FEATURE GENERATION AND DIMENSIONALITY REDUCTION USING GENETIC PROGRAMMING

A THESIS SUBMITTED TO THE UNIVERSITY OF LIVERPOOL
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING

2009

By

Hong Guo

Department of Electrical Engineering & Electronics

Contents

List of Tables

List of Figures

Abstract

Generally speaking, a standard pattern recognition system consists of two components, feature extraction and pattern classification. During the feature extraction process, information relevant to the pattern classification is expected to be extracted from the data and prepared as *features* to the inputs of the classifier. Mathematically, this process is viewed as a transformation of the original parameter vector into a smaller feature vector with useful information for the classification task ahead. Usually the original parameter vector has fairly large number of dimensions, in the order of hundreds (in some cases, even thousands), for the coverage all the possible mathematical relations to the problem. However, this poses a significant computation demand to the classifier. The feature space associated with the original vector expands with the increase of dimensions. As a result of that, the complexity of the classifier might increase exponentially. This is famously called the *curse of dimensionality*. Feature extraction is introduced and designed to tackle this problem, by reducing the dimensionality in the data. In addition, it could possibly improve the classification efficiency. In modern pattern recognition systems, the functionality of feature extraction has become very important for its ability to extract discrimination information from the data and filter out unwanted interference. The success of classification is often linked to the quality of the extracted features.

In this research work, endeavours have been made to address the feature extraction issue utilising the strength of evolutionary computation. A genetic programming based feature extraction framework is proposed for the problem. The idea is inspired by the biological evolution process, in which the stronger features survive and weaker ones are washed out. It is believed that for features undergoing such evolution, the discrimination ability will have been optimised under a certain criterion. Therefore, the classification benefits from such optimisation of features.

Compared to conventional feature extraction methods, such as Principal Components Analysis (PCA), Fisher Linear Discriminant Analysis (FLDA), Kernel Principal Components Analysis (KPCA) and Generalised Discriminant Analysis (GDA), this approach undertakes a genetic search in the feature space. The objective of the search is to get closer to the core of the problem. Rather than over-fitting, the genuine characteristic of each pattern is more likely to be identified. Practically, certain termination criterion are set as the job-done points, such as a threshold on the fitness value, the maximum number of generations, etc. They are empirical values set to achieve satisfactory results for specific applications.

In order to cover different scenarios and thoroughly examine the capability of the proposed method, three systems are designed, including a multi-feature extraction system for multi-class problems, a single-feature extraction system for dual-class problems, and a single feature extraction system for multi-class problems. Real data are utilised for the evaluation of the performance of the systems developed. A series of experiments are conducted to evaluate and compare the results obtained using the combination of different feature extraction methods and classification methods. The classifiers involved range from a simple Minimal Distance classifier (MDC) to a sophisticated Multi-Layer Perceptron (MLP)

neural network.

The results demonstrate that the proposed approach is superior to conventional methods for feature extraction for selected applications. The genetic programming system outperforms other systems in terms of classification success. The experimental results are promising. It is believed that this design can be implemented and applied in practical pattern recognition problems as a remarkably cost-effective solution.

Acknowledgements

There are many people whom I am indebted to for their help in the last four years of my PhD studies. First, I would like to thank my supervisor Prof. Asoke K. Nandi, for the prompt and wise feedback and advice to guide my research throughout the period. I am also thankful for my second supervisor, Dr. Xu Zhu for her many suggestions and support during the research.

I would like to acknowledge the financial support of the Overseas Research Studentship Committee, UK, the University of Liverpool and the University of Liverpool Graduates Association. The work would not have been possible without the financial support from them.

I am truly grateful to Dr. Lindsay Jack, who was leading me to the machine learning research topic. The research life was much more easier with his numerous and priceless help and advice. Due thanks must also be made to my colleagues in the lab. The lab become more friendly place with their warm heart and tolerance. Thanks to Sonu, Dennis, Liang, Tingting, Nancy and Sameh.

Finally, my special thanks must go to my husband, Dr. Qing Zhang, for his love, unfailing support and encouragement both in my research and life. I would like to thank my parents for their constant source of support. Without their support and encouragement, these years would not have been possible.

Declaration

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

Glossary of Abbreviations

ANN	Artificial Neural Network
MLP	MultiLayer Perceptron
PCA	Principal Components Analysis
KPCA	Kernel Principal Components Analysis
FLDA	Fisher Linear Discriminant Analysis
A-FLDA	Alternative-FLDA
M-FLDA	Modified-FLDA
GDA	Generalized Discriminant Analysis
GA	Genetic Algorithm
GP	Genetic Programming
BFD	Bearing Fault Detection
MCM	Machine Condition Monitoring
MDC	Minimal Distance Classifier
KNN	K Nearest Neighbor
SVM	Support Vector Machine
FFT	Fast Fourier Transform
CF	Crest Factor
SP	Shock Pulse

Chapter 1

Introduction

1.1 Motivation

The aim of general pattern recognition algorithms is to classify data or patterns into different groups, based upon information from either a priori knowledge or statistical distribution of the data. It has a vast spectrum of applications, with examples such as automatic human voice recognition [2–5], hand-written text classification [6–9], the automatic recognition of human faces from images [10–13], etc.

A typical pattern recognition system consists of three parts. First, one or several sensors/probes are employed to take measurements. Second, a numeric or symbolic calculation is carried out to pick up any information from the raw data. This process is also called feature extraction. In many applications, because of the large number of available features, this process is also called dimensionality reduction, in the sense that the problem can be described in a space with much smaller dimensions compared with the original required number of dimensions. Finally a classifier will do the job to categorise data samples into different groups based on the discrimination information given in the features. Usually this can be visualised as setting boundaries between data points in the feature space with

reduced dimensions.

