He reported modest improvements with SCPC. Brood recombination was proposed by Altenberg [92] in which fitter individuals are allowed to produce more offspring. Part of this approach has been used in this thesis.

The concept of CPS for classification problems was introduced in the last chapter and it has been used here again. In this chapter the individuals qualified for crossover through CPS process are allowed to produce more offspring and the best out of those offspring are selected. This is similar to crossover in nature because in any natural crossover, some parents produce more offspring than others. Some of the offspring survive and some cannot. The full process can be described as

1. Randomly pick two parents from the population.

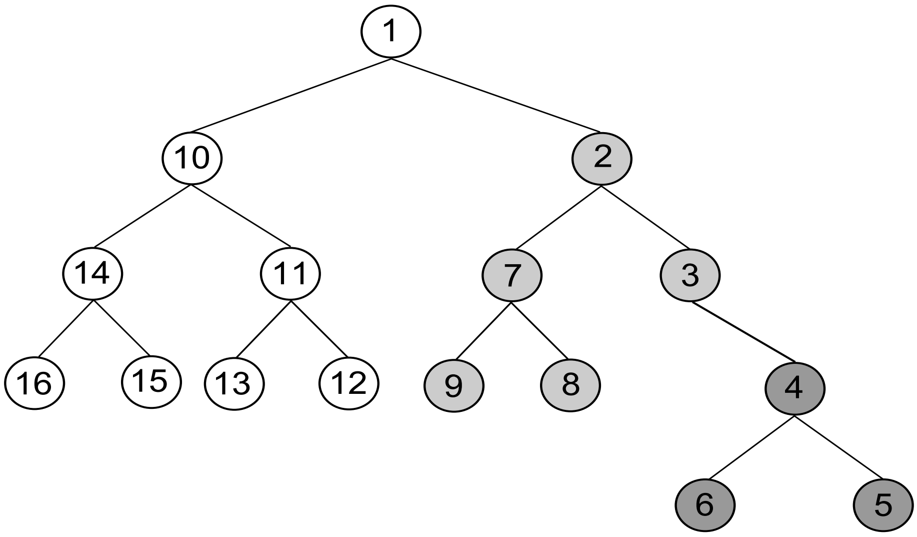


Figure 6.5: A GP tree

2. Check that the two parents qualify for CPS criteria (explained in previous chapter).
3. If they do not qualify just choose the crossover point randomly using standard process and create two offspring.
4. If they qualify for the CPS process, randomly choose N (10 in this study) crossover points on parent trees (5 on each tree) such that each crossover point is different to others, each time creating two different offspring.
5. Evaluate fitness of all offspring, sort them by fitness and select the best two.
6. The best two are included in the new generation while all other offspring are discarded.

Although the destructive crossover may still destroy the good tree but by this strategy the two parents expected to find optimum solution (found using CPS) are given a fair chance to produce many offspring, each with a different crossover point.

6.5 Experiments and Results

All the experiments, results and analysis is presented in this section. The section is divided into two subsections. In first subsection a comparison between CPS and Improved CPS is presented showing the superiority of the later. In the second subsection a comparison between different evolved GP solutions, employing

Table 6.1: Parameters used for the experimental work

Parameter	**Standard Value**
Number of Generation	200 (Exp-C)
Population Size	100-25 (Exp-C)
Function Pool	(plus, minus, times, mydivide, mysquare, mysqrt, reciprocal, sin, cos, myasin, myacos, tan, tanh, negator, abs, mylog)*
Terminal Pool	8 attributes
Genetic Operator	Crossover, Mutation
Operator Probability	0.6, 0.4
Tree Generation	Ramped half-and-half
Initial Maximum Depth	6
Maximum Depth	28
Selection Operator	Roulette
Elitism	Half-elitism

* All the functions starting with ”my” are protected functions.

different fitness function is presented in terms of majority, minority and overall classification accuracy.

All the experiments are performed using the parameters shown in Table 6.1. The Exp-C (explained in previous chapter) has been used for all the training. Exp-C evolves to 200 generations, starting with 100 individuals but later reducing individuals to 25 at 50th generation by picking the best 25 individuals out of 100. The dataset1 of the diabetes dataset (the first of the 10 subsets of the diabetes dataset explained in previous chapter) has been used for all the experiments. For all the experiments the TPs and TNs for fitness evaluation are found using PDF based threshold scheme.

6.5.1 Comparison Between CPS and Improved CPS

The experimental result in this section give a comparison between CPS and Improved CPS, used for improving the performance of genetic operator crossover. The comparison is based on the AF, achieved by both variations during training. Since GP is a stochastic algorithm, the training experiments for both variations were repeated 50 times. The results are summarised in Table 6.2. For each run of the experiment (50 in total) one best tree was produced. The first row of the

Table shows that which genetic operator was used to produce the best solution. For CPS out of 50, 19 best solutions were produced by CPS compared to 25 and 6 by crossover and mutation. Since CPS is supposed to perform better than simple crossover it should be dominant in producing final solution in order to have better average accuracy.

In improved CPS the number of best solutions produced by CPS increase to 31, followed by crossover and mutation with 12 and 7 best solutions. This increase is due to the fact that CPS has been given the relaxation of producing ten offspring in every crossover and best two out of those ten are selected. The second row of the Table shows the AF achieved for solutions produced by mutation, crossover and CPS. For CPS the solutions produced by mutation have the lowest accuracy (75.7%), followed by crossover (76.7%) and CPS (77.1%). The trend in improved CPS is same with mutation having the lowest accuracy (75.6%), followed by crossover (76.8%) and CPS (78.1%).

The accuracy for CPS in improved CPS algorithm has increased by 1% due to the fact that multiple swapping points are tested during crossover producing multiple individuals and the individuals with the best performance are selected. The last row of the Table shows the overall classification accuracy for all the 50 individuals. The overall classification accuracy of improved CPS is comparatively better than CPS for the reason that there were more best individuals produced by CPS criteria in improved CPS than CPS (31 compared to 19), increasing the overall classification accuracy.

The above discussion shows that the improved CPS increases the chances of finding the best individual by giving a chance to CPS selected individuals to produce more offspring. The results show that in improved CPS, the CPS generated individuals dominate the set of best individuals resulting in high overall accuracy. For these reasons the rest of all the experiments are conducted using improved CPS training algorithm.

6.5.2 Comparison of Fitness Functions

This section focuses on finding the statistical differences in terms of sensitivity, specificity and accuracy between different GP fitness functions. Table 6.3 gives the sensitivity, specificity and accuracy values for different fitness functions averaged over 50 runs. The performance is given for both training and test data sets where PDF based threshold technique is used for finding the TPs and TNs.

As expected, the AF fitness function performs poorly for minority class (the

Table 6.2: Comparison of CPS vs Improved CPS using AF for 50 runs

Measure	CPS			Improved CPS		
	Mutation	Crossover	CPS	Mutation	Crossover	CPS
Best solution	6	25	19	7	12	31
AF (%)	75.7 ± 2.8	76.7 ± 3.0	77.1 ± 2.3	75.6 ± 2.7	76.8 ± 2.8	78.1 ± 2.4
Overall AF (%)	76.7			77.4		

diabetes class), resulting in low sensitivity and high specificity. Although the overall accuracy for AF fitness is higher than all other fitnesses, the low sensitivity makes it trivial. In contrast EWF has low overall classification accuracy but provides a balance between sensitivity and specificity. Since the specificity represents the majority class, any change in specificity is more influential in changing overall accuracy and for this reason for EWF, despite an increase in sensitivity, the overall accuracy has decreased due to decrease in specificity. The WSF fitness showed the poorest performance and has the lowest accuracy. Its specificity is similar to EWF for the training data but for the test data both sensitivity and specificity are low resulting in a poor classification accuracy. The results of GF are almost same as EWF with slight variations and it also gives a good balance between sensitivity and specificity.

The results of the Variable VWF are summarised in Table 6.4. The weight in this fitness function controls the contribution of sensitivity and specificity in fitness evaluation. The value of weight has been varied from 0.1 to 0.9 and results in terms of sensitivity, specificity and accuracy are presented in Table 6.4. The results demonstrate that a good balance between sensitivity and specificity is achieved at $W = 0.5$ (same as EWF). None of the other weights provide a better balance between two quantities. As expected the specificity is high for $W < 0.5$ and sensitivity is high for $W > 0.5$. The overall accuracy is maximum at $W = 0.3$ where majority class is favoured. Based on these results it can be said that a weight slightly favouring the majority class results in best overall classification accuracy. At more extreme weights the accuracy decreases which is not surprising as it results in highly biased classifiers.

In order to better visualize the effect of weight on performance an ROC is drawn using different values of weight in Figure 6.6. Figure also shows that an equal weight gives a good balance between sensitivity and specificity and at any

Table 6.3: Performance comparison of different fitness functions averaged over 50 runs

Training Results			
Fitness	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**
AF	51.0 ± 8.4	91.5 ± 3.13	77.4 ± 2.6
EWF	70.5 ± 5.7	74.1 ± 4.10	72.8 ± 1.5
WSF	63.1 ± 32.3	73.2 ± 2.12	69.7 ± 7.4
GF	72.2 ± 6.6	74.4 ± 5.15	73.6 ± 1.9
Test Results			
AF	46.3 ± 13.3	91.2 ± 2.3	75.5 ± 3.6
EWF	71.6 ± 8.23	75.4 ± 6.7	74.2 ± 2.9
WSF	60.4 ± 30.1	68.6 ± 22.6	65.7 ± 7.6
GF	71.5 ± 7.82	74.4 ± 4.7	73.4 ± 2.6

value above or below this weight, a compromise for one of the quantities need to be made. For example at $W = 0.6$ an increase in sensitivity is achieved at the expense of decrease in specificity and opposite is the case on the other side of the weight. As a general rule it can be said that an equal weight provides the best balance in all the quantities.

6.6 Summary

This chapter develops a GP based approach for handling classes with imbalanced datasets with particular emphasis on classification technique and fitness function. The classification technique translates the output of GP individual to a label referring to a particular class. The fitness function quantifies the performance of GP individual by comparing the predicted class label to actual class label. Moreover, a technique is proposed which tries to increase the role of traditional CPS strategy, introduced in previous chapter. The imbalanced diabetes data has been used for testing all the techniques.

The CPS strategy enhances the performance of traditional crossover by favouring crossover between individuals having different strengths and weaknesses. The candidates selected for crossover through CPS process are expected to produce offspring, performing better than most of the individuals in the population. In traditional CPS only two individuals are produced through crossover and many other potential offspring are neglected. In an improved CPS algorithm proposed in this study, multiple (ten) offspring are produced by individuals selected by CPS

Table 6.4: Performance of variable weighted fitness (VWF)

Training Results			
W	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**
0.1	9.25 ± 9.7	99.5 ± 0.35	68.0 ± 3.20
0.2	18.6 ± 12.7	97.9 ± 1.70	70.3 ± 3.40
0.3	43.6 ± 7.50	92.8 ± 4.60	75.6 ± 1.40
0.4	62.2 ± 4.20	82.3 ± 2.50	75.3 ± 1.20
0.5	71.1 ± 4.40	74.7 ± 3.80	73.4 ± 1.70
0.6	86.4 ± 6.10	58.2 ± 7.60	68.1 ± 3.90
0.7	93.2 ± 7.00	36.4 ± 13.6	56.2 ± 6.70
0.8	98.5 ± 1.80	15.5 ± 12.0	44.4 ± 7.20
0.9	99.6 ± 0.20	8.40 ± 32.0	40.2 ± 2.10
Test Results			
0.1	6.67 ± 9.53	96.2 ± 3.30	64.8 ± 4.60
0.2	18.5 ± 14.4	97.2 ± 2.30	69.6 ± 4.50
0.3	41.1 ± 8.30	93.6 ± 3.40	75.2 ± 1.80
0.4	61.5 ± 6.30	83.8 ± 4.30	76.0 ± 3.10
0.5	72.6 ± 5.80	71.6 ± 9.00	71.9 ± 5.30
0.6	85.2 ± 6.30	49.8 ± 8.60	62.2 ± 5.50
0.7	93.0 ± 4.40	27.4 ± 17.2	50.4 ± 10.4
0.8	97.4 ± 3.50	10.4 ± 8.70	40.9 ± 4.80
0.9	97.5 ± 1.80	4.60 ± 4.22	37.1 ± 2.80

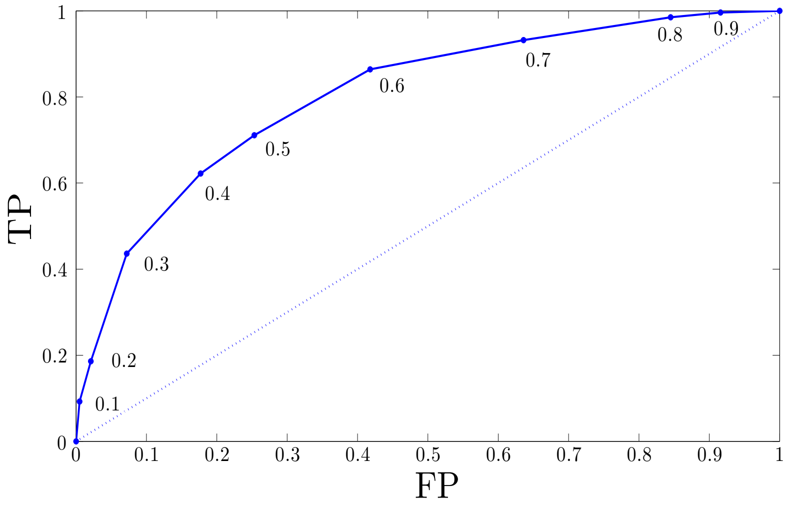


Figure 6.6: ROC curve for different fitness weights

process, each time with a new crossover point and the best two out of these ten individuals are selected. This surely gives a fair chance to CPS selected individuals to search for optimum solution by having multiple crossovers. The experiments on diabetes data demonstrate that the number of best individuals having CPS as origin increase in improved CPS algorithm, resulting in better average accuracy.

A non-static classification strategy is used in this study which is based on the PDF of underlying classes. This strategy gives the optimum threshold point between two classes where the distributions of two classes overlap with each other, maximising the separability of two classes. This strategy performs better than the traditional static classification strategy because of its ability to shift the threshold according to the output of GP individual. This chapter also analyses the performance of different fitness functions in neutralising the influence of majority class in evolution process. The imbalanced diabetes dataset has been used for all the experiments. The experiments for the classification of this binary classification problem reveal that the traditional AF measure fails to do justice to both classes during evolution and results in biased classifiers towards majority class. These classifiers show high accuracy for the majority class and poor accuracy for the minority class.

The EWF and GF use average classification accuracy of different classes as fitness measure. Both of these fitness functions show a balanced accuracy for both the classes without showing any bias towards any class. The only drawback is the low overall accuracy because of the lower accuracy of the majority class. The WSF performs worst among all fitness functions and shows poor performance for both the classes. A VWF has been introduced which varies the contribution of the majority and minority class in fitness function by changing weights. The results evaluated using ROC curve showed that an equal weight gives the best compromise for sensitivity, specificity and overall accuracy. Any weight on either side of equal weight results in biased classifiers despite having high overall accuracy.

In a nutshell the experiments in this chapter show that the choice of fitness function is an important factor for evolving unbiased classifiers showing good accuracy for all the classes. A poor choice of fitness function can result in highly biased classifiers, despite having good overall accuracy.

Chapter 7

Multi-Class Classification Using Genetic Programming

The application of GP for binary classification problem (detection of diabetes) was presented in the previous chapters. In this chapter the application of GP for multi-class classification is discussed. An efficient method for extending the role of GP to multi-class classification is presented. KNN, a simple machine learning classifier has been used within GP for fitness evaluation, enabling GP to handle multiple modulations. Automatic modulation classification (AMC), briefly discussed in Chapter 4 has been used as multi-class problem. Four modulation types have been used as underlying classes.

The chapter is organised as follows: history of GP as a classifier and extensions within GP to make it suitable for multi-class classification are discussed initially. Automatic modulation classification, a multi-class problem, used as case study is discussed in detail. The feature set used for this study and extraction of those features is also discussed. The proposed system and its different steps are presented. At the end, experimental results and comparison with other methods showing the superiority of the proposed method is presented.

7.1 Multi-Class Classification

The study of classification methods for separating more than two classes is called multi-class pattern recognition. It has a variety of applications, such as speech recognition [93, 94], text recognition [95, 96], network intrusion [97], etc. Most of the classifiers are originally built for binary classification problems, e.g., SVM, Perceptron, etc. Generally, an optimised decision boundary is made between any two classes to differentiate them.

One popular approach for extending the role of any binary classifier from binary to multi-class classification is to use multiple binary classifiers. The final decision about the class of a test sample is made by combining the results of all the binary classifiers. A number of approaches are present in the literature to decompose a multi-class classification problem into multiple binary class problems. Some familiar approaches are, decision making by voting, decision making by confidence value and decision making by error correction.

Second approach is to modify the original binary classification algorithm to support the multi-class classification. Some of the algorithms can naturally be extended to multi-class case, e.g., KNN, neural networks, decision trees, etc., while some other methods need major changes to be able to handle multiple modulations. The second approach has been used in this thesis to deal with multi-class problems and for this reason only this approach is discussed in detail.

In this study GP in combination with KNN has been used for multi-class classification. The role of KNN is to help GP in handling multiple modulations and details are provided in the upcoming sections. Multi-class classification using GP is not new and there have been numerous attempts previously to use GP for multi-class classification. Kishore et al. explored the feasibility of GP for multi-class problems for the first time [35]. A GP classifier expression (GPCE) was evolved as a discriminating function for each class. A strength of association (SA) measure was associated with each GPCE representing its effectiveness. Various issues that arise during GP based classification such as creation of training dataset, the role of incremental learning, the functions to be used in function set, conflict resolution for uniquely assigning a class were also discussed. GP was used to develop a decision support system for vehicle dispatching in [98]. A population of utility functions that evaluate vehicles for service requests was evolved.

In [99], GP was used for data preprocessing, to decide automatically whether to do feature selection or feature generation. The method was computationally expensive. In another research Kotani et al. [100] used GP with KNN for feature generation and concluded that GP is an effective tool for feature generation. Few more approaches to use GP as a classifier are discussed in Chapter 3 and interested readers can consult Section 3.5.2.

In this chapter, GP is applied for the classification of four digital modulations, a multi-class pattern recognition problem. Cumulants as features are extracted from received signal and used as input features for training GP. The evolution process of GP is guided by KNN which evaluates the fitness function of GP generated features. The final solution produced by GP is tested using KNN classifier.

The experimental results are evaluated, exploring strengths and weaknesses of the proposed method against other popular methods.

7.2 Automatic Modulation Classification

7.2.1 Signal Model

A general expression for received baseband waveform can be written as,

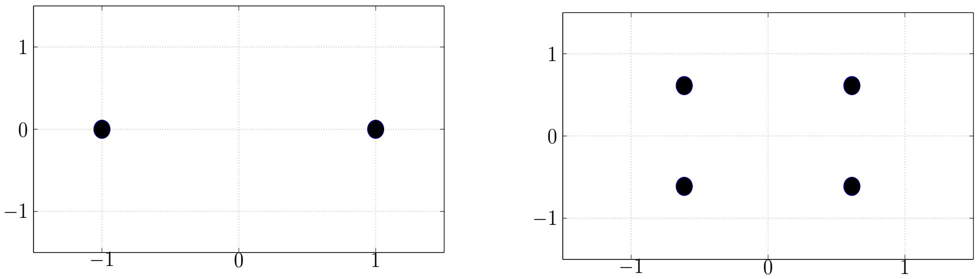
$$y(n) = Ae^{j(2\pi f_o nT+\theta_n)} \sum_{l=-\infty}^{\infty} x(l)h(nT - lT + \epsilon_T T) + g(n) \tag{7.1}$$

where A is the unknown amplitude factor, f_o represents carrier frequency offset, T is the symbol spacing, $h(.)$ represents the residual baseband channel effects, θ_n is the phase jitter which may vary from sample to sample, $x(l)$ is the symbol sequence, ϵ_T is timing error, and $g(n)$ is additive white Gaussian noise (AWGN).

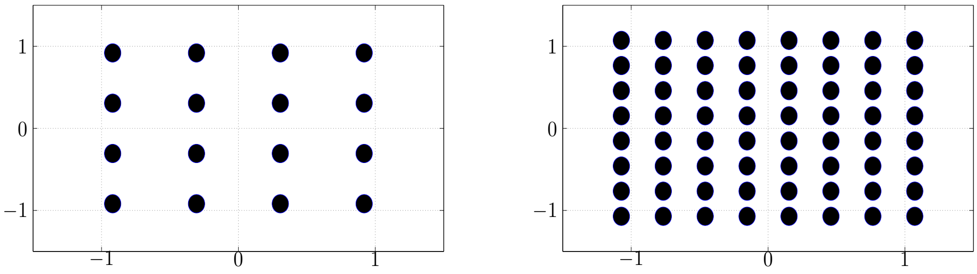
The working environment is assumed to be synchronous and coherent which implies that carrier frequency, data rate and channel coefficients are already known. In practice these parameters are not always known, e.g., these parameters need to be estimated in a furtive environment where signal is intercepted stealthily. However, in some civilian applications these parameters are readily available and one example is the multi carrier (MC) - code division multiple access (CDMA). In MC-CDMA the carrier frequency and data rate are assumed to be known since the modulation type does not depend on these factors [58, 101].

If the carrier frequency is unknown, the power spectral density (PSD) of the received signal can be used to get an estimate of the carrier frequency and baseband in-phase (I) and quadrature (Q) parts of the signal can be recovered using quadrature filter. Various high resolution spectral analysis methods can be used to refine the carrier frequency [102, 103]. A standard fractional sampling scheme can be used to estimate symbol timing and the symbol rate can be obtained using tracking loop [104].

Another assumption made is that the channel effects have been equalized and the residual channel effect $h(.)$ is negligible. Constellation independent algorithms are available to equalize the effects of the pulse shaping filter and the linear channel. Godard's algorithm [105] and the constant modulus algorithm (CMA) [106, 107] can be used for this purpose. The parameters, such as T (the symbol timing), f_o (the carrier frequency offset), ϵ_T (timing error) and θ_n (phase jitter) are assumed to be known. All the above assumptions lead to a simpler model



(a) BPSK (b) QPSK



(c) 16QAM (d) 64QAM

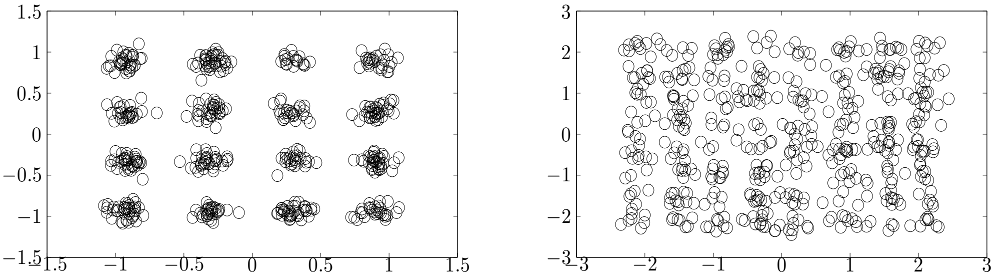
Figure 7.1: The constellation diagrams of the modulations considered in this study

which can be written as,

$$y(n) = Ax(n)e^{(-j2\pi f_o)} + g(n) \tag{7.2}$$

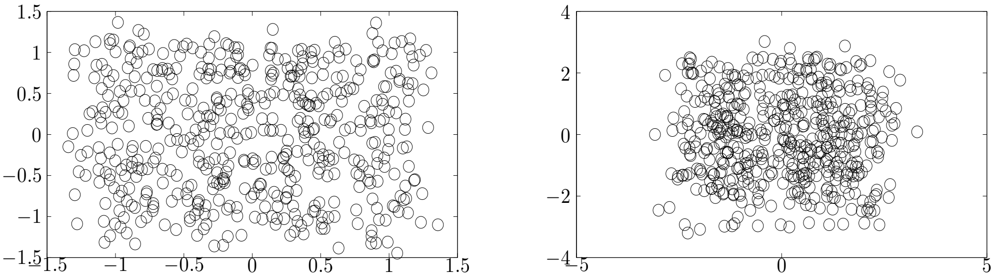
where $y(n)$ is the complex envelop of the received samples. The real and imaginary parts of $x(n)$ are the I and Q components of the received signal. The I and Q parts of the signal can be plotted in a two dimensional scatter plot to get the constellation diagram of the received signal. Each modulation type has a unique constellation diagram. Constellation diagrams for different modulation types used in this study are given in Figure 7.1.

Figure 7.1 shows the signal constellation diagrams under ideal scenario without any Gaussian noise (SNR = ∞), which is not the case in real life. Figure 7.2 and 7.3 show the constellation diagrams for 16QAM and 64QAM under AWGN channel with SNR = 20 dB and SNR = 10 dB respectively. One can see that with decrease in SNR the constellation diagram is distorted, making it difficult to differentiate between different types of modulations.



(a) 16QAM (b) 64QAM

Figure 7.2: 16QAM and 64QAM constellations at an SNR of 20 dB



(a) 16QAM (b) 64QAM

Figure 7.3: 16QAM and 64QAM constellations at an SNR of 10 dB

7.3 Problem Statement

Given N samples of the received signal $y(n)$, with m possible modulation types, M_j, such that

$$M_j = x_{j1}, x_{j2}, ..., x_{jP_j}, \quad where \quad j = 1, 2, ..., m \tag{7.3}$$

where x_{jk} is a point in the constellation and P_j are the total points in that constellation. The received signal $y(n)$ and constellation M_j has been normalised to unit power. The task is to find out the underlying constellation in the received noisy signal $y(n)$, without having any information about the original signal sent by the sender.

7.4 Background

There are two types of classification approaches that are generally used for automatic modulation classification (AMC), likelihood based (LB) approach [108–122] and feature based (FB) approach [52, 123–159]. The former is based upon the likelihood function of the received signal and a decision about the modulation of the received signal is made by comparing the likelihood function of the received signal against a threshold. This solution is optimum in the Bayesian sense as it minimizes the probability of false classification. This optimal solution suffers from computational complexity problem. In FB approach, several features are extracted from the received signal and a decision is made according to their observed values. FB method may not be optimal, but it is usually simple to implement with performance near to optimal if designed properly. Once the modulation type of an incoming signal is identified, it is easy to do signal demodulation and information extraction. A complete survey on different approaches employed for AMC is given in [160].

7.4.1 Likelihood Based Approach

In this approach, AMC is considered as a hypothesis testing problem. The principle of this method is based upon the probability density function (PDF) of the incoming signal. The idea behind this method is that the PDF of the observed signal contains all the information needed for signal classification. It treats AMC as a hypothesis testing problem and compares the likelihood function of an incoming signal with a threshold value. The fundamental assumption made in this approach is that the PDF of the incoming signal is already known. If the actual PDF of unknown parameters is identical to the assumed PDF, the LB method gives optimal solution in the sense that it minimizes the probability of false classification.

This method is further divided into many sub categories depending upon the model chosen for the unknown quantities, Average likelihood ratio test (ALRT), Generalised likelihood ratio test (GLRT), Hybrid likelihood ratio test (HLRT), Quasi ALRT (QALRT) and Quasi HLRT (QHLRT). ALRT treats unknown parameters as random variables with certain PDFs, GLRT treats unknown parameters as unknown deterministic, and the HLRT is a combination of other two approaches. LB approach makes a lot of assumptions and the model itself is computationally complex, and for this reason most of the focus in recent literature has been on FB approach and the same approach is considered in this thesis. Since we are using FB approach in this study, details about LB approach are skipped and interested readers are encouraged to consult [160] for more details.

7.4.2 Feature Based Approach

This approach can be divided in two stages: at the first stage some features are extracted from the received signal for data representation and at the second stage a decision is made according to the values of the features extracted at the first stage. Examples of features that could be extracted are the variance of centred normalized signal amplitude, phase and frequency [52], the variance of zero crossing [127, 128], the variance of magnitude of signal wavelet transform [129], the phase PDF [131, 132] and its statistical moments [133–135], cumulants [139–142], etc. Some other methods involving fuzzy logic [143] and recovery of the constellation shape [144] have also been used for AMC. At the second stage numerous methods could be adopted for decision making. For example Euclidian distance [139–142], Hellinger distance [145], and clustering methods [146] have been used for decision making. Since FB method is adopted in this research, a literature survey of FB method is given below.

Literature Review

Dobre et al. [123] proposed a new set of four features exploiting distinctive characteristics of the signals. The features were the correlation between in-phase and quadrature components, the power of the quadrature component, the average of the signal and the sixth order cyclic moments. The modulations used were amplitude modulation (AM), the suppressed carrier double-sideband (DSB), upper sideband (USB), lower sideband (LSB), binary phase shift keying (BPSK), QPSK, 8-PSK, 16QAM and 64QAM.

In [124], Wong and Nandi used a combination of statistical and spectral features for classification of 2ASK, 4ASK, BPSK, QPSK, 2FSK, 4FSK, 16QAM, V29, V32 and 64QAM. GA was used for selecting optimal feature subset. ANN with multi-layer perceptron (MLP) was trained using the optimal subset of features. Two types of training algorithms namely, back propagation with momentum and adaptive learning rate, and resilient back propagation were used. A separate validation dataset was used to generalise the ANN classifier. The over all classification accuracy achieved was promising at the expense of increased computational complexity. Nandi and Azzouz [126] used two different approaches for classification of a large set of modulations. The first approach was Likelihood based while the second approached involved ANN.

Hong and Ho [129] introduced wavelet transforms for the classification of PSK, FSK and QAM signals. The wavelet transform was used to extract the transient characteristics of modulated signal under AWGN channel and the distinct pattern of each wavelet transform was used for simple identification of modulations. Swami and Sadler [130] proposed a hierarchical scheme employing fourth order cumulants for classification of a large set of modulation types. The method presented was robust in the presence of carrier phase and frequency offsets. The drawbacks were, the assumed noiseless conditions and the high number of data samples (more than 10,000) for more difficult QAM modulations.

Yang and Soliman [131] presented a method based on the phase density function of the incoming signals. The Tikhonov phase density was assumed to be the density of the received signal. The signals used were continuous wave (CW), BPSK, QPSK and 8PSK. Yang and Liu [132] improved the method presented in [131] by deriving an asymptotic classifier using the exact phase density function of the incoming signal. The performance of the algorithm was demonstrated through an example and it was shown to be more effective compared to the results achieved in [131]. In another research [134] Soliman and Hsue used statistical moments of the signal phase to classify MPSK modulation types. They developed a decision rule based on a hypothesis test, deriving an analytic expression for the probability of misclassification.

Schreyogg and Reichert [137] used Discrete Fourier Transform (DFT) of the phase histogram and knowledge about the distribution of the magnitude to classify QAM signals. Swami, Barbarossa and Sadler [138] proposed a method based on constant modulus algorithm (CMA) for classification of QAM signals in a frequency selective fading channel. CMA-initialised alphabet matched algorithm was used for equalisation purpose to remove inter-symbol interference while a low-complexity adaptive classifier based on fourth order cumulants was used for classification. In [139], Spooner proposed the use of sixth-order cyclic cumulants for signals corrupted by co-channel interference and AWGN. It was shown through experiments and simulations that use of sixth order cyclic cumulants provides benefits in certain situations.

Dobre, Ness and Su [140] used the fourth, sixth and eighth-order cyclic cumulants for classification of ASK, PSK and QAM signals. In another research [141], the same authors performed classification of QPSK, 16QAM, 64QAM and 256QAM using cyclic cumulants. Classification of QAM signals was performed in a fading environment by Dobre et al. [142] using cyclic cumulants. The method presented was robust to the presence of phase offset and timing error. Wei and Mendel [143] proposed a fuzzy logic based classification method for non-ideal environments. The experiments showed that the fuzzy method performs consistently better than other methods with increased computational complexity. Mobasseri [144] used constellation shape for digital modulation classification in the presence of AWGN. The modulations used were V.29, 8PSK and 16QAM. The experiments showed that constellation shape exhibits considerable stability in noisy and weakly synchronous environments. In [145], Huo and Donoho developed a simple fast and robust method based on counting. The method provided high classification accuracy for high SNRs but the performance decreased sharply below 10 dB SNR.

Swami and Sadler [146] used clustering technique based on the constellation shape of the received signal. The technique worked very well for high SNRs but at low SNRs all the constellation points reduced to a single cluster and it was difficult to differentiate between different constellations. In [147], Taira and Murkami proposed to use the statistical parameters of the signal for classification of analogue modulated signals. The compactness of instantaneous frequency distribution was used to differentiate between frequency modulated signals and the results were promising over 10 dB SNR. A neural network based hierarchical approach was used in [148]. It was demonstrated that hierarchical neural network was more suited for AMC instead of single, large and fully connected network. Experiments employing ten modulations were conducted showing the superiority of the proposed method compared to other conventional methods.

Yoshioka et al. [149] proposed a method based on symbol selection in Rayleigh fading channels. Instantaneous amplitude and frequency were used as features with nearest neighbour rule used as final classifier. Chung and Polydoros [150] used an envelope based scheme for classifying FSK signals in AWGN channel. The sum of squared envelopes along with pre-defined threshold values was used for classification. In [151], Zhijin and Junna used Winger-Ville distribution (WVD) for identification of FSK, PSK, FM and SSB signal. The WVD was used to extract the frequency contents of the signal and the results verified that the method was robust and effective. Yu, Shi and Su [154] used properties of amplitude spectrum for classification of MFSK signals. They used a classifier based on Fast Fourier Transform (FFT) and reported that the classifier works well for SNR $\geq$ 0 dB. Table 7.1 gives detailed information about different FB methods [160]. Beside these there are many other FB methods present in literature and interested readers can consult [160].

Table 7.1: Summary of feature based methods

Author(s)	**Feature(s)**	**Modulations**	**Unknown Parameters**	**Channel**	**Comments**
Azzouz and Nandi [52]	Power spectral density of amplitude, standard deviation of amplitude, phase and frequency	BPSK, QPSK, 2FSK, 4FSK, 2ASK, 4ASK	———	AWGN	A wide range of analogue and digital modulation classification discussed.
Hsue and Soliman [127,128]	Variance of zero crossing interval sequence, phase difference and zero crossing interval histograms	BPSK, QPSK, 8PSK, 2FSK, 4FSK, 8FSK	———	AWGN	Simple and efficient, works only at high SNRs.
Hong and Ho [129]	Variance of wavelet magnitude	QPSK, 16QAM, 4FSK	———	AWGN	The modulations considered were easy to classify.
Swami and Sadler [130]	Fourth order cumulants	BPSK, QPSK, 8PSK, 16QAM, 64QAM, V29, V29c, V32, 4ASK	Phase, frequency and timing offset	AWGN, Co-channel interference	Noiseless conditions assumed for difficult modulations, large number of samples used.

Spooner [139]	Sixth order cyclic cumulants	QPSK, MPSK, 8QAM, 16QAM, 64QAM, 256QAM, V29	Frequency offset	AWGN, co-channel interference	Large number of samples were used.
Dobre et al. [140]	Eighth order cyclic cumulants	BPSK, QPSK, 8PSK, 4ASK, 8ASK, 16QAM, 64QAM, 256QAM	————	AWGN	It gives very good results for high SNRs but assumes all the parameters already known.
Dobre et al. [141]	Fourth, sixth and eighth order cyclic cumulants	QPSK, 16QAM	Phase jitter, frequency offset	AWGN, impulsive noise	Robust to frequency and phase offsets but limited to QPSK, 16QAM.
Dobre et al. [142]	Eighth order cyclic cumulants	4ASK, 8ASK, BPSK, QPSK, 16QAM, 32QAM, 64QAM	————	Rayleigh and Ricean fading channels	Large number of samples used.
Huo and Donho [145]	Counting method	QPSK, 6PSK	————	AWGN	Very easy, efficient and robust method, limited to two modulation schemes.
Yu et al. [154]	DFT	2FSK, 4FSK, 8FSK, 16FSK, 32FSK	————	AWGN	Only MFSK signals considered.

Shen et al. [155]	Fourth order cumulants	BPSK, QPSK, 8PSK, $\pi/4$DQPSK	Frequency offset, timing error	AWGN, Co-channel interference	Used only for MPSK, showed good results even in the presence of frequency offset.
Mirarab and Sobhani [156]	Higher order cumulants	BPSK, QPSK, 8PSK, oQPSK, 4ASK, 8QAM, 16QAM, 64QAM, 256QAM	Frequency offset	AWGN	Large set of modulations considered, showed good performance in the presence of frequency offset but only at high SNRs.
Xi and Wu [157]	Higher order statistics	BPSK, QPSK, QPSK, 16QAM, 64QAM	—	AWGN, multipath Rayleigh fading	Showed good results in the presence of multipath fading.
Headley et al. [158]	Cyclic spectrum, neural networks	BPSK, QPSK, MSK, FSK	—	AWGN	Works well for low SNR conditions but suffers from computational complexity.
Hong [159]	ICA	BPSK, QPSK	Signal power	AWGN, fading	Works very good at low SNRs, limited to two modulation types.

7.5 Data Preprocessing and the Feature Set

Signal model for a received baseband signal was presented in Section 7.2.1. The received signal will have one of these modulations: BPSK, QPSK, 16QAM and 64QAM. The proposed system uses constellation diagram of each modulation. The constellation diagram of each modulation is normalised to unit power before the addition of any kind of noise. Data normalisation is important for making the data stationary. The data was normalised according to the following equation to make it of zero mean and unit variance

$$\hat{x}_k(i) = \frac{x_k(i) - \bar{x}_k}{\sigma_{x_k}} \tag{7.4}$$

where i is the sample number, $\bar{x}_k$ and σ_{x_k} are the mean and the standard deviation of the k_{th} feature, respectively.

After normalisation, noise (AWGN in our case) is added to the original constellation, and after that features are extracted from the noise corrupted signal. Higher order statistical moments and cumulants are extracted as features from the noisy signal. The moments and cumulants used in this study were explained in Section 4.2.5. The noise used in this study is AWGN and the value of higher order cumulants is zero for AWGN, so higher order cumulants of the received signal consists of the source signal only. This property makes higher order cumulants a perfect candidate (as features) for differentiating between different modulations. The reason for choosing a particular set of cumulants (given in Section 4.2.5) is the inability of other cumulants to differentiate between the modulations considered in this study. For example, all modulations except BPSK have four fold symmetry, which theoretically means some of the cumulants will have zero value for these modulations, and hence non-discriminating, e.g., the following cumulants have zero value for these modulations.

$$\{C_{11}, C_{30}, C_{21}, C_{12}, C_{03}, C_{31}, C_{13}\} = \{0\} \tag{7.5}$$

The cumulants defined in Section 4.2.5 have been used in this study as features. The values of those cumulants for different SNRs are given in Tables 7.2-7.4. It is clear from the tables that BPSK can easily be distinguished from other modulations using any sixth order cumulant C_{60}, C_{61}, C_{62} or C_{63}. QPSK can also be separated from the other two using C_{63} and combination of some other cumulants when the SNR is above 10 dB. Below 10 dB SNR, the difference between cumulant values of QPSK and 16QAM, 64QAM is marginal, making it difficult to separate them. The situation for discrimination between 16QAM and 64QAM is even worse where there is hardly any difference between cumulants values at 0 dB SNR. At 10 dB, C_{63} can be used to differentiate them but the difference is marginal resulting in poor classification accuracy. It can be concluded that at high SNRs the cumulants can provide high classification accuracy, but as the SNR decreases the margin between the values of cumulants for different modulations becomes small, making classification difficult, especially for 16QAM and 64QAM.