From the above basic description, it is clear to see that the function of feature extraction in the pattern recognition problems is of vital importance because, as a bridge, it supports the information flow between the raw data collected by the sensors and the classifier. Ideally, useful information or discrimination features can be transferred to the classifier without any distortion. Unrelated redundant information and interference need to be filtered out, so that the classification is purely based on the characteristic of each class without confusion by the noise. In order to get closer to that optimal goal, a large number of feature extraction techniques have been researched and developed in various problem domains, such as Principal Component Analysis (PCA) [14], Semi-Definite Embedding (SDE) [15], etc. Mathematically, they are based mainly on the statistical analysis/mapping of the data.

Genetic Programming (GP), as a machine learning technique has gained a lot of attention in recent years in the domain of pattern recognition, for its non-parametric search capability. It does not require any priori knowledge of statistical distribution of the data. It has certain advantages over classical feature extraction methods in many complex industry problems.

GP is a subset of evolutionary computation, inspired by the biological evolution process. It builds up programs to solve a problem and evolves the programs in the same way as the biology evolves (superior-win-inferior-wash-out). Compared with the traditional trial-and-error manual selection by expert knowledge, GP tries to build up intelligence through the learning process among a large number of data set.

In fact, the motivation of using GP for the feature extraction task is driven by the fact that modern computers become so powerful that even a home PC is capable of carrying out a large amount of computation in a relatively short period

of time. Laboratory computers or workstations, which are specially designed for computing, can be utilised for more sophisticated mathematical calculations and intelligent search for optimal solutions. For a relatively well-defined problem, optimisation among large number of solutions is possible within a reasonably short period of time.

This research has been conducted with the objective to examine, evaluate and possibly evolve the existing GP techniques towards the application of solving complex industrial problems. Such an attempt is believed to be beneficial to both the theoretical and practical analysis.

1.2 Original Contributions

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

- Construction and evaluation of a GP-based approach for multi-feature generation for multi-class classification problems. The results demonstrate a remarkable improvement in the classification performance using GP-extracted features compared with conventional features.

- Evaluation and comparison of the performance of conventional feature extraction methods, a Genetic Algorithm (GA)-based feature selection method and a GP-based feature extraction method using experimental data sets.

- Evaluation of the performance of a Modified-Fisher Linear Discriminant Analysis (FLDA) method using experimental data for two industry problems.

- Comparison of FLDA, Alternative-FLDA and the proposed Modified-FLDA methods in terms of performance; Comparison of the features generated by

GP combined with FLDA, Alternative-FLDA and Modified-FLDA, in terms of the classification performance.

- Construction of a highly efficient learning tool by combining GP with different nonlinear functions to transform useful information into a one dimensional feature space.

- Construction and evaluation of GP-based single feature generation approach for multi-class pattern recognition problems. Investigation/analysis of the performance of different nonlinear feature extraction methods on different data sets.

- Novel design of fitness function for determining the goodness of features, with relatively fast speed and reduced computation complexity.

1.3 Organisation

The major tasks concerned in this research are feature generation and dimensionality reduction based on GP for pattern recognition problems. In order to elaborate the idea and organise the findings in a readable way, this thesis is broken down into six separate chapters, which are listed as follows:

Chapter 1 presents the introduction, motivation and organisation of the thesis.

Chapter 2 provides a brief overview of the definitions and fundamentals of pattern recognition problems. It also provides the basic concept of different feature extraction/selection and classification techniques used in areas that the research is concerned with. In the later chapters, they will be referred numerous times.

Chapter 3 provides a basic description of genetic algorithm and genetic programming paradigm, including the essential elements and the implementation issues related to the design of the program for the task of feature extraction. The concept of evolutionary computation is reviewed with regard to genetic search algorithms, natural selection schemes, genetic operations, and the functionality of fitness measures.

Chapter 4 introduces a GP-based approach for multi-class feature generation from raw vibration data recorded from a rotating machine with six different conditions. The generated features are then used as the inputs to a neural classifier for the identification of six bearing conditions.

Chapter 5 proposes a new linear feature extraction measure, namely Modified Fisher Linear Discriminant Analysis. A Modified Fisher criterion is developed to help GP optimise features. An experimental data set for breast cancer detection and an industrial data set for bearing fault detection are used.

Chapter 6 proposes GP structure for multi-class nonlinear feature extraction based on the Fisher criterion. This produces a nonlinear feature for multi-class recognition by identifying the discrimination information between classes. The comparison demonstrates that only one single feature obtained by a single run of the GP system provides satisfactory results for the problems in test.

Chapter 7 concludes the thesis and summarises the conclusions obtained in each chapter.

1.4 Publications

- Hong Guo, Lindsay B. Jack and Asoke K. Nandi "Feature Generation using Genetic Programming with Application to Fault Classification" IEEE Trans. System, Man and Cybernetics. Part B., vol. 35, no. 1, pp. 89-99, 2005.

- Hong Guo and Asoke K. Nandi "Breast Cancer Diagnosis using Genetic Programming Generated Feature" Pattern Recognition, vol. 39, no. 5, pp. 980-987, 2006.

- Hong Guo, Qing Zhang and Asoke K. Nandi "Feature Generation and Dimensionality Reduction by Genetic Programming based on Fisher Criterion", Expert Systems, Vol. 25, Issue 5, pp. 444-459, 2008.

- Hong Guo, Lindsay B. Jack and Asoke K. Nandi "Automatic Feature Extraction for Bearing Fault Detection using Genetic Programming" 8th International Conference on Vibration in Rotating Machinery, pp. 363-372, Sep.7-9 2004, Swansea U.K.

- Hong Guo, Lindsay B. Jack and Asoke K. Nandi "Automated Feature Extraction using Genetic Programming for Bearing Condition Monitoring" IEEE International Workshop on Machine Learning for Signal Processing, pp. 519-528, Sep.29-Oct.1, 2004, Sco Lums, Brazil.

- Hong Guo and Asoke K. Nandi "Multi-Class nonlinear feature extraction using genetic programming" The 11th World Congress of International Fuzzy Systems Association (IFSA 2005), pp. 1347-1352, July 26-31, 2005, Beijing, China.

- Hong Guo and Asoke K. Nandi "Breast Cancer Diagnosis Using Genetic

Programming Generated Feature" in Proc. 2005 IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2005), pp. 215-220, September 28 - 30, 2005, Mystic, Connecticut, USA.