Table 7.2: Cumulants values with standard deviation averaged over 1000 runs at an SNR of 20 dB and with 1024 number of samples

Cumulants	**BPSK**	**QPSK**	**16QAM**	**64QAM**
C_{40}	-1.96 ± 0.01	0.98 ± 0.01	-0.66 ± 0.04	-0.60 ± 0.05
C_{41}	-1.96 ± 0.01	0.00 ± 0.06	0.00 ± 0.05	0.00 ± 0.05
C_{42}	-1.96 ± 0.00	-0.98 ± 0.00	-0.67 ± 0.02	-0.61 ± 0.02
C_{60}	15.48 ± 0.06	-0.00 ± 0.41	-0.01i ± 0.30	-0.00 ± 0.28
C_{61}	15.49 ± 0.06	3.85 ± 0.05	1.99 ± 0.15	1.74 ± 0.17
C_{62}	15.49 ± 0.06	-0.00 ± 0.30	-0.01i ± 0.24	-0.00 ± 0.22
C_{63}	15.49 ± 0.06	3.88 ± 0.02	2.03 ± 0.09	1.75 ± 0.09

Table 7.3: Cumulants values with standard deviation averaged for 1000 runs at an SNR of 10 dB and with 1024 number of samples

Cumulants	**BPSK**	**QPSK**	**16QAM**	**64QAM**
C_{40}	-1.65 ± 0.01	-0.82 ± 0.03	-0.56 ± 0.06	-0.51 ± 0.07
C_{41}	-1.65 ± 0.01	0.00 ± 0.06	-0.00 ± 0.05	0.00 ± 0.052
C_{42}	-1.65 ± 0.01	-0.82 ± 0.01	-0.56 ± 0.02	-0.51 ± 0.02
C_{60}	11.98 ± 0.11	-0.01i ± 0.38	0.01 ± 0.28	-0.01 ± 0.28
C_{61}	11.98 ± 0.11	2.98 ± 0.11	1.55 ± 0.18	1.34 ± 0.19
C_{62}	11.98 ± 0.11	-0.010i ± 0.28	0.01 ± 0.21	-0.01 ± 0.20
C_{63}	11.98 ± 0.11	3.01 ± 0.04	1.57 ± 0.09	1.36 ± 0.10

Table 7.4: Cumulants values with standard deviation averaged for 1000 runs at an SNR of 0 dB and with 1024 number of samples

Cumulants	**BPSK**	**QPSK**	**16QAM**	**64QAM**
C_{40}	-0.50 ± 0.09	-0.25 ± 0.11	-0.16 ± 0.12	-0.15 ± 0.13
C_{41}	-0.50 ± 0.09	-0.00 ± 0.06	0.00 ± 0.06	0.00 ± 0.06
C_{42}	-0.50 ± 0.09	-0.25 ± 0.04	-0.17 ± 0.05	-0.16 ± 0.05
C_{60}	1.98 ± 0.52	$0.01i$ ± 0.47	$0.01i$ ± 0.53	$0.01i$ ± 0.56
C_{61}	1.98 ± 0.52	0.50 ± 0.22	0.25 ± 0.24	0.22 ± 0.25
C_{62}	1.98 ± 0.52	0.01 ± 0.15	0.00 ± 0.15	0.00 ± 0.16
C_{63}	1.98 ± 0.52	0.50 ± 0.12	0.26 ± 0.13	0.23 ± 0.13

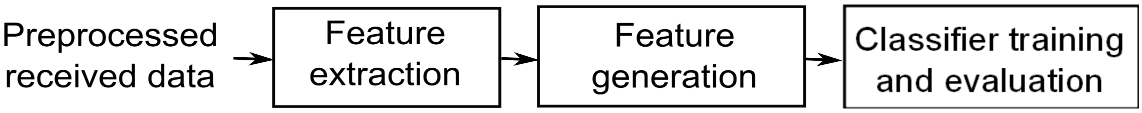

Preprocessed received data
Feature extraction
Feature generation
Classifier training and evaluation

Figure 7.4: Block diagram of the modulation classification process.

7.5.1 Generation of Features

Generation of features is a feature transformation process, where new features are generated by using combinations of the existing features. In this study GP is used for feature generation which makes a single new feature by making combination of the original features.

Different researchers have tried to transform the original features (cumulants in our case) to new, less number of features by manually trying different combinations of the cumulants. Swami and Sadler [130], instead of using cumulants directly, used normalised values of cumulants by dividing higher order cumulants and lower order cumulants, and reported improvement in classification accuracy. This encouraged other researchers to use modified cumulants or combinations of cmulants for classification. Most of the methods in the literature created new or normalised cumulants by combining or dividing existing cumulants. All these methods used statistical distribution of cumulants to decide how to normalise or combine cumulants. The process of selection of cumulants was manual, depending on the intuition of the researchers. In this study, the process of finding the best combination of cumulants has been automated by using GP. GP evaluates different combinations of input cumulants and returns the combination best suited for classification for the given modulations. The full process will be explained shortly.

7.6 The Proposed System

The general process of modulation classification is shown in Figure 7.4. The process starts from the pre processing of the received data, followed by feature extraction. This is followed by an optional process of feature generation. It is optional because sometimes the features extracted from the preprocessed data are directly fed to the classifier. In this research the feature generation process is performed by GP which transforms the original features into new single dimension feature. The generated features are passed to the classifier which is trained using the training data. Finally the trained classifier is evaluated using the test data

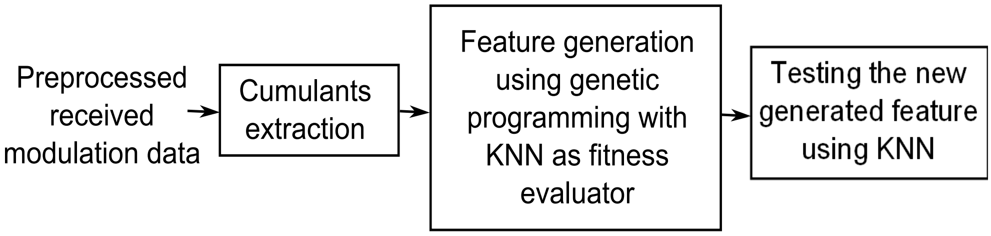

Preprocessed received modulation data
Cumulants extraction
Feature generation using genetic programming with KNN as fitness evaluator
Testing the new generated feature using KNN

Figure 7.5: Block diagram of the proposed system.

and the classification accuracy is calculated.

In this study a variation of GP, aimed at making GP a multi-class classifier, has been proposed. GP has been used in the past for multi-class classification purpose and a detailed analysis was presented in Section 7.1. In this study GP has been used to transform the original cumulant features into a new single feature and KNN has been used for fitness evaluation of the new feature generated by GP. The use of KNN in the feature evaluation process helps GP in multi-class classification task. As the GP evolves, different combinations of the input features (moments and cumulants) are tested by GP as possible candidate solutions for separating the given modulations. In the past literature, the existing features have been used for classification but a combination of those features should surely return better performance than individual features. GP automates this complex process of trying different combinations of the existing features in an efficient way. Once GP returns the best individual, it is tested by KNN classifier, used during fitness evaluation.

The proposed system is shown in Figure 7.5. The received modulation data is normalised to unit power (processed) initially before being passed to the next block where cumulants are extracted from the processed data. The cumulants used in this study were explained in Section 4.2.5. The extracted cumulants are fed to GP which uses them as input features. The main task during the evolution of GP is to maximise the separation between underlying classes. The process is guided by KNN which evaluates the fitness function and helps GP in handling multiple modulations. The role of KNN for fitness evaluation and hence making GP suitable for multi class classification is discussed in detail in Section 7.6.1. At the end of evolution GP returns a new feature which is made up of combination of input cumulants. The performance of this new feature in terms of separating different underlying modulations is tested by KNN classifier at the last step.

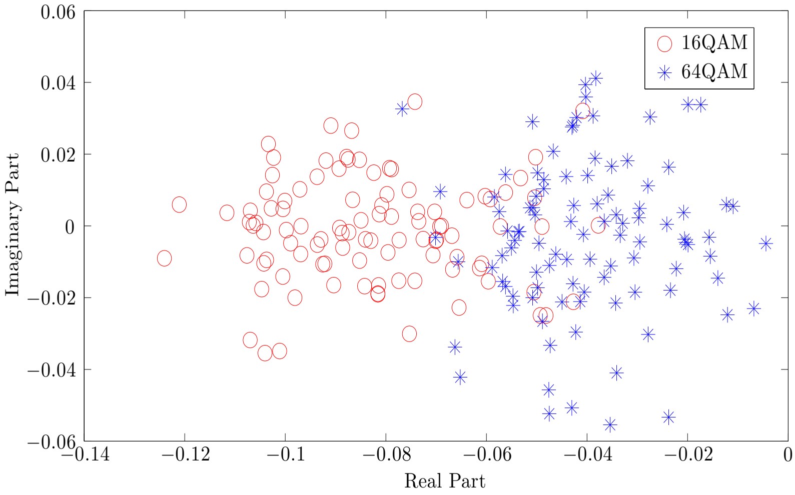


Figure 7.6: Feature distribution for 16QAM and 64QAM at an SNR of 10 dB and 1024 number of samples as returned by a GP-tree using KNN as fitness evaluator.

7.6.1 Fitness Evaluation Using KNN

Fitness function is an attribute of an individual which tells about the strength of an individual in solving the given problem. The common trend in calculating fitness is to have a set of predefined outputs and GP individual (tree) is asked to produce those outputs from given inputs. The inverse of the number of mistakes made by an individual is taken as the fitness of that individual. The higher the fitness, the fitter is the individual. Here, we also take the inverse of the number of mistakes made by an individual as the fitness of that individual but KNN is used for calculating the mistakes made by that individual. Training data is divided into two parts: reference data and evaluation data. The reference data is used by the individual for making reference samples and evaluation data is used to evaluate the individual and assign it a fitness value. The evaluation data is different to the test data later used for testing of the best tree. For each of those evaluation samples, distance from the all reference samples is calculated and the most frequent class among k nearest neighbors is considered to be the class of this sample. An example of reference feature space created for KNN classifier for 16QAM and 64QAM is given in Figure 7.6.

The formula for calculating the distance of test sample from neighbouring samples is given as

$$ {d_{er}}^2 = |f_e - f_r|^2 \tag{7.6} $$

Table 7.5: KNN parameters

Parameters	Values	
	Reference samples	Evaluation samples
Training	300	400
Testing	2000	40000
Value of k	19	19

where f_e is the evaluation sample, f_r is the reference sample, $*$ represents the complex conjugate and d_{er} is the distance between the reference and the evaluation sample. Here, both reference and evaluation samples are the outputs of the tree generated by GP. These are the new feature values obtained from existing cumulants. This way all the evaluations samples are given a class label (predicted). The predicted class label is compared with actual class label to find out the classification mistakes. The inverse of the number of wrong classifications made by an individual is used as fitness value as given below

$$f = \left[\sum_{i=1}^{n} e_i\right]^{-1} \tag{7.7}$$

where f is the fitness function, n is the total number of classes (4 in our case) and e_i represents the number of wrong classifications made by the individual for class i. The individuals having higher fitness values have less number of errors and have better chances of transferring to the next generation. The testing experiments related to GP (KNN) are explained in a later section.

Table 7.5 shows different parameters used for KNN. It shows the number of reference and evaluation samples used for training and testing phase. The value of k was chosen in a similar fashion as done in the previous chapter, where few values of k were tested for a random modulation data and the one giving the maximum performance was chosen. The value of k for all the experiments was chosen as 19.

7.6.2 Complexity of the Proposed Classifier

Many researchers think that GP classifier will take a long time for classification as the time for evolution could be very long but the computational complexity of the final classifier is not to be confused with the training time of the classifier. Once we get the final solution from training a GP, that final solution is used for classification and the computational complexity of training a GP does not come into account while using the final solution. The final solution produced by GP has inputs as cumulants and some functions from the function pool so the complexity of this particular solution really depends on the particular cumulants and functions used by final GP solution. The new GP solution is tested by KNN classifier which has the complexity of $O(nd)$ where n is the number of reference samples and d represents dimensions of the reference data. Here we have used two dimensional data in the form of complex numbers but the function pool contains an *abs* function which returns the magnitude of complex number as output when the input is a complex number. If the final solution is using *abs* function in the last stage the final output could be a real value. In a nutshell, the complexity of our final classifier is $O(nd)$ + complexity of final solution.

7.7 Experiments and Results

7.7.1 Simple KNN Classifier

In this section the results are presented when the cumulants are directly tested by KNN classifier. These results are obtained to test the effectiveness of KNN and input features, without involving GP. This will help us to make a comparison of input features performance with KNN against the new feature's performance (generated by GP with KNN). There are seven features used here for performance evaluation of simple KNN classifier. These seven features are the cumulants defined before in Section 4.2.5. The method is quite simple. Initially some of the feature values are used as reference data for KNN. As there are seven input features so this makes a seven dimensional space for KNN. After this some test values of these features are given to KNN which finds out k neighbouring samples using Euclidean distance. The class having the maximum number of neighbours is considered as the class of test sample. The number of reference samples used for KNN are 200, each modulation type having 50 samples. The number of samples for test are 40,000, each modulation having 10,000 samples. The equation for calculating the distance of test sample from reference samples is given below.

$$d^2 = \sum_{i=1}^{7} |f_i - f_r|^2 \tag{7.8}$$

where f_i is the input feature value and f_r is reference feature value. The maximum value of variable i is 7 which represents seven cumulant features. Table 7.6

summarizes the result of simple KNN classifier. The analysis and comparison of these results is given in the next section.

7.7.2 GP with KNN Classifier

In this section the results for GP generated features, with KNN as fitness evaluator are presented. Some of the best trees (features) produced by GP are included in Appendix A.2. The number of generations used for all the experiments was 100. The number of individuals in each of the experiments was 25. Total number of training experiments done is also 25 so the total number of individuals or solutions created was 625. The best tree out of these 625 trees was tested with test data and results are analysed here. All the training experiments have been done in MATLAB/GPLab [161] environment. Testing experiments are also carried in MATLAB/GPLab environment. The number of samples used are 512, 1024, 2048 and 4096 respectively, and the SNRs used are 5 dB, 10 dB, 15 dB and 20 dB. For each value of SNR and number of samples, 10,000 realization are produced. These 10,000 realizations are tested with the best tree and results are summarized. Table 7.6 shows the results obtained for particular combination of SNRs and number of samples. The results for simple KNN are also shown in the same Table. It is clear from these results that GP (KNN) produces better results compared to simple KNN classifier. The improvement in performance is especially noticeable for small number of samples and low SNRs, e.g., at 512 number of samples and 5 dB SNR, the improvement in performance is about 5% while at 1024 number of samples and same SNR the improvement in performance is 6%. There is less improvement in performance at high SNRs and number of samples where the maximum improvement is about 1%. The improvement provided at low SNRs and small number of samples is important since classification is most difficult under these conditions.

The performance of GP for different SNRs and at 1024 number of samples is given in Table 7.7 in the form of confusion matrix. The classification performance for BPSK and QPSK is 100% in all the cases as shown in matrix. The performance for 16QAM and 64QAM on the other hand is low and decreases further with a decrease in SNR. From the Table it is can be concluded that the main reason for low classification accuracy at low SNRs are 16QAM and 64QAM modulations. One of the reasons of low accuracy for 16QAM and 64QAM is the drift in GP during evolution towards the classes easier to separate. Since the first two modulations are easier to classify, the evolution process pays more attention

Table 7.6: Accuracy of performance (%) using KNN and GP-KNN with 1 standard deviation for different SNRs and number of samples

Method	SNRs	512	1024	2048	4096
KNN	5 dB	79.0	81.0	91.0	96.0
	10 dB	88.0	93.0	99.0	100.0
	15 dB	90.0	97.0	100.0	100.0
	20 dB	93.0	98.0	100.0	100.0
GP (KNN)	5 dB	84.0±4.0	87.0±3.0	92.0±3.0	97.0±2.0
	10 dB	92.0±2.0	95.5±0.7	100.0±0.3	100.0±0.1
	15 dB	96.6±2.0	98.7±0.4	100.0±0.1	100.0±0.0
	20 dB	98.5±1.0	99.7±0.2	100.0±0.0	100.0±0.0

to them as they result in high overall classification accuracy. This ultimately results in low accuracies for the 16QAM and 64QAM modulations. This problem is addressed in the next chapter where the problem is tackled by dividing the training process into two stages. The modulations difficult to separate at the first stage are passed through another GP training stage for further separation.

7.7.3 Comparison with Existing Methods

This section gives a comparison of the proposed method with other methods present in the literature. In this chapter comparison with only one of the research which used exactly same modulations and same channel has been presented. More detailed analysis and comparison with many other methods present in the literature will be presented in the next chapter. In [162], Wong, Ting and Nandi presented results for the same modulations that we have used in this research. They used Naïve Byaes classifier and SVM classifiers with C_{42} and C_{63} as extracted features. Table 7.8 presents the results of Naïve Bayes and SVM classifier along with GP (KNN).

The SNRs used are 5 dB, 10 dB, 15 dB, 20 dB and the number of samples used are 512, 1024, and 2048. The results demonstrate the superiority of GP over other methods for all the SNRs and number of samples. The performance of the Naïve Bayes classifier is worst out of three methods especially at small number of samples and low SNRs. As the SNR increases its performance starts to improve and becomes comparable with the performance of SVM but is always less than

Table 7.7: Confusion matrix for GP-Tree at different SNRs and at 1024 number of samples

SNR	Modulation Type	BPSK	QPSK	16QAM	64QAM
5 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	7655	2865
	64QAM	0	0	2345	7135
10 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	9160	979
	64QAM	0	0	840	9021
15 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	9756	298
	64QAM	0	0	244	9702
20 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	9945	70
	64QAM	0	0	55	9930

Table 7.8: Comparison of GP (KNN) with other methods in terms of classification accuracy (%)

SNR in dB	Naïve Bayes			SVM			GP (KNN)		
	512	1024	2048	512	1024	2048	512	1024	2048
5	82.4	85.9	89.7	83.0	86.0	90.0	84.0	87.0	92.0
10	90.0	93.9	97.7	91.2	94.8	97.6	92.0	95.5	100.0
15	93.5	97.0	99.4	96.4	98.3	99.2	96.6	98.7	100.0
20	94.4	98.1	99.7	98.3	98.9	99.4	98.5	99.7	100.0

GP's performance. The performance of SVM although better than Naïve Bayes but is always less than GP (KNN). The improvement in performance is more evident at low SNRs and number of samples where the difference in performance is 1% or more. As these two parameters increase, the performances become closer, with GP (KNN) always performing better than SVM.

Complexity Comparison

Each of the methods (Naïve Byaes, SVM, GP(KNN)) uses cumulants as features and the cost of extracting cumulants is same for all of the methods which is $O(M)$, where M is the number of operations. For AMC, the complexity which matters is the complexity of the final trained classifier which will be used for testing. The complexity involved during training will not come into equation while doing modulation classification. For this reason the complexity of all the classifiers discussed in this section is the complexity involved during testing.

The computational complexity of Naïve Byaes classifier is $O(Np)$, where N is the number of samples and p is the number of features involved. Only two features (C_{42} and C_{63}) were used in [162], so the total cost of Naïve Byaes classifier is $O(N)$. The computational complexity of SVM classifier during testing is $O(N_{sv}p)$, where N_{sv} is the number of support vectors and p is the number of features involved. For GP (KNN), the calculation of computational complexity is more complicated. Since each GP training run produces a new GP feature which may be completely different each time, the complexity of GP feature can vary a great deal. The complexity of any GP feature depends on the functions used from the function pool and the number of original features as terminal. Depending on the type of functions used, the complexity can vary a lot. The new GP features are tested in this research by KNN classifier which has a complexity of $O(np)$, where n is the number of reference samples and p is the number of features. Since GP produces a super-feature, p will always be 1. So the total complexity of GP (KNN) would be $O(n)$ + the complexity of GP feature. Based on above discussion we can say that the total complexity of GP (KNN) is more than the other two classifiers but it is over-weighed by gain in performance.

7.8 Summary

Many methods have been presented in the past for making GP a multi-class classifier. One method used simple threshold values for different classes for evaluating fitness value where GP features were trained to fit the corresponding classes in those thresholds. This method is simple and works well if the classes are distributed well in a non-overlapping manner but if there is considerable overlap, the method fails. Another approach is to use multiple binary classifiers and in the end combine results to get overall classification rate. For an n class problem, the resources required for this method are n times the resources required for binary problem. This method is suitable only if n is small because for a large n, a large number of GPs need to be trained increasing the computational complexity.

A new approach to extend the role of GP for multi-class classification has been proposed. Machine learning classifier KNN is used to make GP suitable for multi-class classification. The classifier is used for fitness evaluation of the features during training where the classification error rate is employed as fitness value. The assignment of fitness evaluation task to KNN enables GP to deal with multiple classes.

Modulation classification is used as a case study problem for evaluating the performance of the proposed method. It is a well known communication problem and has various civilian and military applications. Four modulations BPSK, QPSK, 16QAM and 64QAM which have gained much popularity in communication systems recently have been used. FB method using cumulants as features is used for classification. The channel is assumed to be AWGN.

Although using cumulants as features have improved performance, but a combination of cumulants will surely return better performance. In literature, different combinations of cumulants have been tried but the combinations were made manually by researchers. This is time consuming and it is still difficult to try all the combinations. In this study, the process of combining cumulants to make a new feature having more discriminatory power has been automated by using GP. GP evolves different combinations and returns the combination best suited for classification. The process is well guided by KNN which evaluates the ability of each combination in separating the modulations.

The best combination returned by GP is tested using test data by KNN classifier. The performance achieved is compared with Naïve Bayes and SVM classifiers at different SNRs and number of samples. The comparison shows the superiority of GP (KNN) against other methods at all SNRs and number of samples. The results show that classification accuracies of the classes which are easier to separate are much higher than others. One possible reason is the bias in GP during evolution towards the classes, easier to separate. This problem is dealt with in the next chapter.