- Hong Guo, Qing Zhang and Asoke K. Nandi "Feature Generation and Dimensionality Reduction by Genetic Programming" in Proc. 15th European Signal Processing Conference (EUSIPCO 2007), September 3 - 7, 2007, Poznan, Poland.

- Hong Guo, Qing Zhang, Asoke K. Nandi "Breast Cancer Detection using Genetic Programming" in Proc. of the First International Conference on Biomedical Electronics and Devices (BIOSIGNALS 2008), Volume 2, 334-341, January 28-31, 2008, Funchal, Madeira, Portugal.

- Hong Guo, Qing Zhang and Asoke K. Nandi "Bearing Condition Monitoring using Feature Generated by Genetic Programming based on Fisher Criterion", In Proc. 9th International Conference on Vibrations in Rotating Machinery, Sept, 8-10, 2008, Exeter, UK.

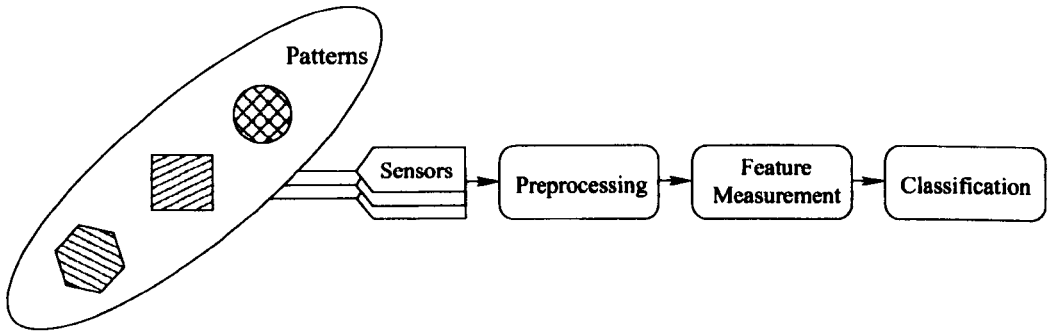
Chapter 2

Preliminaries

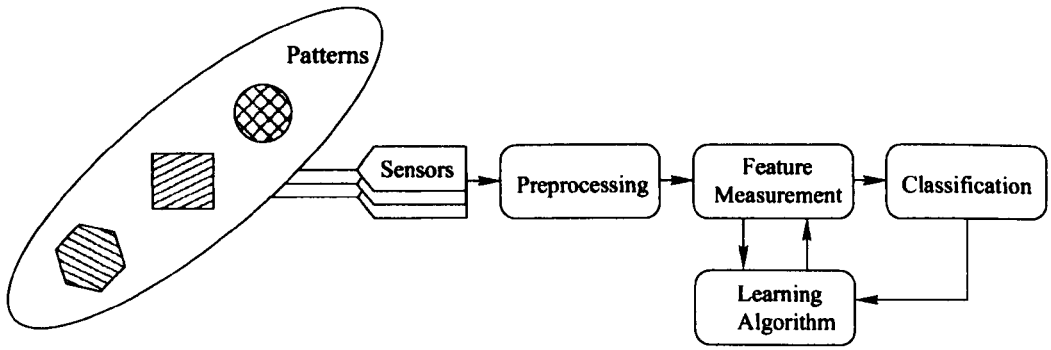
2.1 Pattern Recognition Systems

Pattern recognition is the study of mathematical methods to analyse the observable information from raw data collected by sensors or processors and seek to discover objects of difference from their background and make a classification decision for each pattern. Generally speaking, the goal of pattern recognition is to group the objects into a number of categories. The grouping or classification of objects are important tasks, which cover the studies of statistics, artificial intelligence, computer science and many others. It is popular in many applications, such as machine vision, character recognition, computer-aided diagnosis and speech recognition. As the need for pattern recognition systems increases in industrial applications, advanced technologies have been demanded and researched for such problems.

Typical structures of pattern recognition systems are illustrated in Figure 2.1. There are three major building blocks: preprocessing, feature measurement and classification. Traditionally, these blocks are enough for the pattern recognition task as the number of features is not large and expert knowledge is sufficient for



(a) A traditional pattern recognition system.



(b) A pattern recognition system with machine learning intelligence.

Figure 2.1: Typical structures of modern pattern recognition systems.

the manual selection of features. For sophisticatedly pattern analysis problem with large number of features, a more efficient way to do the feature measurement is to incorporate machine learning intelligence. The learning algorithm block acts as a feedback channel to allow automated evaluation and selection of features based on the classification results. In addition, the learning algorithm might be able to evaluate the features by different criteria without knowing the classification results. We will address this in later chapters.

These blocks are not independent. They are interrelated and dependent upon each other's performance. To improve the overall performance, redesigning may be required in various stages. Usually combined blocks (feature extraction and classification) can be very useful in optimising the system performance.

In recent years, machine learning approaches have been used extensively for the task, such as Genetic Algorithm (GA) for feature selection [16–19] and Artificial Neural Network (ANN)/Support Vector Machines (SVM) [20–24] for pattern classification. The critical tasks of pattern recognition consist of feature extraction and classification. They can be carried out by either supervised or unsupervised learning algorithms. In supervised learning, the training patterns have been assigned to a predefined class. Unsupervised methods discover the relationships between patterns without any priori information of classes. This will be addressed in details in the later sections.

2.2 Dimension Reduction

Dimension reduction is a concept originated from the domain of statistics, where the collection of large amount of data or observations results in information overload. The dimension is the number of variables measured on each observation.

High-dimensional data sets pose many difficulties to data analysis. For example, not all the measured variables are important for understanding the underlying problem. Each individual variable may contain useful information as well as noise or interference. Some variables might be correlated to each other, hence resulting in redundancy. This happens quite often, particularly in data sets with very high dimensionality, in the order of hundreds or even thousands.

In addition, high-dimensional data sets are computationally expensive. The infamous *curse of dimensionality* [25,26] is the best explanation of such problem. Simply put, within a one dimensional space, randomly-sampling 100 points within a unit interval is sufficient to achieve a resolution of at least 0.01. With ten dimensions, the same level of resolution requires a sampling number of 10^{20}. Apparently, the required computation increases exponentially by the increase of dimensions. Without the dimension reduction, the demand for computer power is enormous even for a relatively simple problem.

There are a number of modern techniques for the reduction of dimensions. Generally speaking, there are two main categories, dimension reduction by feature selection and by feature extraction.

2.2.1 Feature Selection

Also known as variable selection, feature reduction, attribute selection or variable subset selection, feature selection approaches intend to find a subset of the original variables. The functionality of feature selection is to help understand the data by identifying important/non-important features and their suitability to the problem in hand. The basic logic is not complicated. Candidate subsets are evaluated and modified iteratively until desired output is achieved or termination criteria are met. A ranking system is used to provide a score value for each candidate. A

search strategy is used to alter the subsets after each evaluation and explore the feature space.

The scoring methods can be broken into two groups, wrappers and filters. Both wrapper type and filter type search through the possible feature space by varying the combination of features. Wrapper-type approaches evaluate the features by running a model, (such as a classifier) and using the model results (such as the classification error) as the score. Filter-type approaches rank each subset based on the output of the filter, such as correlation or mutation values. Other popular metrics include,

- Class separability
  - Error probability
  - Inter-class distance
  - Probabilistic distance
  - Entropy
- Consistency-based feature selection
- Correlation-based feature selection

Apparently, the filter-type approach demands less computation in terms of evaluating the subset. Search algorithm determines the way to alter the subset after each evaluation and the way to explore the feature space. It varies by applications and data size. Typical methods include,

Exhaustive: Complete search of the feature space but computationally expensive hence impractical for large data set.

Best first: Once a satisfactory candidate is identified, the score will never drop below.

Simulated annealing: Provide better diversity over the area /subspace with high-performance features.

Genetic algorithm: Inspired by the biological evolution process, it is a simulation of the real-world surviving scheme by running generations.

Greedy forward selection: One type of the greedy algorithms

Greedy backward elimination: One type of the greedy algorithms

Sequence feature selection: Feature selection according to the sequence information.

2.2.2 Feature Extraction

In the domains of pattern recognition and image processing, feature extraction is such a powerful word that it can actually includes the concept of feature selection and other related techniques. A good definition of feature extraction in the pattern recognition problems is quoted from Wikipedia (2008),

`When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (much data, but not much information) then the input data will be transformed into a reduced representation set of features (also named feature vector). Transforming the input data into the set of features is called features extraction.`

Evidently, a subset of the original features can also be a form of reduced representation. In this thesis, there is neither intention nor necessity to differentiate between these two terms as they are just different usage in different domains. In the domain of statistics, the focus is on how the number of dimensions is reduced,

by a selection or a mapping of the original features. In the domains of pattern recognition and image processing, the term "feature extraction" is used to describe the extraction of information from the original feature space. During the feature extraction process, the extracted features are expected to contain relevant information to the problem in order to perform the desired task using a reduced representation instead of the full size input.

The objective of feature extraction is the same in all domains. That is to simplify the amount of resources required to accurately describe the problem. This is particularly useful when analysing complex data sets, where a large number of variables are involved. Traditionally, it requires a large amount of memory and computation power. The required resources expand exponentially with the increase of number of variables/dimensions. This has been previously described as the *curse of dimensionality.* Even with the significant improvement of computer power and cheaper memory chips in recent years, super computer power and large memory should not be the basis of solving relatively simple problems. From the point of view of improving efficiency, the benefits of feature extraction are invaluable.

Another benefit of feature extraction, in particular for classification problems, is the ability to generalise the problem. With a large number of variables, a classifier algorithm is prone to overfit, which picks up the noise in the data during the training process. The issue of generalisation will be addressed separately later.

For a specific well-defined pattern recognition problem, it is usually the case that the best results are achieved by an expert analysing the data and constructing a set of application-dependent features. However, if no such expert knowledge is available, or a certain level of automatic extraction of features is required to reduce the original large number of variables, general feature extraction techniques is designed to help.

There is a wide spectrum of feature extraction methods available for the task of dimension reduction. Based on different criteria, they can be categorised into different groups, such as linear/nonlinear methods, supervised/unsupervised, parametric/non-parametric, etc. The aim of following sections is to provide a comprehensive literature review to address the wide spectrum of feature extraction methods.

Data modelling or analysis always starts from a simple linear model as linear equations are not difficult to solve and yield analytic solutions, which are desired for most of problems. If linear solution exists, it is always preferred over nonlinear solutions. In fact, more complicated nonlinear models are usually based on localised linear models. Without a good understanding of linear feature extraction methods, one would not be able to fully utilise the functionality of nonlinear methods. To this regard, a large proportion of the discussion will focus on linear methods, including the well-known Principal Component Analysis (PCA), Fisher's Linear Discriminant Analysis (FLDA) and the generalised version of FLDA, namely General Discriminant Analysis (GDA). After that, major nonlinear methods will be addressed, such as the popular kernel trick, which can be applied to the linear methods. The study of general nonlinear methods for feature extraction extends into the domain of manifold learning, which will be addressed briefly. The basic idea of Maximum Variance Unfolding (MVU) and Isometric feature mapping (ISO-MAP) will be described. There is no intensions to address all the techniques in every detail, as they can be found in the relevant literature. The expected outcome of reading this section is to have a general understanding of the mathematical basis of linear/nonlinear feature extraction methods, the strength and limitation of each technique and relation to each other, as well as to other domains of studies.

For the annotation of equations and mathematics, a bold text system is used throughout this thesis. Typically, a bold text small letter indicates vectors and a bold text capital letter indicates matrices. Scalars are marked by normal small or capital letters. For simplicity, the subscripts indicate the index of dimension and the index of observation. The superscript wrapped by curved brackets indicates the class label. For example, $x_{ij}^{(k)}$ is the jth observation of feature i from class k. It is one of the elements of feature vector $\mathbf{x}_j^{(k)}$, which is one of the elements of feature matrix $\mathbf{X}^{(k)}$ for class k. By default, $\mathbf{x}$ is a column vector.