Chapter 8

Multi-Stage Genetic Programming for Improved Modulation Classification

8.1 Introduction

In the previous chapter we proposed a method for making GP suitable for multi-class classification. The proposed method in comparison with other state of the art systems showed considerable improvement. One of the drawback of the proposed method, the favouritism shown by GP during training to easily separable classes, was mentioned in the previous chapter. This drawback results in high classification accuracies for some classes and low for others. In this chapter this drawback is addressed by dividing the training process into two stages. A separate second stage is used for the classes having low classification accuracy at the first stage.

A similar drawback of GP, the inability to deal with imbalanced data, was discussed in detail in Chapter 6. The favouritism shown by GP towards majority class was tackled by a weighted fitness function. Although the problem faced here is similar (GP's favouritism to certain classes) but the cause of the problem is different.

This chapter proposes a method to deal with this problem by dividing the training process into two stages. At the first stage the classes difficult to separate are treated as a single class and a second stage is used solely for separating the difficult classes. The first stage is similar to the method presented in previous chapter where KNN was used for fitness evaluation. The difference is that the difficult classes are treated as one class at the first stage and are separated at the second stage. Different machine learning classifiers including SVM and ANN are used at the second stage for binary classification. Since the problem is reduced to binary classification at the second stage, Fisher criterion is also used for evaluating fitness function (used for diabetes detection previously). A comparison of the results using single and two stages is presented showing the improvement achieved using two stages. Moreover, a comparison between different GPs (employing different fitness evaluators) used at the second stage is provided.

The division of the classes (easy and difficult) has to be done before the training process. The training process at each stage is performed according to this division. In this research the statistical distribution of the original features has been used to decide which classes are easier and which classes are difficult. Although, the proposed system is specifically aimed at improving the classification accuracy of modulation classification problem, it could be employed for any multi-class classification.

The chapter is organised as follows: the proposed method and its different stages are presented initially. Different fitness evaluators and their use in GP is discussed in detail. The experimental results, analysis and comparison with other methods is presented at the end.

8.2 System Design

In this chapter, a two stage genetic programming is developed to tackle the bias shown by GP during training towards classes, easier to classify. In the last chapter, KNN based fitness evaluator was developed within GP paradigm to enable GP in handling multiple classes. Modulation classification was used as a case study problem with four modulations BPSK, QPSK, 16QAM and 64QAM. Cumulants were extracted as features from the received noisy modulated data. The statistical distribution of the cumulants for different SNRs showed that BPSK and QPSK are easier to classify compared to the other two modulations (see Table 7.2 - Table 7.4). It was demonstrated through experiments that the evolution of GP was biased towards BPSK and QPSK, resulting in poor classification accuracies for 16QAM and 64QAM. In this chapter a new method is proposed which divides the training process of GP into two stages where the second stage is solely devoted for the classification of 16QAM and 64QAM. Introduction of stages helps to improve the classification accuracies and it is demonstrated through experiments.

8.2.1 Two-Stage GP

The choice of classes for the two stages was not only based on the statistical distribution of features but the previous literature also helped to make this decision. In [130], Swami and Sadler gave a detailed analysis of AMC for different modulation schemes. They used higher order cumulants for classification of a large set of modulation schemes. They also did classification of BPSK, QPSK and QAM(>4). Their results showed that the classification of 16QAM and 64QAM was difficult compared to other modulations. Wong and Nandi [162] also used higher order cumulants for the classification of above mentioned modulations. Also they came to the conclusion that the classification of QAM(>4) is difficult as compared to classification of other two modulation schemes. This shows that classification of BPSK, QPSK and QAM(>4) is easier as compared to the classification of 16QAM and 64QAM. In order to cope with this problem we have used a two-stage genetic programming.

At the first stage, classification of BPSK, QPSK and QAM(>4) is performed and at the second stage GP is used again to do the classification of remaining two classes. Since there are four classes at the first stage and GP needs to do multi-class classification, KNN has been used as a fitness evaluator. The use of KNN for fitness evaluation makes GP a multi-class classifier. SVM or ANN can also be used at the first stage but these two classifiers are computationally complex compared to KNN. Since the training of GP is a computationally complex process, KNN for fitness evaluation is the preferred choice because of its simplicity. At the second stage KNN, SVM, ANN and Fisher criterion are used for fitness evaluation of binary classification problem, producing four solutions one for each GP combination. As the second stage is independent from the first and solely devoted for the classification of QAM(>4) modulations, the classification accuracy should increase.

In addition to the other two stages (used for increasing classification accuracy of 16QAM and 64QAM), the proposed system can be divided into training and testing phases. During the training phase cumulants are given as input features to GP. Different combinations of cumulants are tested by GP during the evolution and the best combination generated during the evolution is returned as final result. GP differs from other machine learning classifiers in the sense that it does not return a trained classifier as a result of training and instead returns a solution by using combinations of the input features, later tested by an independent classifier. The system model showing training phase is shown in Figure 8.1.

The training process can be explained as: initially cumulants are extracted

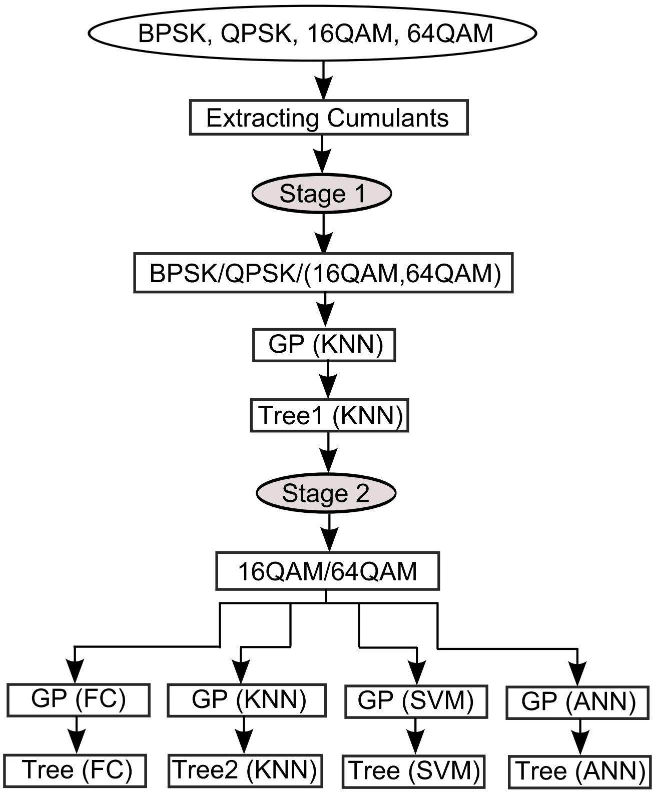


Figure 8.1: System model showing training phase of two-stage modulation classification.

from the received signal. These cumulants are given as inputs to GP (KNN) at the first stage where KNN has been used in combination with GP to make GP a multi-class classifier. KNN is used for fitness evaluation during the training. At the first stage, three-class classification is performed as explained earlier. Since classification of BPSK and QPSK is easier compared to 16QAM and 64QAM, a second stage classification is performed to separate 16QAM and 64QAM. After the completion of first stage GP (KNN) produces Tree1 (solution 1) which is the best tree produced during all GP generations, in terms of separation between these three classes. The remaining two modulations are fed into the second stage where three types of classifiers (KNN, SVM and ANN) are used for fitness evaluation of GP trees. Details about how these classifiers are used for fitness calculation are given in the next sections. In addition, Fisher criterion (FC) has also been used at the second stage which has the ability to provide a cost efficient solution for binary classification. Once the training phase is completed, five trees are produced, one from the first stage and four from the second stage. In Figure 8.1 the term inside parenthesis is used for fitness evaluation. For example in GP (KNN), KNN has been used for fitness evaluation of trees and same is the case for other notations.

Once the training process is complete, the trees produced by GP are tested with test data. Figure 8.2 shows the full testing process. We again start with extracting cumulants from the received signals. These cumulants are fed into the Tree1 produced by GP (KNN) at the first stage which performs three-class classification. Since Tree1 was produced by GP (KNN), the performance of this tree is tested using KNN (other classifiers can also be used). BPSK and QPSK are separated at this stage and the remaining two classes are fed into the second stage. Four trees produced during the second stage of the training phase, through different GP combinations are used at this stage, with each of them trying to separate 16QAM and 64QAM. The performance of each tree is tested with the same classifier that was used during fitness evaluation. For example, the performance of the tree produced by GP (SVM) is evaluated using SVM classifier and same is the case for other classifiers. For tree returned by GP (FC), KNN is used for testing because of its simplicity. Figure 8.2 shows different stages of the testing phase.

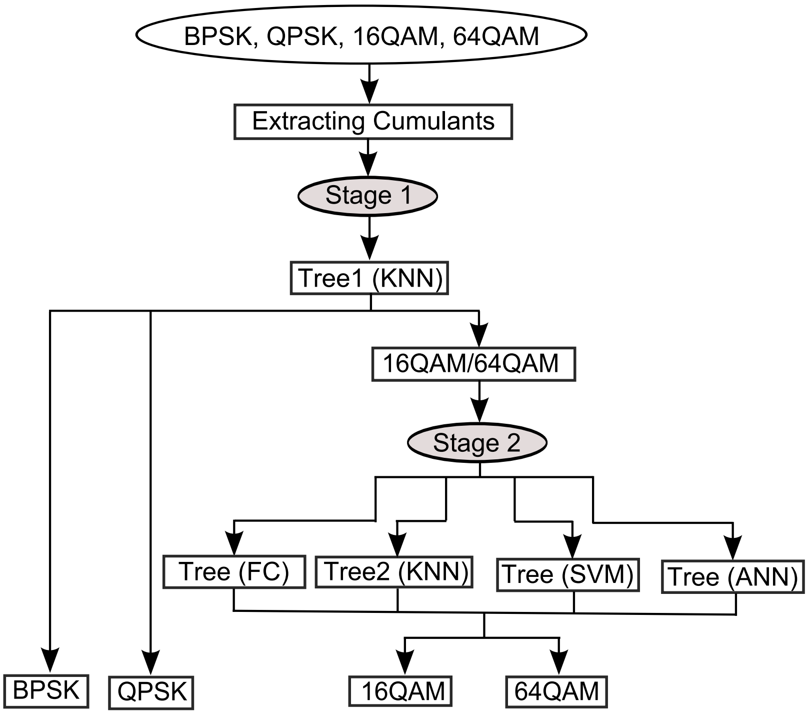


Figure 8.2: System model of testing phase for modulation classification.

8.2.2 Fitness Evaluation Using KNN

The evaluation of fitness using KNN was presented in Section 7.6.1. The same method is used here so it will not be discussed again and interested readers are referred to Section 7.6.1.

8.2.3 Fitness Evaluation Using SVM

The basic structure and working of SVM was explained in Section 2.6.2. SVM has been used for fitness evaluation during the training phase (stage 2) and later on it has been used for testing the best GP individual. Fitness evaluation is similar to the one used for KNN where the inverse of the number of errors made by individual was used as the fitness value. The process of fitness evaluation follows the following steps: four hundred evaluation values (two hundred for each modulation) are given as input to a GP tree. The output of GP tree is fed into SVM for obtaining separation between the two modulations. SVM makes a boundary (linear or non-linear) for separating two modulations. The samples which are on the wrong side of boundary are counted (also called as error). The inverse of the number of errors 'e' is the fitness function and can be written as,

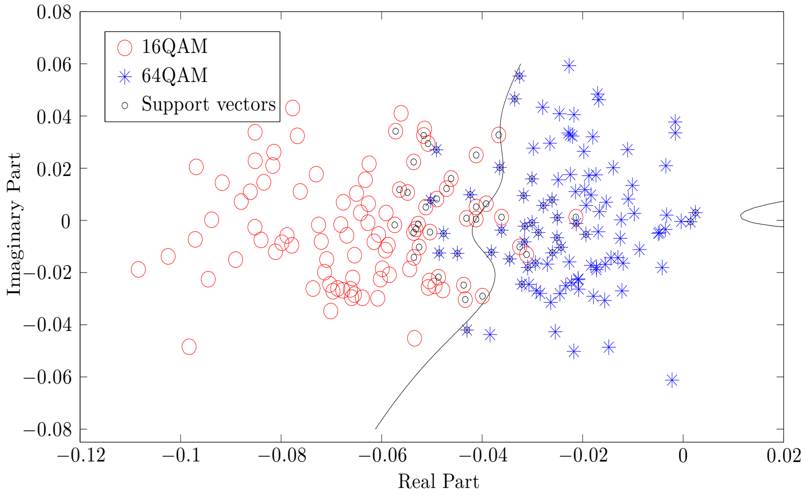


Figure 8.3: Feature space with 200 signal realizations for 16QAM and 64QAM using 10dB SNR and 512 number of samples.

$$f = (e_{16QAM} + e_{64QAM})^{-1} \tag{8.1}$$

where e_{16QAM} represents the 16QAM samples on the wrong side of the separation plane and e_{64QAM} represents the samples of 64QAM ending up on the the wrong side of the separation plane.

The boundary shown in Figure 8.3 is a trained separation plane using SVM with fifth order polynomial kernel function. During the training phase, SVM was trained using a third order polynomial kernel function for a more efficient evolutionary process. A higher order polynomial could also be used but it will increase the complexity of fitness evaluation during the training phase of GP which ultimately will result in longer time taken by GP for producing a solution. An important point to mention is that the job of SVM is just to assist GP in finding the best individual during the training phase and the SVM trained during training phase has no role at the testing phase. A new SVM with fifth order polynomial kernel function is used at the testing phase using the test data. The ability of the best individual returned during the training phase to discriminate between 16QAM and 64QAM is tested by the new SVM. Table 8.1 shows the parameters values used for SVM during training and testing phase.

Table 8.1: SVM parameters

Parameters	Values	
	Training phase	Testing phase
Number of training samples	400	800
Number of testing samples	N/A	20000
Kernel function	Polynomial	Polynomial
Polynomial order	3	5
Method	Quadratic programming	Quadratic programming

Table 8.2: ANN parameters

Parameters	Values	
	Training phase	Testing phase
Number of training samples	400	800
Number of testing samples	N/A	20000
Hidden layer size	3	10

8.2.4 Fitness Evaluation Using ANN

ANN has been used in a similar fashion as SVM for evaluating the fitness of individuals. The inverse of the number of errors made by ANN is taken as fitness of individual where higher fitness means better individual. Four hundred reference samples for both modulations have been used to evaluate the fitness value. These 400 values of new feature (the output of a GP tree) are given as input to ANN which tries to separate them as much as possible. The inverse of the number of errors (fitness value) made by ANN is calculated as done in SVM case.

The best individual returned by GP (ANN) is tested by a more complex ANN during testing phase. The ANN used during testing phase is different to the ANN used during training and the reason behind this is the same as explained in previous section for SVM. Since the use of ANN makes the process of training slow, lesser number of samples has been used during the training phase while at the testing phase, the new ANN with larger number of samples is used. Table 8.2 shows the parameters values used for ANN during training and testing phase.

8.2.5 Complexity of the Proposed Classifiers

The complexity of the final solution depends on three factors; the complexity of calculating cumulant features, the functions used by GP returned solution, and the complexity of the classifier used at testing phase (KNN/ANN/SVM). The complexity of the solution returned as a result of training phase depends on the function used by this solution from function pool and the the cumulants used from the terminal pool. Depending on the functions used (linear or non-linear), complexity can vary a great deal. This solution is tested by one of the classifiers using the test data. Each classifier has its own complexity which depends on different parameters used for that classifier. The computational complexity of these state-of-the-art classifiers is widely available in literature [130, 163–165] . The overall complexity is equal to *the complexity of the GP returned solution + the complexity of the classifier.*

Since only the final solution returned by GP is used for making new feature, the time taken for training of GP should not be included in the complexity of final classifier. However, if the time taken for the training phase by different GP classifiers is compared, the time taken for training by GP (ANN) is highest followed by GP (SVM), GP (KNN) and GP (FC) respectively.

8.3 Experiments and Results

In this section results are presented for each method separately. We start with GP (KNN) and then the results for the other methods are explained. The parameters used for GP are given in Table 8.3. We have used 100 generations for all the experiments with a population of 25. The SNRs used are 0 dB to 20 dB (with an increment of two) and the numbers of samples are 512, 1024, 2048 and 4096. There are 16 combinations of SNR and the number of samples, and for each combination 25 experiments were performed. Since there were 25 individuals in every generation, a total of 625 trees were generated for each combination. The best tree was chosen out of these 625 trees and this tree was tested with 10,000 test samples. For evaluating average performance and standard deviation, 25 best trees out of 625 were selected. All the experiments were carried in MATLAB/GPLab environment.

Table 8.3: GP program parameters

Parameter	**Standard Value**
Number of Generation	100
Population Size	25
Function Pool	(plus, minus, times, mydivide, mysquare, mysqrt, reciprocal, sin, cos, myasin, myacos, tan, tanh, negator, abs, mylog)*
Terminal Pool	Moments and Cumulants
Genetic Operator	Crossover, Mutation
Operator Probability	0.9, 0.1
Tree Generation	Ramped half-and-half
Initial Maximum Depth	6
Maximum Depth	28
Selection Operator	Roulette
Elitism	Half-elitism

* All the functions starting with ”my” are protected functions.

8.3.1 GP with KNN

The performance of the best GP tree for different SNRs and 1024 number of samples is given in the form of confusion matrix in Table 8.4. The classification accuracies of BPSK and QPSK are collected from the first stage and the accuracies of 16QAM and 64QAM are collected from the second stage. If we compare the accuracies of QAM(>4) using two stage GP (Table 8.4) and one stage GP (Table 7.7), there is a clear advantage shown by two stage GP in terms of classification accuracy. At SNR of 5 and 10 dB the advantage in classification accuracy is more than 4% and 2% respectively. For SNRs greater than 10 dB (15 and 20 dB) the advantage is less evident and is always less than 2%. Since classification is most difficult at low SNRs, the improvement in classification accuracies at low SNRs is very important.

The classification accuracies for four-class classification, using different SNRs and number of samples are shown in Figure 8.4. It is clear from the Figure that using 2048 number of samples at 10 dB SNR, performance reaches 100%. The dotted lines around performance curves show the standard deviation of performance. One can see that the standard deviation is low showing the robustness of the method. Performance is quite steady above 10 dB SNR for 1024 and 2048 number of samples but as SNR goes below 10 dB, performance decreases

Table 8.4: Confusion matrix for GP (KNN) at 1024 number of samples and different SNRs

SNR	Modulation Type	BPSK	QPSK	16QAM	64QAM
5 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	8091	2780
	64QAM	0	0	1909	7220
10 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	9560	423
	64QAM	0	0	440	9577
15 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	9880	130
	64QAM	0	0	120	9870
20 dB	BPSK	10000	0	0	0
	QPSK	0	10000	0	0
	16QAM	0	0	9949	60
	64QAM	0	0	51	9940

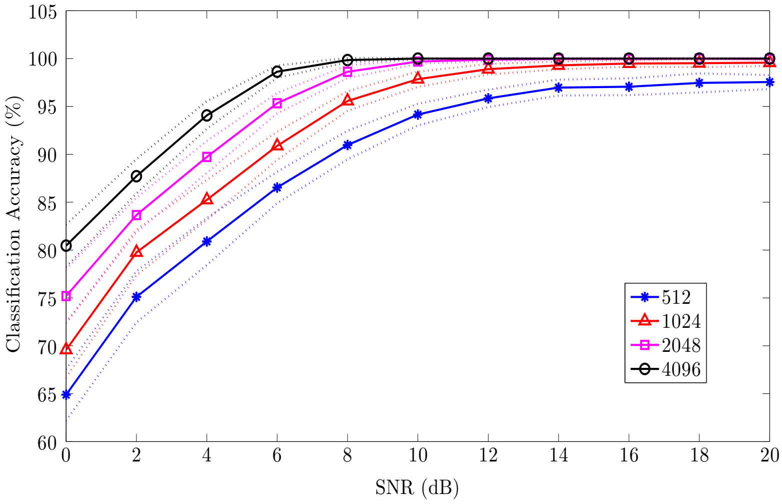


Figure 8.4: Classification accuracies against SNR for different number of samples for four-class (BPSK, QPSK, 16QAM, 64QAM) classification using GP (KNN).

sharply. We have seen in Table 8.4 that BPSK and QPSK are easily separable from 16QAM and 64QAM, so the decrease in performance at low SNRs is due to 16QAM and 64QAM.

Figure 8.5 shows the classification accuracies for 16QAM and 64QAM for different number of samples. It is clear from the figure that the performance for these particular modulations is low compared to the four-class classification performance. Again, the dotted lines showing the low standard deviation of performance prove the robustness of our method.