2.3 Linear Feature Extraction Methods

One of the most basic and common mathematical relations is the linear relation, in which one variable is the sum of a constant and the products of the first power of the other variables with a set of constants, as indicated in Equation 2.1,

$$y = \sum_{i=1}^{n} a_i x_i + a_0 \tag{2.1}$$

where n is the number of controlling variables x_i. In a two-dimensional Euclidean space ($n = 1$), linear relation appears as a straight line between x axis and y axis. In a three-dimensional Euclidean space ($n = 2$), if the first axis is x_1 and the second axis is x_2, the value of y constructs a plane. In a multi-dimensional Euclidean space ($n > 2$), the corresponding plane is called a hyperplane. The well-known term "linearly separable" is used to describe a type distribution, in which any two distributions of data points can be separated by a line, a plane or a hyperplane, depending on the number of dimensions involved.

A set of linear equations can be used to solve the unknown variable x_i with

known values of y_i,

$$y_1 = \sum_{i=1}^{n} a_{1i}x_{1i}$$
$$y_2 = \sum_{i=1}^{n} a_{2i}x_{2i}$$
$$\cdots$$
$$y_m = \sum_{i=1}^{n} a_{mi}x_{mi}$$

Note that the constant a_0 in Equation 2.1 can be ignored here because they can be combined into the known variable set $y1, y2, \cdots y_m$. Equivalent matrix form simplifies the notation by,

$$\mathbf{y} = \mathbf{A} \cdot \mathbf{x} \tag{2.2}$$

where $\mathbf{A}$ is a m by n matrix. Depending on the relation between m and n, there could be three types of systems,

- Under-determined system ($m < n$)
- Square system ($m = n$)
- Over-determined system ($m > n$)

The process of calculating vector $\mathbf{y}$ from vector $\mathbf{x}$ is called a linear transformation. Linear feature extraction is basically the linear transformation of original feature vector into a new feature vector.

2.3.1 Principal Components Analysis

Probably the most popular and well-known linear feature extraction method, Principal Components Analysis (PCA) plays a very important role in the foundation of the feature extraction theory. Many of the modern nonlinear feature

extraction methods find their roots in PCA.

The idea is simple: to present the data in a way that the most significant variation can be observed. This is achieved by projecting the original variables into a series of dimensions, where the projected variables are not correlated with each other. The new projected variables are called principal components. If a feature vector set is visualised as a set of sample points in a high-dimensional space, PCA provides a lower-dimensional picture for one to view from the most informative viewpoint.

Since this research is not a theoretical study of statistics, only the methods for computing PCA will be addressed. In-depth study of PCA and further discussions can be found in [14,27]. PCA is commonly calculated by the eigenvalue decomposition of the covariance matrix or singular value decomposition of the data matrix. Given n number of data samples or observations, each of which is a m dimensional feature vector, a m by n matrix $\mathbf{X}$ can be constructed to represent the whole data set. The general steps for conducting PCA using the eigenvalue approach is listed as follows,

1. Before the actual PCA calculation, each data row has to be zero-meaned in order for the PCA to work properly.

2. After that, the corresponding covariance matrix can be calculated as a m by m symmetric matrix. The covariance matrix is commonly computed by the following formula,

$$\mathbf{C_{m\times m}} = \sum_{i=0}^{n-1}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$$

where $\mathbf{x}_i$ is the ith feature vector in the data matrix $\mathbf{X}$ and $\boldsymbol{\mu}$ is the mean vector of the data samples.

3. Then the covariance matrix is solved for m eigenvalues and corresponding linearly independent m eigenvectors.

4. Sort the m eigenvectors based the order of the eigenvalues (highest to lowest).

5. Derive the "new" feature vector by multiplying the original data matrix with a or a set of selected eigenvectors. The product from the eigenvector with the highest eigenvalue will exhibit the largest variation in the resultant feature space.

Theoretically PCA is the linear feature extraction scheme, which transforms the data into a new coordinate such that maximum variation can be observed. It does not require priori knowledge of the probability distribution of data hence it is non-parametric. The answer is also independent of the data probability distribution. However, the non-parametric property is deemed as strength as well as weakness since no distribution knowledge can be incorporated into the feature extraction process hence when the number of data points is limited, it might experience over-fitting problem.

Moreover, PCA has to take two important assumptions for the feature extraction tasks,

Assumption on linearity: Theoretically, it is only suitable for linearly-separable problems. This is improved in its nonlinear version, namely Kernel-PCA.

Assumption on the importance of large variance: It only works for large Signal-to-Noise Ratio (SNR). When the SNR is low, the dynamics from the noise is also viewed as useful variance. In addition, when PCA is used for clustering, there is no way to take into account the class separability. There

is no guarantee that the directions of maximum variance will contain good features for discrimination.

Essentially, PCA performs rotation of coordinates and aligns the transformed axes to the direction of maximum variance. It is a fast and effective method when certain conditions are met. As a general feature extraction method, it has weakness and limitations.

2.3.2 Linear Discriminant Analysis

As discussed, PCA is a popular technique in pattern recognition. However it is not optimised for class separability, which is taken into account by another popular feature extraction method, namely, Linear Discriminant Analysis (LDA).

General purpose discriminant analysis is a concept originated from the domain of statistics. The objective is to determine the discriminant variables between groups or classes. Usually, a group of variables are recorded on a classification problem with known class labels. Based on the recorded data, the task is to find out those variables that discriminate between classes. The analysis is conducted based on the statistics rather than expert knowledge, under the assumption of normal distribution of data.

For example, in order to differentiate between football players and basketball players, a series of measurements can be made, including body height, body weight, age, maximum speed of running, maximum height of jumping, etc. From common knowledge, we can tell that the body height and maximum height of jumping should be the features for differentiation. Discriminant analysis does not use this knowledge. Instead, it conducts statistical tests against each variable and analyse the results based on the statistical test results. As a bad example, age might be picked up as a discriminating feature because a young football team

and an old basketball team are picked up for the analysis (not enough sampling). However, statistically, age is in fact a discriminating variable within these two groups of players.