8.3.2 GP with SVM

The 25 best trees obtained by GP (SVM) are tested at different SNRs and using different number of samples. The results obtained are presented in Figure 8.6. The performance shown by GP (SVM) is similar to the performance of GP (KNN) especially at high SNRs. The performance curves of GP (KNN) are very smooth compared to GP (SVM) which tells us that the performance of GP (KNN) is steady compared to GP (SVM). The standard deviation of GP (SVM) is shown with dotted lines around performance curves. The standard deviation is relatively high compared to GP (KNN).

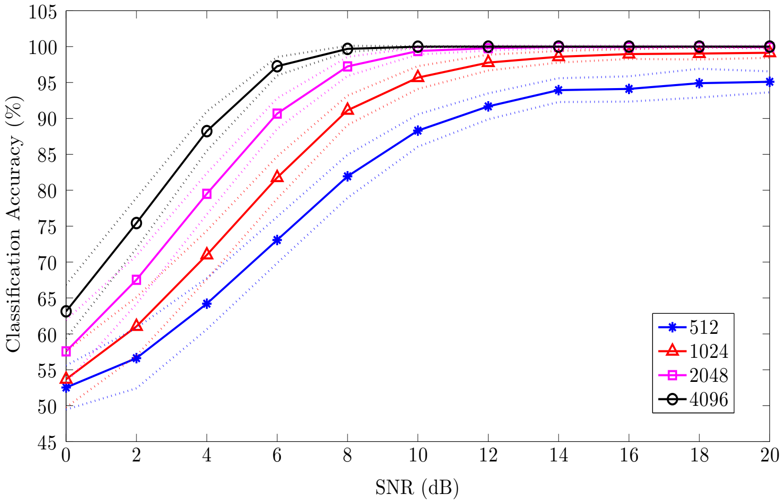


Figure 8.5: Classification accuracies against SNR for different number of samples for two-class (16QAM, 64QAM) classification using GP (KNN).

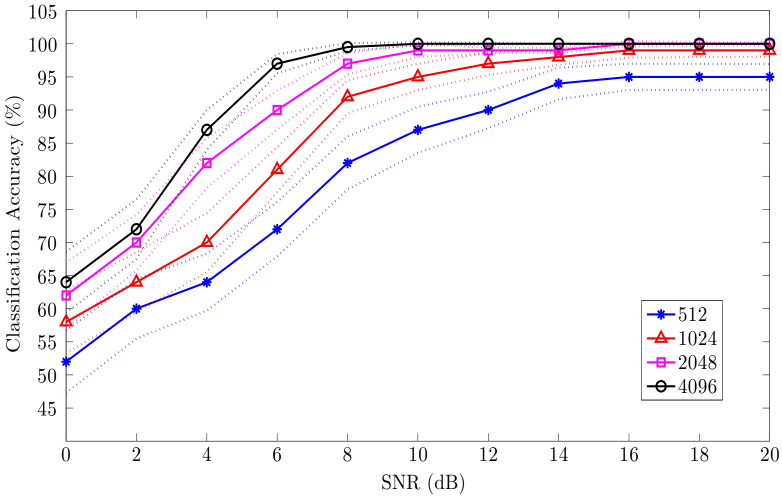


Figure 8.6: Classification accuracies against SNR for different number of samples for two-class (16QAM, 64QAM) classification using GP (SVM).

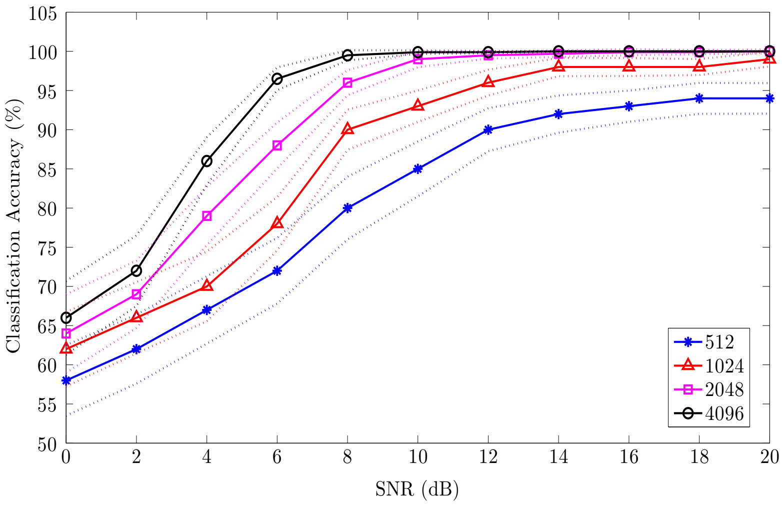


Figure 8.7: Classification accuracies against SNR for different number of samples for two-class (16QAM, 64QAM) classification using GP (ANN).

8.3.3 GP with ANN

The results after testing the 25 best trees with ANN are shown in Figure 8.7. The results demonstrate the effectiveness of ANN for classification of QAM signals. The results show that this method is able to give a performance of about 95% at and above 10 dB SNR using 1024 number of samples. It appears that the performance achieved using this method is similar to the other two except at low SNRs, where it performs better. The standard deviation is represented through dotted lines around the main curves.

8.3.4 GP with Fisher Criterion

Fisher criterion was used in the same fashion as in Chapter 5. The results are shown in Figure 8.8. The results demonstrate that this method outperforms all other methods at high SNRs (10 dB onwards) but its performance is lower than other methods at low SNRs (below 10 dB). Moreover, it has low standard deviation and less training time compared to all other proposed classifiers which makes it superior.

8.4 Comparisons

In this section results of GP with different classifiers are compared with each other and also they are compared to some other methods.

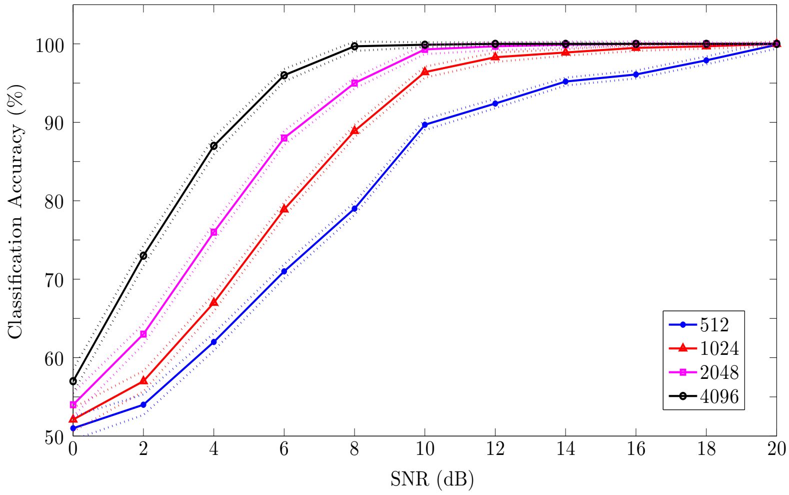


Figure 8.8: Classification accuracies against SNR for different number of samples for two-class (16QAM, 64QAM) classification using GP (FC).

The classification performance of GP in combination with different classifiers is shown in Figure 8.9. The number of received signal samples used is 512. It can be seen that roughly all the classifiers provide similar classification accuracy for most of the SNR range. However, in terms of computational complexity and time taken during training, GP (FC) is least complex followed by GP (KNN), GP (SVM) and GP (ANN).

The classification performance of GP with different classifiers against the number of received signal samples at an SNR of 10 dB is shown in Figure 8.10. From the Figure it is clear that that the classification accuracy achieved using GP (FC) is better than the other three methods for most of the number of samples.

Table 8.5 summarizes the performance of different methods used recently for QAM classification. Due to different sets of modulations and different channel conditions, the direct comparison with some of the methods present in literature is not possible but a fair comparison considering all the conditions is presented.

Wong, Ting and Nandi [162] presented results for the same modulations and we discussed and compared few of their results with ours in the previous chapter. In this chapter a more detailed comparison is presented. They used Naïve Bayes, SVM and maximum likelihood (ML) methods for classification. We have produced ML results ourselves using the same method because the results reported in their research do not look correct. Figure 8.11 shows the comparison of our results with their method. ML gives the upper bound performance but have more computational complexity. It is clear from the Figure that our method gives bet-

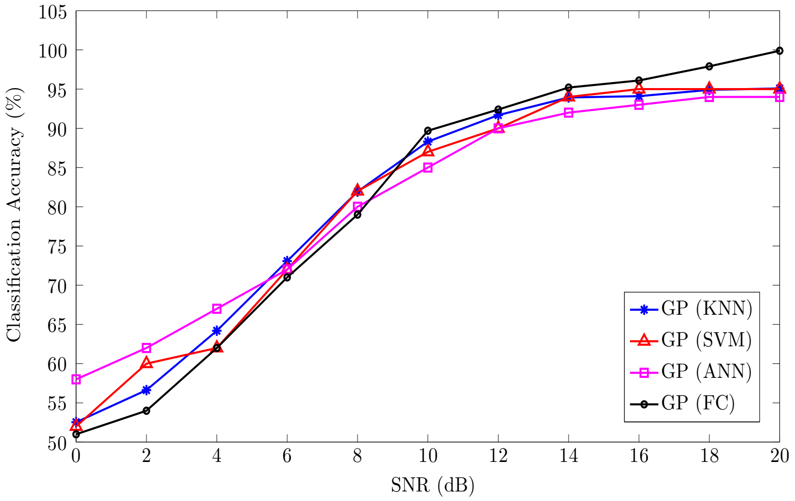


Figure 8.9: Performance comparison of different GP methods against SNR for 512 numbers of samples.

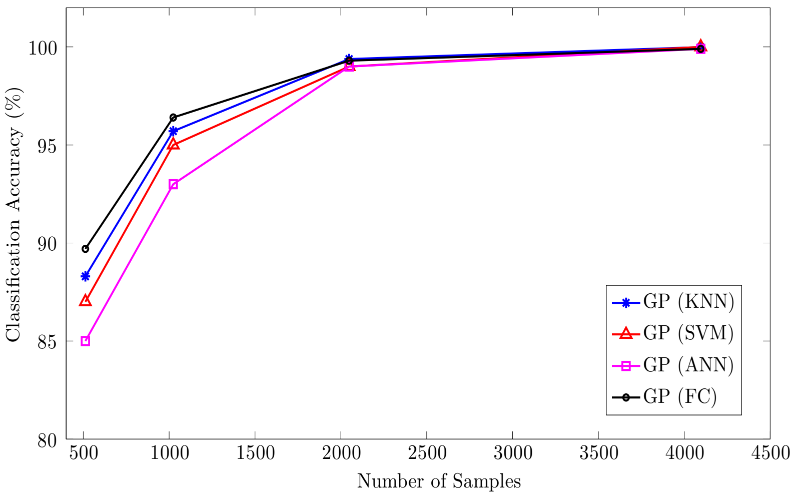


Figure 8.10: Performance comparison of different GP methods against number of samples at an SNR of 10 dB.

ter results compared to SVM and Naïve Bayes method. Up to 3 dB SNR, the performance of our method is same as the other two methods but after 3 dB, the performance of our method improves compared to the other two.

In [130], Swami and Sadler presented a detailed analysis about the classification of different digital modulation schemes using cumulants. They also presented classification results for 16QAM and 64QAM. They reported classification accuracy of 90% but their method had some limitations. There were two limitations: firstly the conditions were assumed to be noiseless and secondly the number of samples used was more than 10,000. Our method achieved 100% accuracy at 15 dB SNR and using 2048 number of samples. In order to compare our results with cumulants results we implemented the method presented in [130] to classify the four modulations at a range of SNRs. The same tree structure and the same cumulants as presented in [130] were used for results but with noisy conditions. Thresholds for classification tree were not specified in [130] but a Table was presented with theoretical values of cumulants and we have calculated thresholds from that Table.

Classification accuracies were calculated at different SNRs using 1024 number of samples. These classification accuracies were averaged over 10,000 runs. We found classification accuracies (in percentage) for cumulants to be 98.2±0.22 at 20 dB (compared with 100±0 using GP (KNN)), 87.1±0.45 at 15 dB (compared with 99±0) and 57.3±0.35 at 10 dB (compared with 98±0). The results clearly demonstrate the improvement in performance using our method compared to cumulants method. However, it will be possible to augment the method presented in [130] by making thresholds SNR dependant and thereby to increase classification accuracies but not challenging the proposed method.

Xi and Wu [157] reported classification accuracy of 94% for an SNR of 10 dB using 2000 number of samples and the modulations used were QPSK, 16QAM and 64QAM. They used fourth order cumulants for this purpose. We achieved performance of 99.6% at an SNR of 10 dB and using 2048 number of samples. Mirarab and Sobhani [156] reported performance of 94% at an SNR of 15 dB using 2000 number of samples. They used eighth order cumulants and a threshold dependent tree structure for this purpose. At the same SNR and at 2048 number of samples we achieved performance of 100%.

Dobre, Ness and Su [140] reported classification accuracy of 70% at an SNR of 10 dB using 2000 samples. They used eighth order cyclic cumulants for classification. At 10 dB and 2048 number of samples we achieved 100% classification accuracy. Wu, Saquib and Yun [167] presented results for QPSK, 16QAM and

Table 8.5: Performance comparison of GP with other methods

Method	**Modulations**	**Channel**	**Performance/ No. of Samples/ SNR (dB)**
Naïve Bayes [162]	BPSK/QPSK/ 16QAM/64QAM	AWGN	90%/ 512/ 10
SVM [162]	BPSK/QPSK/ 16QAM/64QAM	AWGN	91%/ 512/ 10
GP (FC) (This study)	**BPSK/QPSK/ 16QAM/64QAM**	**AWGN**	**95%/ 512/ 10**
Cumulants [130]	16QAM/64QAM	Noise Free	90%/ 10,512/ No Noise
GP (FC) (This study)	**16QAM/64QAM**	**Noise Free**	**100%/ 10, 512/ No Noise**
Cumulants [140]	16QAM/64QAM	AWGN	95%/ 8,000/ 5
GP (ANN) (This study)	**16QAM/64QAM**	**AWGN**	**98%/ 8,000/ 5**
Kolmogorov-Smirnov [166]	QPSK/16QAM/ 64QAM	AWGN	78%/ 100/ 10
GP (FC) (This study)	**QPSK/16QAM/ 64QAM**	**AWGN**	**82%/ 100/ 10**

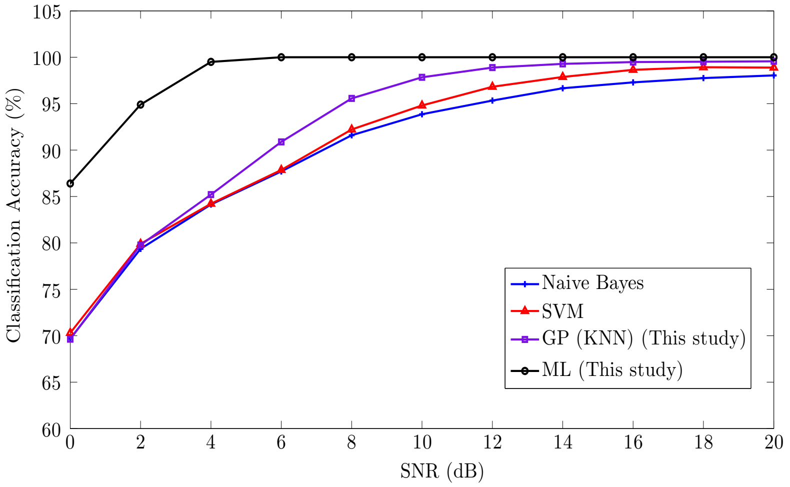


Figure 8.11: Performance comparison of GP-KNN with other methods at different SNRs and using 1024 number of samples.

64QAM using higher order statistics. They reported classification accuracy of 94% at an SNR of 10 dB and using 2000 number of samples. We have achieved 99.6% classification accuracy at the same SNR and using 2048 number of samples for four class classification.

In [166], F. Wang and X. Wang used the Kolmogorov-Smirnov (KS) test for smaller signal samples. The modulations used were QPSK, 16QAM and 64QAM. The method presented provided good classification accuracy using low number of samples but the improvement in performance was relevant only for high SNRs. It was demonstrated in [166] that KS results and cumulants results were very similar from 0 dB to 10 dB and the improvement was only from 10 dB to 20 dB. We have done experiments to compare our results with their results using the same three modulations and using 100 samples. We found that at lower SNR, GP (FC) performs much better (e.g., 77% versus 67% at 5 dB and 82% versus 78% at 10 dB), while at higher SNR their method performs better (e.g., 86% versus 98% at 15 dB and 87% versus 100% at 20 dB). This shows that GP (FC) is quite robust for a long range of SNRs and possesses graceful performance degradation. In [168], the complexity of KS classifier has been reduced by using off line CDF curves but the empirical CDF calculation and comparisons with offline CDF curves are still there. Although the complexity of our method is a little higher than the KS method (after quantization) but the improvement in performance at lower SNR and robustness over a range of SNR makes it competitive.

8.5 Summary

It was demonstrated in the previous chapter that the biased behaviour of GP towards the classes, easily separable, makes it difficult to achieve high classification accuracy for the classes, difficult to separate. In this chapter a two-stage GP has been introduced to overcome this limitation. At the first stage GP (KNN) has been used for handling multi-class classification problem. The difficult classes are treated as a single class at the first stage. At the second stage another GP training process is run for the separation of remaining classes. Since in this particular problem there were only two classes at the second stage, GP as a binary classifier was used.

At the second stage in addition to KNN some other machine learning classifiers including SVM and ANN have also been used for fitness evaluation. Using these classifiers during training phase to guide the process is believed to increase the final classification accuracy as the GP generated features are ultimately tested by one of these classifiers at testing phase. Fisher criterion has also been used at the second stage for fitness evaluation.

The comparison between different GPs (employing different fitness evaluators) shows that the performance of all these GPs is similar with slight variations. The standard deviation in case of GP (FC) is much lower making it more robust than the other two. Moreover, in terms of computational complexity also GP (FC) is simpler than the other two, making it superior than others for fitness evaluation. However, the main drawback of GP (FC) is the sharp decrease in performance as the SNR goes below 10 dB. The performance at low SNRs is important because classification is most difficult at these SNRs.

The experimental results and comparison with existing methods showed that the GP generated features provided the best classification results for most of the SNRs. There were some methods which performed better than the proposed method at some SNRs but the computational complexity of those methods was high.

Chapter 9

Conclusions

In this thesis feature generation, an important part of any PR system has been explored by using an evolutionary computation technique called GP. The design presented in this research benefits from the inherent feature selection and transformation capabilities of GP. Some new strategies are developed to enable GP overcome some of its limitations and to allow GP to deal with classification problems more efficiently. This chapter outlines the contributions of this research and suggests some possible future directions.

9.1 Discussion & Remarks

In the coming text, the proposed changes to overcome the deficiencies of the standard GP approach for classification problems will be discussed and conclusions will be drawn regarding the important changes made to GP to make it more suitable for classification purpose.

Creation of ***Super-features*** using GP

The goal of this thesis was to modify and make extensions to standard GP to make it more suitable for feature generation to assist in classification tasks. In this thesis the feature generation is performed by GP which performs feature selection and feature transformation in a single step. The benefit of using GP for feature generation is that it does not require any knowledge about the statistical distribution of the features. Moreover, the fitness parameter of GP makes sure that once a certain quality level for solutions is achieved, it never drops below that level. During the evolution of GP unwanted features are ignored and only the features with discrimination information contribute to the solution. GP is evolved to generate a single feature (referred as *super-feature*) by making non-linear combinations of the input features. This could be considered as a special case of dimension reduction where the resulting number of dimension is always one. GP is given a generalised set of linear and non-linear functions which proved to be capable of finding the better combination of input features, resulting in a single feature for classification problems.

Introduction of Binary String Fitness Characterisation for classification problems

In standard GP, only one parameter, the fitness of the individual is used to check the ability of individual in solving the problem. This can result in a population wide weakness because of the lack of genetic material to solve all the training examples. A new method of binary string fitness characterisation is introduced to evaluate the relative strengths and weaknesses of individuals for classification problems. The binary string is updated in a unique way to assist the primary task of separating the underlying patterns which ultimately results in better classification.

Introduction of Comparative Partner Selection in crossover operation for real world classification problems

Following the strategy of the binary string, introduced to promote population wide strength, more efficient technique for selection of parents for genetic operator crossover is used. A computationally efficient method based on logic operations is used to favour the crossover between the individuals having strengths in different areas. This partner selection strategy (called CPS) promotes the individuals capable of solving all the training examples. It is demonstrated through experiments that this strategy not only improves the population fitness during evolution but also increases the classification accuracy of a complex real world binary classification problem. The strategy has been applied for diabetes detection, a binary classification problem, successfully with considerable improvement in classification accuracy compared to other state of the art systems available in the literature.