Linear discriminant analysis (LDA), or Fisher's linear discriminant, is designed to find the linear combinations of data which best separate two or more classes. The resultant combinations are used for the later classification task. It can be viewed as a supervised version of PCA. Instead of evaluating the dynamics/variance in the data, LDA tests the class separability and searches for the best view angle, from which the classes are most separated.

This view angle, or more precisely, transformation vector $\mathbf{w}$, determines the level of class separability from that particular direction. The well-known statistician Ronald Aylmer Fisher [28] defines the class separability as the scalar ratio of between-class scatter over the within-class scatter after the transformation,

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}} \tag{2.3}$$

where the within-class scatter $\mathbf{S}_w$ is calculated by the sum of the covariance matrices $\mathbf{C}^{(k)}$ of all classes, each of which is multiplied with a class prior probability $p^{(k)}$,

$$\mathbf{S_w} = \sum_k p^{(k)} \mathbf{C}^{(k)}$$

The between-class scatter is the covariance matrix of the mean vectors. Therefore, both $\mathbf{S}_w$ and $\mathbf{S}_b$ are m by m matrices. Within-class scatter $\mathbf{S}_w$ sums the variation measured within each class. Between-class scatter $\mathbf{S}_b$ indicates how far apart the classes are by measuring the variation of class means.

Solving Equation 2.3 yields the solution for $\mathbf{w}$. That is, when $\mathbf{w}$ is an eigenvector of $\mathbf{S}_b \mathbf{S}_w^{-1}$, the class separability J equals to the corresponding eigenvalue.

Apparently, the eigenvector corresponding to the largest eigenvalue defines the view angle from which the most significant class separation can be observed.

There are practical limitations in using LDA for real-world problems, where most of time the class means and covariances are not known and can only be estimated. Usually, this is done through the training data sets. Consequently, the optimality of the estimation depends on the size, sampling interval and other related factors of the training data. Also, the estimate of the covariance matrix needs to have full rank to be inverted. In another word, the number of samples must be larger than the number of observation variables. In order to handle the situation where the number of collected samples is less than the number of observations, re-sampling or pseudo inverse techniques are required. Somehow the most serious weakness of LDA probably is the linearity. It is not capable of dealing with nonlinear problems. This obstacle has to be addressed by the kernel tricks, which will be addressed in the later section.

Overall, LDA is an effective and fast method for linear feature extraction for linearly-separable classification problems by promoting the class separability in the reduced subspace. However, it has limitations in practical usage.

2.3.3 General Discriminant Analysis

Linear discriminant analysis can be "generalised" by incorporating "General Linear Models" (GLM). The concept of GLM originated from the use of multiple regression in the analysis of dependent variables. The purpose of multiple regression is to quantify the linear relationship between several independent variables and a dependent variable,

$$y = a_0 + a_1 x_1 + a_2 a_2 + ... + a_m x_m.$$

where x denotes the variables independent to each other and y denotes the variable correlated to $x_1, x_2, \cdots, x_m$. For example, a property market price y could be a linear function of number of bedrooms x_1, total area size x_2, property age x_3, and location factor x_4, etc. An estate agent will be interested in finding out how the actual market price y is associated with those variables, by going through a multiple regression procedure. General linear model analysis, as an extension multiple regression, involves more than one dependent variables. In another word, the dependent variable y becomes a vector and a set of linear equations are set up for the problem.

General Discriminant Analysis (GDA) is called a "general" discriminant analysis because general linear models are applied to the discriminant function analysis problem. Hence it is possible to apply complex models to the set of predictor variables, such as categorical predictor variable and continuous predictor variables. Details of explanation and implementation of GDA can be found in [29,30].

2.4 Nonlinear Feature Extraction Methods

All of the above-mentioned method fall into the categories of linear feature extraction. New features are constructed by linearly combining original data/features. In the feature subspace with reduced dimensions, it is expected to conduct the classification under the assumption that class distributions are linearly-separable. For many practical problems, this is usually not the case. The boundary between clusters are not straight lines, planes, or hyper-planes. Rather, they are curves, surfaces, or hyper-surfaces. The patterns from a particular class are distributed along a *manifold*, which can't be described by a linear function. Using any of the linear methods, such PCA or LDA, one will not be able to find an optimal view angle from which the class separation is sufficient for classification. Certainly,

one will argue that this optimal view angle is subjective. It is rare to find an angle with 100% separation. However, the reason of introducing nonlinear feature extraction methods is to handle those problems with the nature of nonlinearity.

A good example of the nonlinear distribution of patterns is the well-known "Swiss roll", illustrated by Figure 2.2, where different colours indicate different class labels. Visual examination reveals that no planes can be placed to separate the classes. If the distribution can be unrolled and plotted on a flat surface, the classification will become a much easier job. It simply can be solved by setting boundary lines between colours. Therefore, the major issue is to map the original distribution onto a subspace, where linear methods are capable of handling the variability. The process of such mapping is crucial in the success of nonlinear features. The original surface or hyper-surface, along which data is distributed, is called a manifold. The process of finding a such manifold from the data is called *manifold learning*. In fact, the term *manifold learning* is used interchangeably with *nonlinear feature extraction* in literature.

2.4.1 Kernel Principal Component Analysis

In order to handle the nonlinearity, kernel methods have been developed with capability to identify features in a high dimensional space.

The basic idea of KPCA is to map the data from the original feature space $\mathbb{R}^M$ (M is the number of dimensions) onto a high dimensional space $\mathbb{R}^{M'}$, where standard PCA is applied to find the best angles to view the dynamics of data variation. This mapping is done through a nonlinear operator $\mathbf{\Phi}$,

$$\mathbf{x}' = \mathbf{\Phi}(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^M, \mathbf{x}' \in \mathbb{R}^{M'}, (M' \gg M)$$

The resultant feature vector $\mathbf{x}'$ is the nonlinear combination of elements of $\mathbf{x}$.