Introduction of Improved Comparative Partner Selection

An improved CPS strategy is introduced which gives more importance to CPS selected parents. The CPS selected parents are allowed to produce ten offspring while all other parents can produce only two. The ten offspring are produced by crossover of CPS selected parents, each time with a different crossover point. The idea is to explore the search space close to CPS selected parents more efficiently since they are supposed to have the ability to produce optimum solutions. The new strategy is applied on diabetes classification problem and it shows considerable improvement in performance compared to CPS strategy.

Imbalanced data classification using GP

GP shows a bias during evolution towards the class having more number of samples. New classification strategy and fitness functions are introduced to neutralise this bias. The classification strategy is based on the PDF of the underlying classes and finds the optimum threshold for separating the classes. The input samples are then labelled according to this threshold. Different fitness functions are also used to keep a balance between sensitivity and specificity during evolution. Experiments reveal that giving equal weight to sensitivity and specificity during evolution gives the best results.

Multi-class classification using GP

Generally a multi-class classification is decomposed into several 2-class classification problems for GP. In this thesis one of the simplest machine learning classifier KNN has been used for fitness evaluation which makes GP suitable for multi-class classification at the cost of a small increase in computational cost. KNN not only extends the role of GP to multi-class classification but improves the classification performance of the final classifier since each individual is already tested during evolution. A drawback of this technique was the favouritism shown by GP towards classes which are more easily separable to increase the overall classification accuracy. This was dealt by adding another binary classification stage for the classes difficult to classify.

Introduction of machine learning classifiers for fitness evaluation of GP individuals

Previous attempts to use GP for classification tasks have reported the tendency of GP to converge at unsatisfactory solutions mainly because fitness was not suited for classification methodology. The problem has been dealt in this thesis by employing machine learning classifiers for fitness evaluation namely KNN, SVM and ANN. Since these classifiers are mainly used for classification, they are better suited for evaluating fitness. Moreover, since the solution returned by GP ultimately needs to be tested by one of these classifiers as a part of pattern recognition procedure, using these classifiers as part of evolution process improves the performance of the final classifier.

Competitive solutions for modulation classification

A broad set of features has been proposed by researchers for modulation classification problem. Recently higher order statistics based features have been used extensively for modulation classification. Researchers who tried different combinations of the the features [130] suggested that a combination of the original features returns better performance. In the past, different combinations were tried manually to find out the best combination. In this research the process of finding the best combination of features is automated by using GP. GP not only ignores the less useful input features but produces a one single feature by making non-linear combinations of input features. Different machine learning classifiers are used to evaluate the fitness function of GP, resulting in many new features. A series of experiments have been conducted to evaluate the performance of new features generated by different GP combinations. Experimental results demonstrate the capability of GP for identifying the useful features among the raw features and making the best possible combination of features for classification. GP generated features not only provide better classification accuracy for modulation classification but also show robustness over a range of SNRs when compared to other classical classification methods.

9.2 Future Directions

The work of this thesis suggests the following directions for future work.

1. The focus of this research was on the feature generation using GP and variation in the performance by changing different parameters of the GP was not investigated. It is a valuable research issue to investigate the effect of variation of different GP parameters on the performance for classification problems. For example, a generalised function set was used for all classification problems and an analysis of the impact of choice of function set on feature generation would not only yield a better understanding of function set but will also help in improving the performance of GP.

2. Like other machine learning methods GP also demands significant computation especially if the number of input features is high so an interesting area is to develop methods for reducing the computational complexity of the standard GP. Although the training time taken by GP does not reflect in the computational complexity of the final solution, it will still be useful to reduce the training time of GP to improve its capability to handle large amount of data.

3. The binary nature of the binary string makes it well suited for logical problems but the evaluation of binary string for non binary problems is not well defined. A simple yet effective method for evaluation of the binary string is presented in this thesis for classification problem. This area is not well explored and there is room for improvement.

4. Although the CPS method provides significant performance improvement for most cases, it does not always guarantee the improvement in performance. One possible reason is that the CPS method is not always able to find suitable partners. The current strategy to deal with this scenario is to use a standard method for partner selection ignoring the CPS with increase in mutation rate. A better solution could be to find out for which training examples the individuals perform poorly and weight the fitness function in a way to give more importance to those examples.

5. For the diabetes problem, the missing values (not mentioned by donor) are just taken as zeros and more sophisticated way to deal with missing values (e.g using the mean) could be employed. More analysis on the role of missing values for this problem is required as it can make a huge impact on final performance.

6. The channel environment for modulation classification problem is taken as ideal with the addition of AWGN only. In a real world scenario there are many other factors that degrade the signal quality as it passes through channel which must be taken into account. A detailed analysis on the impact of noisy fading channel on modulated signal needs to be carried out.

7. In all modulation experiments, it is assumed that the SNR is already known which is not the case in reality. Although many SNR estimation algorithms are available in the literature which could be used, this author believes that there is perhaps potential in GP to estimate SNR more efficiently.

Appendix A

Best Trees Produced by GP

A.1 Best Trees for Diabetes Detection

The best trees obtained for diabetes detection problem are included in this section. A lot of trees were produced during training to take an average of GP performance but only a few are included here. The purpose is to give an idea to the reader about the general structure of GP trees and functions and terminals used in these trees. Three trees are included here for diabetes problem. These three trees are for first three sets of diabetes dataset among 10 sets. Since the classification of this dataset was quite hard especially because of presence of missing values, most of the trees created by GP are quite huge. The third tree is in particular huge with 333 nodes. The variables $X1$-$X8$ represent eight input diabetes features.

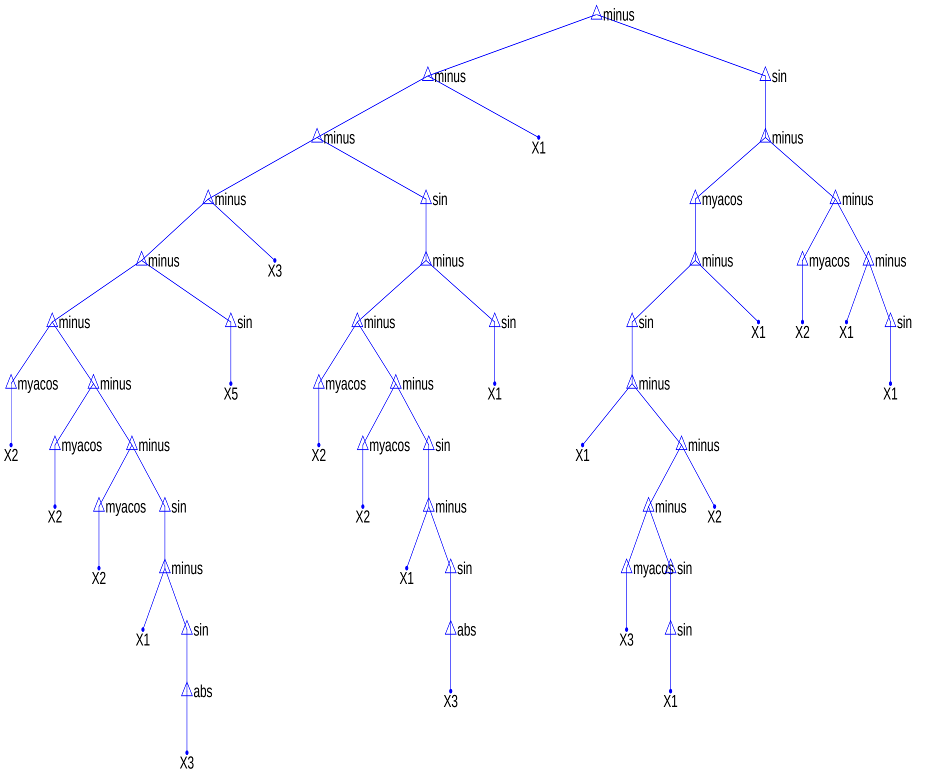


Figure A.1: One of the best trees for dataset1 of diabetes using five features

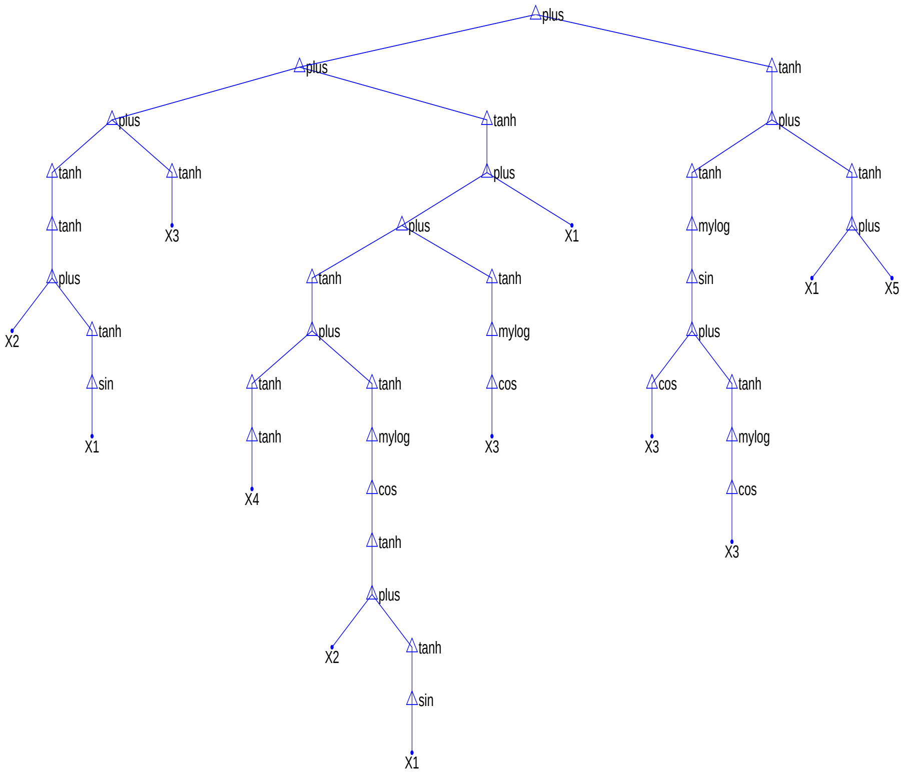


Figure A.2: One of the best trees for dataset2 of diabetes using five features

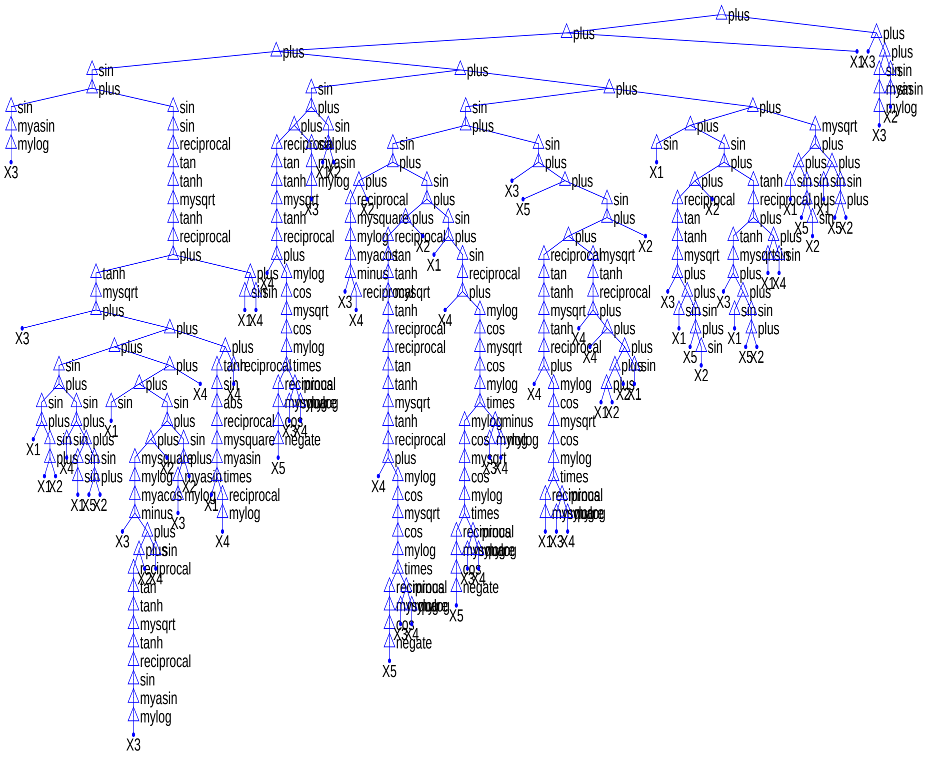


Figure A.3: One of the best trees for dataset3 of diabetes using five features

A.2 Best Trees for Modulation Classification

This section gives few trees for modulation classification. All the trees are for GP (KNN) and using 1024 number of samples. Trees are given for different SNRs for both 4-class ans 2-class modulation classification. The variables $X1$-$X7$ represent the cumulants given in Section 4.2.5, while the variable $X8$ and $X9$ represent the mean of real and imaginary part of the received signal respectively.

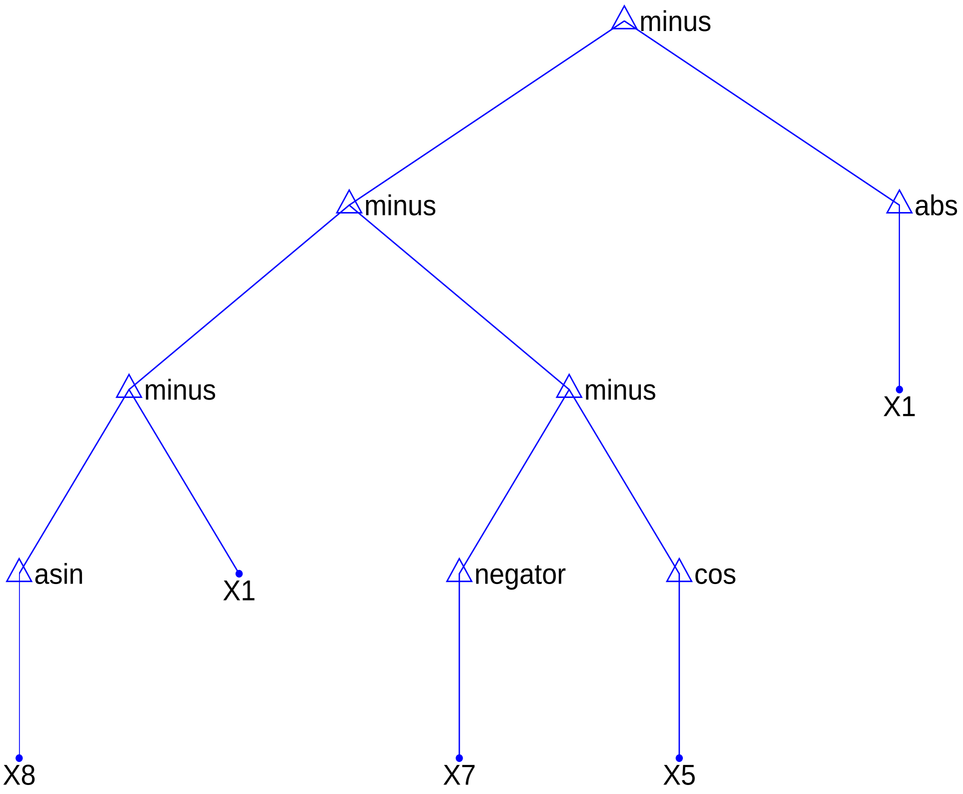


Figure A.4: One of the best trees for 4-class modulation classification for 1024 number of samples at 20 dB SNR

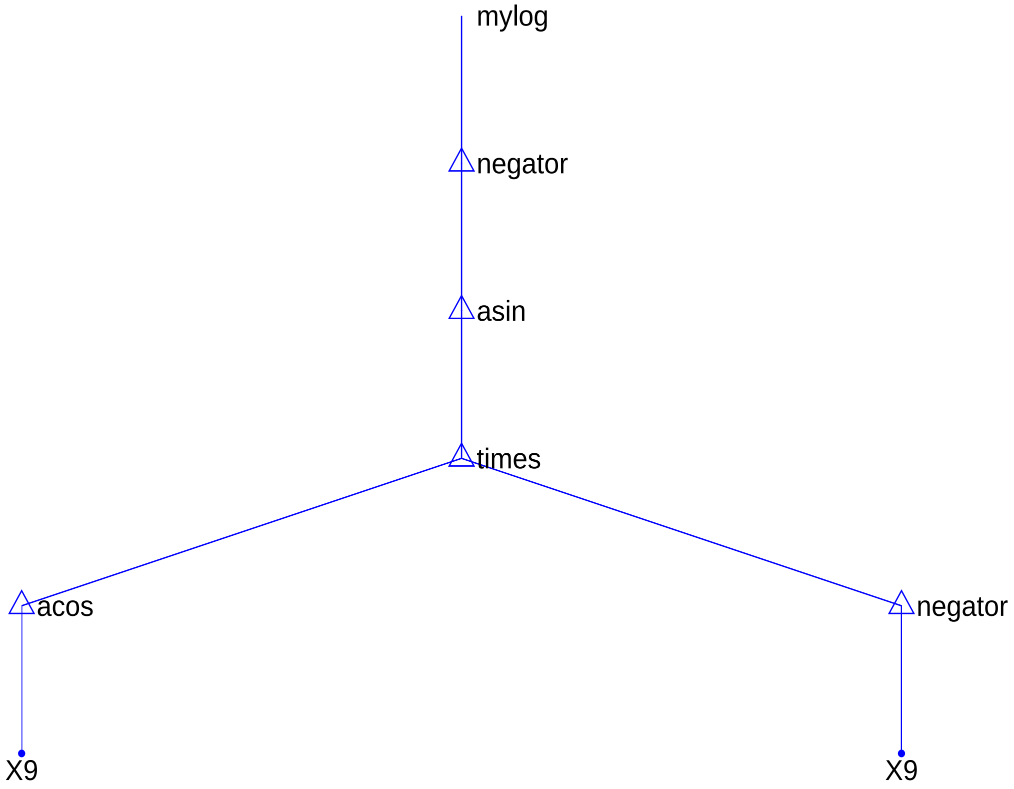


Figure A.5: One of the best trees for 2-class modulation classification for 1024 number of samples at 20 dB SNR

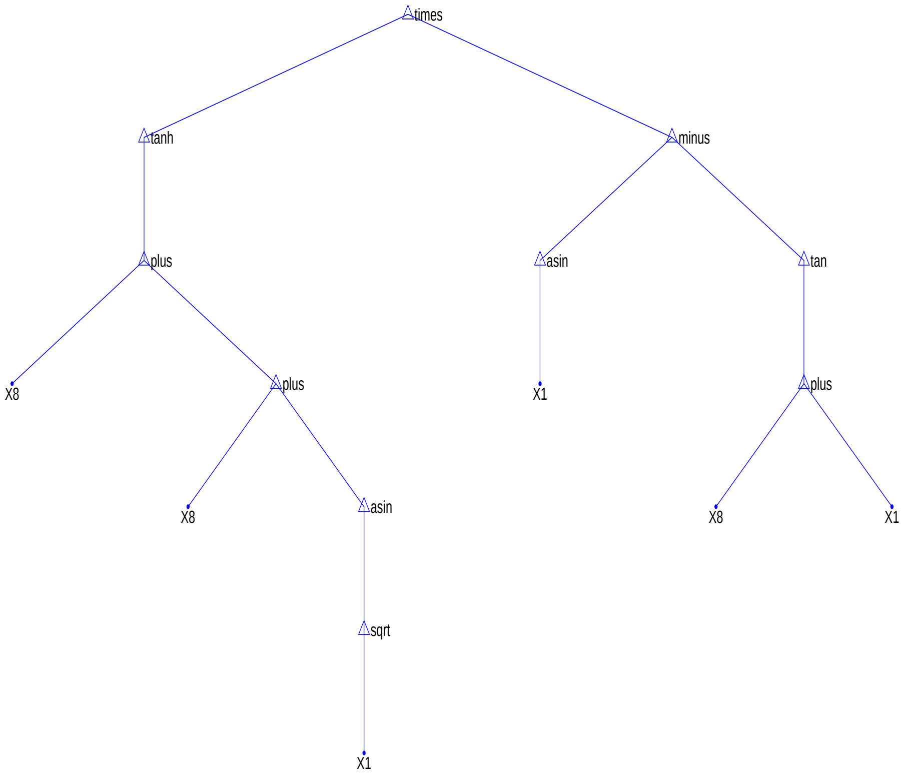


Figure A.6: One of the best trees for 4-class modulation classification for 1024 number of samples at 15 dB SNR

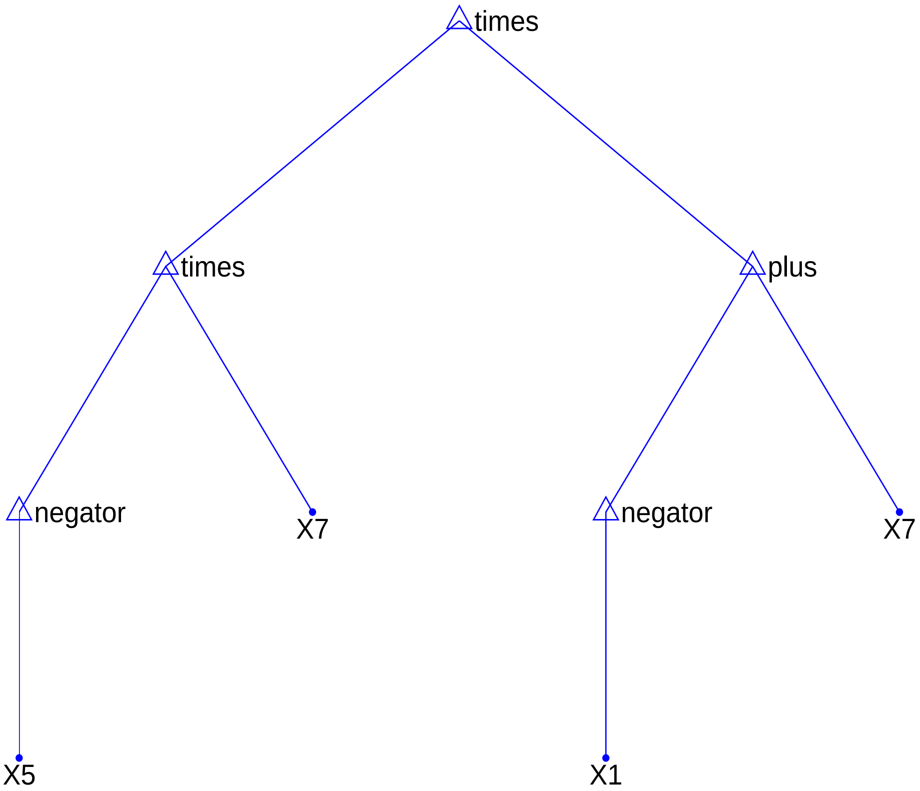


Figure A.7: One of the best trees for 2-class modulation classification for 1024 number of samples at 15 dB SNR

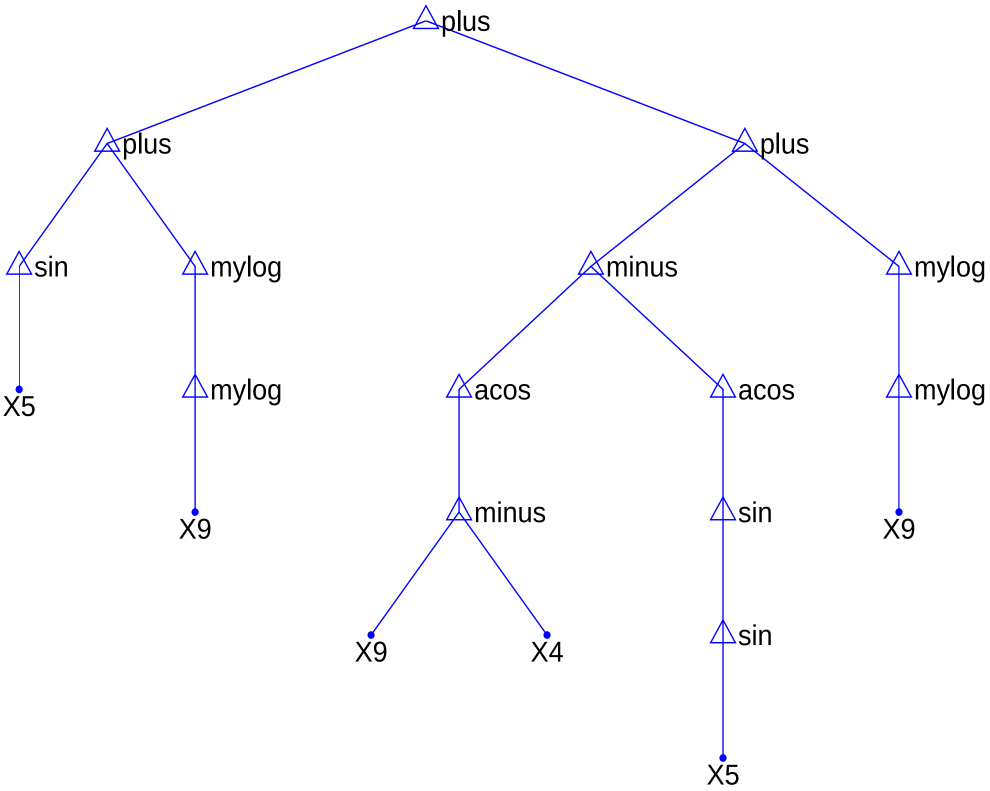


Figure A.8: One of the best trees for 4-class modulation classification for 1024 number of samples at 10 dB SNR

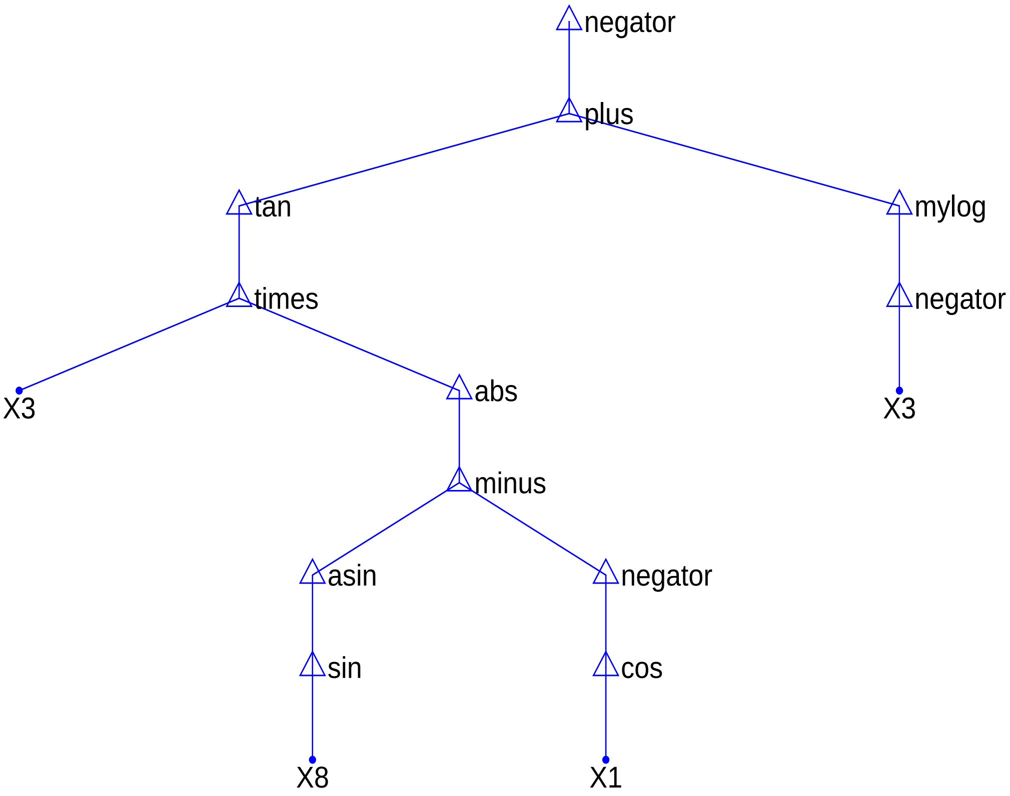


Figure A.9: One of the best trees for 2-class modulation classification for 1024 number of samples at 10 dB SNR

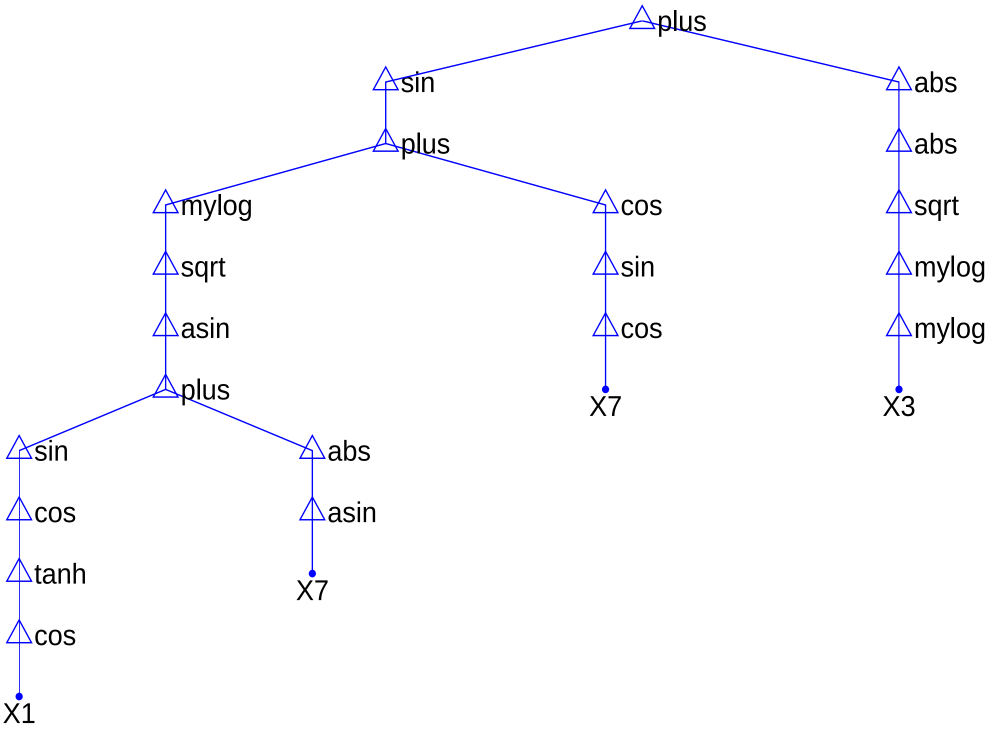


Figure A.10: One of the best trees for 4-class modulation classification for 1024 number of samples at 5 dB SNR

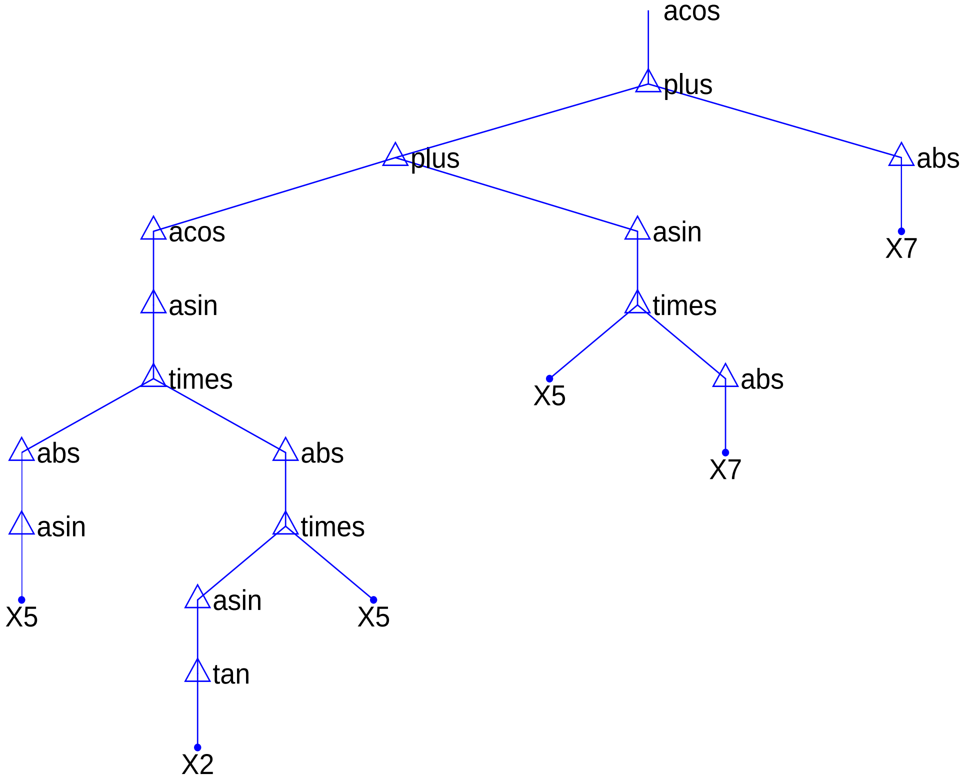


Figure A.11: One of the best trees for 2-class modulation classification for 1024 number of samples at 5 dB SNR

Bibliography

[1] J. Shlens. A tutorial on principal component analysis. Technical report, 2002.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Willey & Sons, Inc., 2000.

[3] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press., 2003.

[4] R. Bellman. *Adaptive Control Processes:* A Guided Tour. Princeton University Press. Princeton, 1961.

[5] C. C. Aggarwal. On randomization, public information and the curse of dimensionality. In *IEEE 23rd International Conference on Data Engineering (ICDE).*, pages 136–145, Apr. 2007.

[6] F. Y. Kuo and I. H. Solan. Lifting the curse of dimensionality. *NOTICES: Notices of the American Mathematical Society*, 52, 2005.

[7] Imola Fodor. A survey of dimension reduction techniques. Technical report, 2002.

[8] Hong Guo. *Feature generation and dimensionality reduction using genetic programming*. PhD thesis, University of Liverpool, 2009.

[9] W. H. Press, S. H. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press., 1992.

[10] I. M. Chakravarti, R. G. Laha, and J. R. Handbook of methods of applied statistics. *John Wiley and Sons.*, I:392–394, 1967.

[11] Yi wei Chen. Combining svms with various feature selection strategies. In *Taiwan University*. Springer-Verlag, 2005.

[12] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[13] H. F. Kaiser. The application of electronic computers to factor analysis. *Educational and Psychological Measurement*, 20:141–151, 1960.

[14] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22:4–37, 2000.

[15] E. Fix and J. Hodges. Discriminatory analysis: Non-parametric discrimination: Consistency properties. Technical report, USAF School of Aciation Medicine, 1951.

[16] V. N. Vapnik. On the uniform convergence of relative frequencies of events to their probabilities. *Soviet Mathematics: Doklady*, pages 915–918, 1968.

[17] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of ESANN, Belgium*, Apr. 1999.

[18] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan, 1994.

[19] R. Rojas. *Neural Networks - A Systematic Introduction.* Springer-Verlag, 1996.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, 1986. Chapter 8.

[21] S. S. Roy and J. J. Shynk. Analysis of the momentum lms algorithm. *IEEE Transaction on Accoustics, Speech and Signal Processing*, 38:2088–2098, 1990.

[22] R. A. Jacobs. Increased rates of convergence through learning rate adapation. *Neural Networks*, 1:295–307, 1988.

[23] M. T. Hagan and M. Menhaj. Training feedforward networks with the marquhardt algorithm. *IEEE Transaction on Neural Networks*, 5(6):989–993, 1998.

[24] H. Demuth and M. Beale. *Neural Networks Toolbox: Users Guide.* The Mathworks Inc., Nactick, 3rd edition.

[25] John R. Koza. Genetic Programming*: On the Programming of Computers by Means of Natural Selection.* MIT Press., 1992.

[26] J. D. Lohn, G. S. Hornby, and D. S. Linden. An evolved antenna for deployment on nasa's space technology 5 mission. In *Genetic Programmetry Theory and Practice II*, volume 8, pages 301–315, 2005.

[27] S. Handley. A new class of function sets for solving sequence problems. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 301–308, 1996.

[28] J. M. Daida, T. F. B. Begey, S. J. Roses, and J. F. Vesecky. Computer-assisted design of image classification algorithms : Dynamic and static fitness evaluations in a scaffolded genetic programming environment. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 279–284, 1996.

[29] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human Competitive Machine Intelligence.* Kulwer Academic Publishers, 2003.

[30] P. G. Espejo, S. Ventura, and F. Herrera. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics : Part C*, 40(2):121–144, Mar. 2010.

[31] J. Eggermont, A. Eiben, and J. V. Hemert. A comparison of genetic programming variants for data classification. In *Advances in Intelligent Data Analysis*, volume 1642, pages 281–290. Springer Berlin / Heidelberg, 1999.

[32] A. L. G. Almanza and E. P. Tsang. Evolving decision rules to predict investment opportunities. *International Journal of Automation and Computing*, 5(1):22–31, 2008.

[33] P. S. Ngan, M. L. Wong, W. Lam, K. S. Leung, and J. C. Y. Cheng. Medical data mining using evolutionary computation. *Artificial Intelligence in Medicine*, 16(1):73–96, 1999.

[34] R. J. Gilbert, J. J. Rowland, and D. B. Kell. Genomic computing: explanatory modelling for functional genomics. pages 551–557, 2000.

[35] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *IEEE Transaction on Evolutionary Computation*, 4:242–258, 2000.

[36] D. P. Muni, N. R. Pal, and J. Das. A novel approach to design classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation*, 8(2):183–196, Apr. 2004.

[37] L. Zhang and A. K. Nandi. Fault classification using genetic programming. *Mechanical Systems and Signal Processing*, 21:1273–1284, 2007.

[38] H. Guo, L. B. Jack, and A. K. Nandi. Feature generation using genetic programming with application to fault classification. *IEEE Transactions Systems, Man, and Cybernetics : Part B*, 35:89–99, Feb. 2005.

[39] H. Guo and A. K. Nandi. Breast cancer diagnosis using genetic programming generated feature. In *IEEE Workshop on Machine Learning for Signal Processing*, pages 215–220, 2005.

[40] L. Zhang, L. B. Jack, , and A. K. Nandi. Extending genetic programming for multi-class classification by combining k-nearest neighbor. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*, pages 349–352, 2005.

[41] M. Zhang, V. B. Ciesielski, and P. Andreae. A domain independent window approach to multi-class object detection using genetic programming. *EURASIP Journal on Applied Signal Processing*, 8:841–859, 2003.

[42] M. Loog and R. H. Umbach. Multi-class linear dimension reduction by generalized fisher criteria. pages 508–517, 2000.

[43] http://diabetes.niddk.nih.gov/dm/pubs/pima/pathfind/pathfind.htm.

[44] http://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes.

[45] B. P. Lathi. *Modern Digital and Analog Communication Systems.* Oxford University Press., 1998.

[46] Perry M. Mistry. Third generation cellular (3g): W-cdma & td-cdma. In *Proceedings of WESCON, Anaheim, CA*, pages 227–231, Sep. 1998.

[47] R. Prasad. *CDMA for wireless personal communications.* Artech House, 1996.

[48] Andrew J. Viterbi. Cdma: Principles of spread spectrum communication. *The Addison-Wesley Wireless Communication Series. Addison-Wesley Pub. Co. Boston*, 1995.

[49] L. Becchetti, F. D. Priscoli, T. Inzerilli, P. Mahonen, and L. Munoz. Enhancing ip service provision over heterogeneous wireless networks: a path toward 4g. *IEEE Communications Magazine*, 39(8):74 –81, Aug. 2001.

[50] M. Dinis and J. Fernandes. Provision of sufficient transmission capacity for broadband mobile multimedia: a step toward 4g. *IEEE Communications Magazine*, 39(8):46 –54, Aug. 2001.

[51] http://standards.ieee.org/getieee802/download/802.11-2012.pdf.

[52] E. E. Azzouz and A. K. Nandi. *Automatic Modulation Recognition of Communication Signals.* Kulwer Academic Press, London, 1996.

[53] N. Alyaoui, H. Ben Hnia, A. Kachouri, and M. Samet. The modulation recognition approaches for software radio. In *2nd International Conference on Signals, Circuits and Systems (SCS).*, pages 1–5, Nov. 2008.

[54] Friedrich K. Jondral. Software-defined radio-basic and evolution to cognitive radio. *EURASIP Journal on Wireless Communication and Networking*, pages 275–283, 2005.

[55] M. Islam, M. A. Hannan, S. A. Samad, and A. Hussain. Modulation technique for software defined radio application. In *Proceedings of the 3rd WSEAS international conference on Circuits, systems, signal and telecommunications*, pages 179–182, 2009.

[56] K. E. Nolan, L. Doyle, P. Mackenzie, and D. O'Mahony. Modulation scheme classification for 4g software radio wireless networks. In *International Conference on Signal Processing, Pattern Recognition, and Applications*, pages 25–31, 2002.