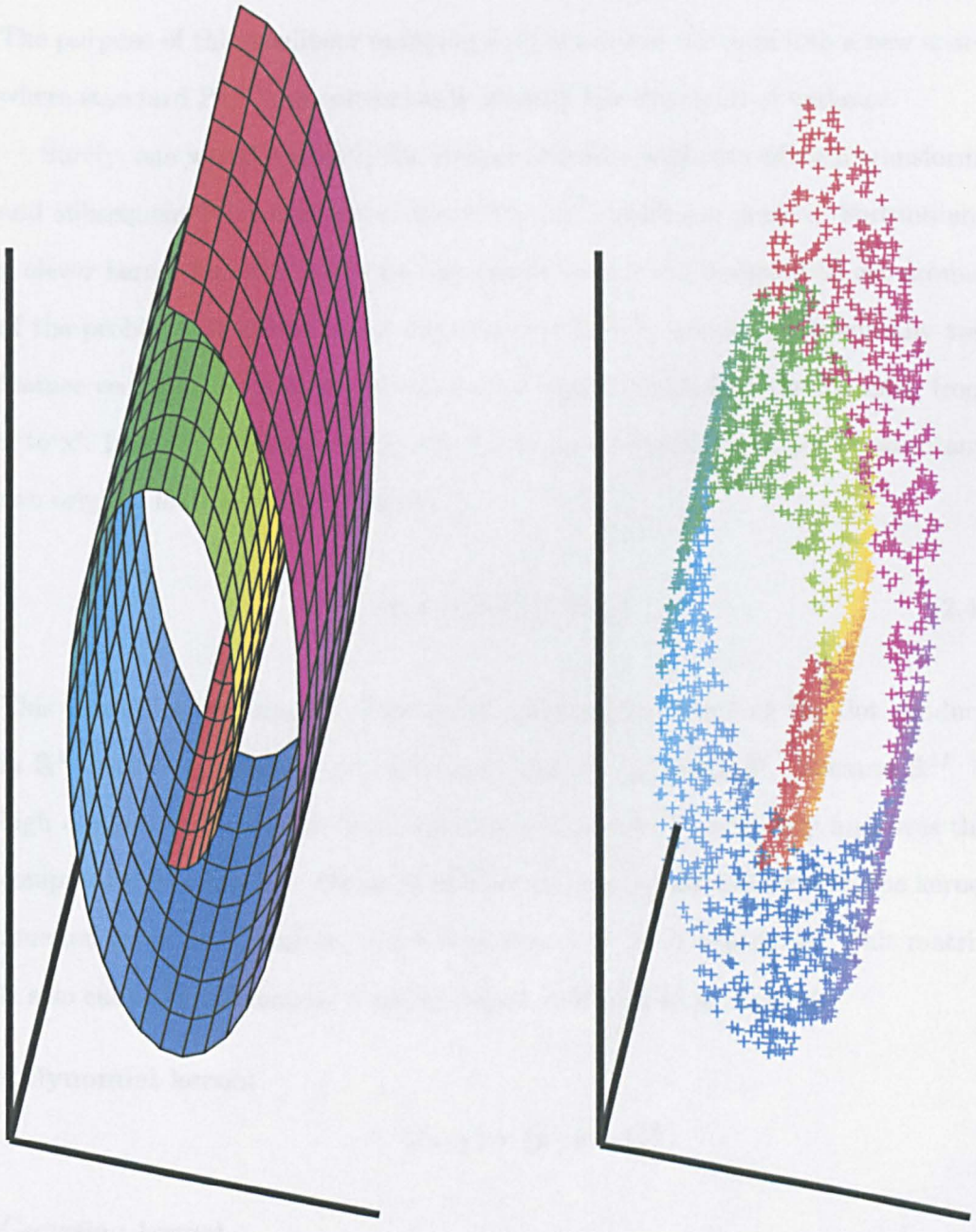


Figure 2.2: The illustration of nonlinear distribution of patterns along a "Swiss Roll".

The number of combinations, in another word the vector size M' is a fairly large number, (could be infinite) in order to cover as many combinations as possible. The purpose of this nonlinear mapping is to transform the data into a new space where standard PCA can conveniently identify the dynamics of variance.

Surely, one would question the computational complexity of such transform, and subsequent PCA analysis of the M' by M' covariance matrix. Fortunately, a clever kernel function based on dot product has been designed to get around of the problem. It allows us to only evaluate the dot product between any two feature vectors. Hence it is not required to explicitly calculate the mapping from $\mathbf{x}$ to $\mathbf{x}'$. Instead, it is only required to know the dot product values between any two original feature vectors $\mathbf{x}$ and $\mathbf{y}$,

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{y}) \tag{2.4}$$

This kernel representation allows us to compute the value of the dot product in $\mathbb{R}^{M'}$ without having to actually carry out the mapping $\mathbf{\Phi}$. Because $\mathbb{R}^{M'}$ is high dimensional, a closed-form expression of $k(\mathbf{x}, \mathbf{y})$ significantly improves the computational efficiency. Given N number of original feature vectors, the kernel function gives us N^2 values, which construct a N by N matrix $\mathbf{K}$. This matrix is also called kernel matrix. Popular kernel functions in use include,

Polynomial kernel

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d$$

Gaussian kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right)$$

Neural Network type kernel

$$k(\mathbf{x}, \mathbf{y}) = \tanh([\mathbf{x} \cdot \mathbf{y}] + b)$$

Besides the above stated kernels there are other varieties of kernel functions, even combinations of kernel functions. Choosing a suitable kernel function for problem at hand is a process to tailor the type of nonlinearity in the data. Details of solving the kernel function and derivation of the principal components can be found in [31]. A general procedure for conducting a kernel-PCA is listed in following steps.

1. Centre the observations in the original feature space by calculating and subtracting the mean of each variable.

2. Compute the N by N kernel matrix $\mathbf{K}$ (N is the number of samples).

3. Obtain the N eigenvectors $\mathbf{v}_i (i = 1, 2, \cdots, N)$ and corresponding eigenvalues λ_i for matrix $\mathbf{K}$.