[57] S. Haruyama and R. M. Zaragoza. A software defined radio platform with direct conversion: Soprano. In *IEEE VTS 54th Vehicular Technology Conference (VTC)*, volume 3, pages 1558–1560, 2001.

[58] K. E. Nolan, L. Doyle, D. O'Mahony, and P. Mackenzie. Signal space based adaptive modulation for software radio. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, volume 1, pages 510–515, Mar. 2002.

[59] B. Ramkumar, T. Bose, and M. S. Radenkovic. Combined blind equalization and automatic modulation classification for cognitive radios. In *IEEE*

13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop (DSP/SPE), pages 172–177, Jan. 2009.

[60] B. Ramkumar. Automatic modulation classification for cognitive radios using cyclic feature detection. *IEEE Circuits and Systems Magazine*, 9(2):27–45, 2009.

[61] H. Wu, Y. Wu, J. C. Principe, and X. Wang. Robust switching blind equalizer for wireless cognitive receivers. *IEEE Transactions on Wireless Communications*, 7(5):1461–1465, May. 2008.

[62] Jr. Peyton Z. Peebles. *Probability, Random Variables, and Random Signal Principles.* McGraw-Hill, Inc., 1993.

[63] Alan Stuart and Keith Ord. *Kendall's Advanced Theory of Statistics, Volume 1: Distribution Theory.* Oxford University Press. Inc., 1994.

[64] P. Day and A. K. Nandi. Binary string fitness characterization and comparative partner selection in genetic programming. *IEEE Transaction on Evolutionary Computation*, 12:724–735, 2008.

[65] E. K. Burke, S. Gustafson, G. Kendall, and N. Krasnogor. Is increased diversity in genetic programming useful? an analysis of the the effects on performance. In *Proceedings of the Congress on Evolutionary Computation*, pages 1398–1405, 2003.

[66] C. Ryan. Pygmies and civil servants. *In Advances in Genetic Programming, Chapter 11*, pages 243–1263, 1994.

[67] M. W. Aslam and A. K. Nandi. Detection of diabetes using genetic programming. In *Proceedings of the 18th European Signal Processing Conference (EUSIPCO)*, pages 1184–1188, Aug. 2010.

[68] S. Balakrishnan, R. Narayanaswamy, and I. Paramasivam. An empirical study on the performance of integrated hybrid prediction model on the medical datasets. *International Journal of Computer Applications*, 29:1–6, Sep. 2011.

[69] K. Polat and S. Gunes. An expert system approach based on principal component analysis and adaptive neuro-fuzzy inference system to diagnosis of diabetes disease. *Digital Signal Processing*, 17:702–710, Jul. 2007.

[70] H. Temurtas, N. Yumusak, and T. Feyzullah. A comparative study on diabetes disease diagnosis using neural networks. *Expert Systems with Applications*, 36:8610–8615, May. 2009.

[71] K. Polat, S. Gunes, and A. Arslan. A cascade learning system for classification of diabetes disease: Generalized discriminant analysis and least square support vector machine. *Expert Systems with Applications*, 34(1):214–221, 2008.

[72] I. Gadaras and L. Mikhailov. An interpretable fuzzy rule-based classification methodology for medical diagnosis. *Artificial Intelligence in Medicine*, 47:25–41, 2009.

[73] J. Lin, H. Ke, B. Chien, and W. Yang. Classifier design with feature selection and feature extraction using layered genetic programming. *Expert Systems with Applications*, 34(2):1384–1393, Feb. 2008.

[74] M. Brameier and W. Banzhaf. A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Transactions on Evolutionary Computation*, 5(1):17 –26, Feb. 2001.

[75] C. Lee and M. Wang. A fuzzy expert system for diabetes decision support application. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(1):139–153, Feb. 2011.

[76] R. Kala, H. Vazirani, N. Khanwalkar, and M. Bhattacharya. Evolutionary radial basis function network for classificatory problems. *International Journal of Computer Science and Applicatipons*, 7:34–49, Oct. 2010.

[77] S. Lekkas and L. Mikhailov. Evolving fuzzy medical diagnosis of pima indians diabetes and of dermatological diseases. *Artificial Intelligence in Medicine*, 50:117–126, Oct. 2010.

[78] L. B. Goncalves, M. M. B. R. Vellasco, M. A. C. Pacheco, and F. J. de Souza. Inverted hierarchical neuro-fuzzy bsp system: a novel neuro-fuzzy model for pattern classification and rule extraction in databases. *IEEE Transactions Systems, Man, and Cybernetics : Part C*, 36:236–248, Mar. 2006.

[79] U. Bhowan, M. Johnston, and M. Zhang. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(2):406–421, Apr. 2012.

[80] K. McCarthy, B. Zabar, and G. Weiss. Does cost-sensitive learning beat sampling for classifying rare classes? In *Proceedings of the 1st international workshop on Utility-based data mining*, pages 69–77, 2005.

[81] A. Orriols and E. B. Mansilla. Class imbalance problem in ucs classifier system: fitness adaptation. In *The 2005 IEEE Congress on Evolutionary Computation.*, volume 1, pages 604–611, Sep. 2005.

[82] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[83] J. H. Holmes. Differential negative reinforcement improves classifier system learning rate in two-class problems with unequal base rates. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 635–642, Jul. 1998.

[84] M. J. Pazzani, C. J. Merz, P. M. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 217–225, 1994.

[85] J. Eggermont, A. E. Eiben, and J. I. V. Hemert. Adapting the fitness function in gp for data mining. In *Proceedings of the Second European Workshop on Genetic Programming*, pages 193–202. Springer-Verlag, 1999.

[86] D. Song, M. I. Heywood, and A. N. Z. Heywood. Training genetic programming on half a million patterns: an example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3):225–239, Jun. 2005.

[87] G. Patterson and M. Zhang. Fitness functions in genetic programming for classification with unbalanced data. In *Advances in Artificial Intelligence*, volume 4830, pages 769–775. Springer Berlin / Heidelberg, 2007.

[88] R. Barandela, J. S. Snchez, V. Garca, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36:849–851, 2003.

[89] M. Zhang and W. Smart. Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recognition Letters*, 27:1266–1274, 2006.

[90] E. Zannoni and R. G. Reynolds. Learning to control the program evolution process with cultural algorithms. *Evolutionary Computation*, 5(2):181–211, 1997.

[91] P. D'haeseleer. Context preserving crossover in genetic programming. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, volume 1, pages 256–261, Jun. 1994.

[92] Lee Altenberg. Emergent phenomena in genetic programming. In *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 233–241, Feb. 1994.

[93] P. Day and A. K. Nandi. Robust text-independent speaker verification using genetic programming. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):285 –295, Jan. 2007.

[94] I. A. Maaly and M. E. Obaid. Speech recognition using artificial neural networks. In *2nd International Conference on Information and Communication Technologies (ICTTA)*, volume 1, pages 1246–1247, 2006.

[95] A. Bawakid and M. Oussalah. A semantic-based text classification system. In *IEEE 9th International Conference on Cybernetic Intelligent Systems (CIS).*, pages 1–6, Sep. 2010.

[96] T. Yamada, K. Yamashita, N. Ishii, and K. Iwata. Text classification by combining different distance functions withweights. In *Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 85–90, Jun. 2006.

[97] G. Kou, Y. Peng, Z. Chen, and Y. Shi. Multiple criteria mathematical programming for multi-class classification and application in network intrusion detection. *Information Sciences*, 179(4):371–381, 2009.

[98] I. Benyahia and J. Y. Potvin. Decision support for vehicle dispatching using genetic programming. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28(3):306–314, May. 1998.

[99] Jamie Sherrah. *Automatic Feature Extraction for Pattern Recognition.* PhD thesis, University of Adelaide, South Australia, Jul. 1998.

[100] M. Kotani, S. Ozawa, M. Nakai, and K. Akazawa. Emergence of feature extraction function using genetic programming. In *Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, pages 149–152, Dec. 1999.

[101] T. Keller and L. Hanzo. Prolog to adaptive multicarrier modulation: a convenient framework for time-frequency processing in wireless communications. *Proceedings of the IEEE*, 88(5):609 –610, May. 2000.

[102] M. Luise and R. Reggiannini. Carrier frequency recovery in all-digital modems for burst-mode transmissions. *IEEE Transactions on Communications*, 43(234):1169–1178, 1995.

[103] T. Brown and M. M. Wang. An iterative algorithm for single-frequency estimation. *IEEE Transactions on Signal Processing*, 50(11):2671– 2682, Nov. 2002.

[104] Simon Haykin. *Communication Systems*. John Willey & Sons, Inc., 4th edition, 2001.

[105] Asoke Kumar Nandi. *Blind Estimation Using Higher Order Statistics*. Kulwer Academic Press, London, 1999.

[106] J. J. Shynk and C. K. Chan. Performance surfaces of the constant modulus algorithm based on a conditional gaussian model. *IEEE Transactions on Signal Processing*, 41(5):1965–1969, May. 1993.

[107] R. Jr. Johnson, P. Schniter, T. J. Endres, J. D. Behm, D. R. Brown, and R. A. Casas. Blind equalization using the constant modulus criterion: a review. *Proceedings of the IEEE*, 86(10):1927–1950, Oct. 1998.

[108] J. A. Sills. Maximum-likelihood modulation classification for psk/qam. In *IEEE Military Communications Conference (MILCOM)*, volume 1, pages 217–220, 1999.

[109] W. Wei and J. M. Mendel. Maximum-likelihood classification for digital amplitude-phase modulations. *IEEE Transactions on Communications*, 48(2):189–193, Feb. 2000.

[110] A. Polydoros and K. Kim. On the detection and classification of quadrature digital modulations in broad-band noise. *IEEE Transactions on Communications*, 38(8):1199–1211, Aug. 1990.

[111] C. Huan and A. Polydoros. Likelihood methods for mpsk modulation classification. *IEEE Transactions on Communications*, 43(234):1493–1504, 1995.

[112] L. Hong and K. C. Ho. Classification of bpsk and qpsk signals with unknown signal level using the bayes technique. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages IV–1 – IV–4, May. 2003.

[113] B. F. Beidas and C. L. Weber. Higher-order correlation-based approach to modulation classification of digitally frequency-modulated signals. *IEEE Journal on Selected Areas in Communications*, 13(1):89–101, Jan. 1995.

[114] B. F. Beidas and C. L. Weber. Asynchronous classification of mfsk signals using the higher order correlation domain. *IEEE Transactions on Communications*, 46(4):480–493, Apr. 1998.

[115] A. E. El-Mahdy and N. M. Namazi. Classification of multiple m-ary frequency-shift keying signals over a rayleigh fading channel. *IEEE Transactions on Communications*, 50(6):967–974, Jun. 2002.

[116] P. Panagiotou, A. Anastasopoulos, and A. Polydoros. Likelihood ratio tests for modulation classification. In *IEEE Military Communications Conference (MILCOM)*, volume 2, pages 670–674, 2000.

[117] L. Hong and K. C. Ho. Antenna array likelihood modulation classifier for bpsk and qpsk signals. In *IEEE Military Communications Conference (MILCOM)*, volume 1, pages 647–651, Oct. 2002.

[118] O. A. Dobre and F. Hameed. Likelihood-based algorithms for linear digital modulation classification in fading channels. In *Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1347–1350, May. 2006.

[119] A. Abdi, O. A. Dobre, R. Choudhry, Y. Bar-Ness, and W. Su. Modulation classification in fading channels using antenna arrays. In *IEEE Military Communications Conference (MILCOM).*, volume 1, pages 211–217, 2004.

[120] H. Li, O. A. Dobre, Y. Bar-Ness, and W. Su. Quasi-hybrid likelihood modulation classification with nonlinear carrier frequency offsets estimation using antenna arrays. In *IEEE Military Communications Conference (MILCOM)*, volume 1, pages 570–575, Oct. 2005.

[121] Y. Lin and C. C. J. Kuo. Classification of quadrature amplitude modulated (qam) signals via sequential probability ratio test (sprt). *Signal Processing*, 60(3):263–280, 1997.

[122] Y. Lin and C. C. J. Kuo. Sequential modulation classification of dependent samples. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 2690–2693, May. 1996.

[123] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su. The classification of joint analog and digital modulations. In *IEEE Military Communications Conference (MILCOM)*, volume 5, pages 3010–3015, Oct. 2005.

[124] M. L. D. Wong and A. K. Nandi. Automatic digital modulation recognition using artificial neural network and genetic algorithm. *Signal Processing*, 84(2):351–365, 2004.

[125] M. L. D. Wong and A. K. Nandi. Automatic digital modulation recognition using spectral and statistical features with multi-layer perceptrons. In *Sixth International, Symposium on Signal Processing and its Applications.*, volume 2, pages 390–393, 2001.

[126] A. K. Nandi and E. E. Azzouz. Algorithms for automatic modulation recognition of communication signals. *IEEE Transactions on Communications*, 46(4):431–436, Apr. 1998.

[127] S.-Z. Hsue and S.S. Soliman. Automatic modulation recognition of digitally modulated signals. In *IEEE Military Communications Conference (MILCOM)*, volume 3, pages 645–649, Oct. 1989.

[128] S. Z. Hsue and S. S. Soliman. Automatic modulation classification using zero crossing. *IEE Proceedings on Radar and Signal Processing*, 137(6):459–464, Dec. 1990.

[129] L. Hong and K. C. Ho. Identification of digital modulation types using the wavelet transform. In *IEEE Military Communications Conference (MILCOM)*, volume 1, pages 427–431, 1999.

[130] A. Swami and B. M. Sadler. Hierarchical digital modulation classification using cumulants. *IEEE Transactions on Communications*, 48(3):416–429, Mar. 2000.

[131] Y. Yang and S. S. Soliman. A suboptimal algorithm for modulation classification. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):38–45, Jan. 1997.

[132] Y. Yang and C. Liu. An asymptotic optimal algorithm for modulation classification. *IEEE Communications Letters*, 2(5):117–119, May. 1998.

[133] Y. Yang and S. S. Soliman. Statistical moments based classifier for mpsk signals. In *Global Telecommunications Conference (GLOBECOM)*, volume 1, pages 72–76, Dec. 1991.

[134] S. S. Soliman and S. Z. Hsue. Signal classification using statistical moments. *IEEE Transactions on Communications*, 40(5):908–916, May. 1992.

[135] Y. Yang and S. S. Soliman. An improved moment-based algorithm for signal classification. *Signal Processing*, 43(3):231–244, 1995.

[136] Luo Lichun. Comments on "signal classification using statistical moments". *IEEE Transactions on Communications*, 50(2):195, Feb. 2002.

[137] C. Schreyogg and J. Reichert. Modulation classification of qam schemes using the dft of phase histogram combined with modulus information. In *IEEE Military Communications Conference (MILCOM)*, volume 3, pages 1372–1376, Nov. 1997.

[138] A. Swami, S. Barbarossa, and B. M. Sadler. Blind source separation and signal classification. In *Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1187–1191, 2000.

[139] C. M. Spooner. On the utility of sixth-order cyclic cumulants for rf signal classification. In *Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 890–897, Nov. 2001.

[140] O. A. Dobre, Y. Bar-Ness, and W. Su. Higher-order cyclic cumulants for high order modulation classification. In *IEEE Military Communications Conference (MILCOM)*, volume 1, pages 112–117, Oct. 2003.

[141] O. A. Dobre, Y. Bar-Ness, and W. Su. Robust qam modulation classification algorithm using cyclic cumulants. In *IEEE Wireless Communications and Networking Conference (WCNC)*, volume 2, pages 745–748, Mar. 2004.

[142] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su. Selection combining for modulation recognition in fading channels. In *IEEE Military Communications Conference (MILCOM)*, volume 4, pages 2499–2505, Oct. 2005.

[143] W. Wei and J. M. Mendel. A fuzzy logic method for modulation classification in nonideal environments. *IEEE Transactions on Fuzzy Systems*, 7(3):333–344, Jun. 1999.

[144] B. G. Mobasseri. Constellation shape as a robust signature for digital modulation recognition. In *IEEE Military Communications Conference (MILCOM)*, volume 1, pages 442–446, 1999.

[145] X.Huo and D. Donoho. A simple and robust modulation classification method via counting. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 6, pages 3289–3292, May. 1998.

[146] A. Swami and B. Sadler. Modulation classification via hierarchical agglomerative cluster analysis. In *First IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications*, pages 141–144, Apr. 1997.

[147] S. Taira and E. Murakami. Automatic classification of analogue modulation signals by statistical parameters. In *IEEE Military Communications Conference (MILCOM)*, volume 1, pages 202–207, 1999.

[148] C. Louis and P. Sehier. Automatic modulation recognition with a hierarchical neural network. In *IEEE Military Communications Conference (MILCOM)*, volume 3, pages 713–717, Oct. 1994.

[149] H. Yoshioka, Y. Shirato, I. Toyoda, and M. Umehira. A fast modulation recognition technique using nearest neighbor rules with optimized threshold for modulation classification in rayleigh fading channels. In *the 5th International Symposium on Wireless Personal Multimedia Communications*, volume 3, pages 1049–1052, Oct. 2002.

[150] C. Chung and A. Polydoros. Envelope-based classification schemes for continuous-phase binary frequency-shift-keyed modulations. In *IEEE Military Communications Conference (MILCOM)*, volume 3, pages 796–800, Oct. 1994.

[151] Zhao Zhijin and Shang Junna. A new method for modulation type recognition based on the time frequency representations. In *6th International Conference on Signal Processing*, volume 1, pages 208–211, Aug. 2002.

[152] S. L. Wood, M. J. Ready, and J. R. Treichler. Constellation identification using the radon transform. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1878–1881, Apr. 1988.

[153] S. L. Wood, M. G. Larimore, and J. R. Treichler. Modem constellation identification: a performance comparison of two methods. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1651–1654, Apr. 1990.

[154] Z. Yu, Y. Q. Shi, and W. Su. M-ary frequency shift keying signal classification based-on discrete fourier transform. In *IEEE Military Communications Conference (MILCOM)*, volume 2, pages 1167–1172, Oct. 2003.

[155] L. Shen, S. Li, C. Song, and F. Chen. Automatic modulation classification of mpsk signals using high order cumulants. In *8th International Conference on Signal Processing*, volume 1, 2006.

[156] M. R. Mirarab and M. A. Sobhani. Robust modulation classification for psk /qam/ask using higher-order cumulants. In *Information, Communications Signal Processing, 2007 6th International Conference on*, pages 1–4, Dec. 2007.

[157] S. Xi and H. Wu. Robust automatic modulation classification using cumulant features in the presence of fading channels. In *IEEE Wireless Communications and Networking Conference (WCNC)*, volume 4, pages 2094–2099, Apr. 2006.

[158] W. C. Headley, J. D. Reed, and C. R. C. da Silva. Distributed cyclic spectrum feature-based modulation classification. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1200–1204, 2008.

[159] Liang Hong. Classification of bpsk and qpsk signals in fading environment using the ica technique. In *Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory (SSST)*, pages 491–494, Mar. 2005.

[160] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su. Survey of automatic modulation classification techniques: classical approaches and new trends. *IET Communications*, 1(2):137–156, Apr. 2007.

[161] http://gplab.sourceforge.net/.

[162] M. L. D. Wong, S. K. Ting, and A. K. Nandi. Naïve bayes classification of adaptive broadband wireless modulation schemes with higher order cumulants. In *2nd International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–5, Dec. 2008.

[163] M. Gordan, A. Georgakis, O. Tsatos, G. Oltean, and L. Miclea. Computational complexity reduction of the support vector machine classifiers for image analysis tasks through the use of the discrete cosine transform. In *IEEE International Conference on Automation, Quality and Testing, Robotics*, volume 2, pages 350–355, May. 2006.

[164] R. Cardenes, M. Sanchez, and J. R. Alzola. Computational geometry computation and knn segmentation in itk. *The Insight Journal - MICCAI Open Science Workshop.*, 2006.

[165] T. A. C. Bragatto, G. S. I. Ruas, and M. V. Lamar. Real-time hand postures recognition using low computational complexity artificial neural networks and support vector machines. In *3rd International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 1530–1535, Mar. 2008.

[166] F. Wang and X. Wang. Fast and robust modulation classification via kolmogorov-smirnov test. *IEEE Transactions on Communications*, 58(8):2324–2332, Aug. 2010.

[167] H. Wu, M. Saquib, and Z.Yun. Novel automatic modulation classification using cumulant features for communications via multipath channels. *IEEE Transactions on Wireless Communications*, 7(8):3098–3105, Aug. 2008.

[168] F. Wang, R. Xu, and Z. Zhong. Low complexity kolmogorov-smirnov modulation classification. In *Wireless Communications and Networking Conference (WCNC)*, pages 1607–1611, Mar. 2011.