4. Rearrange the eigenvectors according to the eigenvalues, so that,

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$$

5. Normalise the eigenvectors by requiring

$$\lambda_i (\mathbf{v}_i \cdot \mathbf{v}_i) = 1$$

   The normalised version of eigenvector $\mathbf{v}_i$ is the ith principal component.

6. The projection of an original feature vector $\mathbf{x}$ onto the ith dimension in the

new feature space is obtained by,

$$x_i' = \Phi(\mathbf{x}) \cdot \mathbf{v}_i = \sum_{j=1}^{N} v_{ij} \left[\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x})\right]$$

where v_{ij} is the jth component of vector $\mathbf{v}_i$.

Important properties in using kernel-PCA are

- Compared to linear PCA, kernel PCA extracts features which are more useful for later classification purposes in many real-world problems, where patterns are not linearly separable.
- Kernel PCA can extract up to N features (N is the total number of samples) while linear PCA extracts up to M features (M is the size of the feature vector). Feature extraction is not limited to the original dimension size. As the number of samples increases, the number of extracted features increases.
- To achieve the similar classification results, fewer features are required from kernel-PCA than the linear PCA, resulting in more economic solutions.
- Compared to other nonlinear feature extraction methods, the most significant advantage of kernel PCA is that it does not require nonlinear optimisation, but just the solution of an eigenvalue problem.
- The main drawback of kernel PCA compared to linear PCA is that up to now, there is not a simple method for reconstructing the patterns from the principal components.

2.4.2 Kernel Discriminant Analysis

Analogue to the kernel-PCA, Kernel Discriminant Analysis (KDA) utilises the kernel trick to overcome the computational complexity in applying a nonlinear transformation for the optimal class separability. A general procedure for conducting KDA is described in following steps,

1. Calculate each component of the mean vector of each class in the mapped feature space by,

$$u_i^{(k)} = \sum_{j=1}^{N^{(k)}} k(\mathbf{x}_i, \mathbf{x}_i^{(k)})$$

   Note that the size of the mean vector is the same as the number of samples N.

2. Calculate the covariance of the mean vectors as the between-class scatter $\mathbf{S}_b$, which is a N by N matrix.

3. Calculate the covariance of each class by,

$$\mathbf{C}^{(k)} = \mathbf{L}^{(k)}(\mathbf{I} - \frac{1}{N^{(k)}}) \left[\mathbf{L}^{(k)}\right]^T$$

   where $\mathbf{L}^{(k)}$ is the kernel matrix of each class, with each element given by,

$$L_{ij}^{(k)} = k(\mathbf{x}_i, \mathbf{x}_j^{(k)})$$

   and $\mathbf{I}$ is the identity matrix. Note that $\mathbf{C}^{(k)}$ is a N by N matrix.

4. Calculate the within-class scatter by adding up the covariance matrices, each multiplied with a class prior probability p_k,

$$\mathbf{S}_w = p_k \sum_{k=1}^{c} \mathbf{C}^{(k)}$$

KDA enjoys the same benefit provided by the kernel trick.

2.4.3 Isometric Feature Mapping (ISOMAP)

Another main stream nonlinear dimensionality reduction method is the collection of techniques under the theory of multidimensional scaling (MDS). The basic idea of MDS is to map the original data from the high dimensional space onto a low dimensional space, while preserving the pairwise distances between every two patterns. This is achieved by analysing the similarity/disimilarity of each pair of patterns in the original feature space.

ISOMAP is a nonlinear generalisation of classical MDS. Instead of performing MDS in the original space, ISOMAP conducts the scaling in the geodesic space of the nonlinear data manifold. The geodesic distance is defined as the shortest path between two points along the curved surface of the manifold. The distance is measured as if the surface is flat and approximated by a sequence of short steps between neighbouring sample points. ISOMAP applies MDS to the geodesic rather than straight line distances to find a low-dimensional mapping that preserves the pairwise distances.

General ISOMAP follows three steps,

1. Locate the neighbouring points of each data point in the high-dimensional data space. This is achieved by either identifying the k nearest neighbours or choosing all points within a certain radius.

2. Compute the geodesic pairwise distances between all points by approximation. There are different algorithms [32,33] available for this.

3. Embed the data via MDS so as to preserve these distances.

ISOMAP is a highly efficient nonlinear dimensionality reduction technique. Generally, it can be applied to a broad range of data and applications.

2.4.4 Autoencoder

Another branch of techniques for nonlinear dimensionality reduction is called autoencoder, which is in fact an Artificial Neural Network (ANN) as a learning machine to obtain a compressed representative of the data. The basic theory of an artificial neural network will be addressed in the classifier section later as it is only used as a classifier in this research. In this part, the basic idea of autoencoder will be briefly described without requiring the understanding of ANN.

A typical autoencoder contains three layers. The first layer or input layer contains the same number of neurons as the number of variables in the data. The second layer, or the hidden layer, contains less number of neurons than the first layer as a reduced representative of the input data. The third layer, or the output layer contains the same number of neurons as the number of variables, with the task of reproduce the input. The idea of the whole network is to replicate the input data in the output layer so that the middle layer, which has smaller number of neurons, hence smaller dimensions, acts as a concise representative of the data, because it contains all the information required to reproduce the original data. The results obtained from the middle layer are the features extracted.

As a general neuron network, there are three phases in the feature extraction process, training, validation and testing. The purpose of these three will be addressed in details in the ANN section later. Basically, it requires a dedicated data set just to train the network to make it work, and another data set to generalise the network. After that, the tuned network is ready to work on new data set to produce "features".

This method is often effective for simple or linear data structures. However, when dealing with nonlinear data relationship, usually more than on hidden layers are required for finding solutions. Often, when the errors are back-propagated through the first few layers, they become less significant and hence less effective for the weight adjusting algorithm to tune the neurons. This usually results in slow learning process and poor solutions.

2.5 Inferential Statistics

Statistics is a mathematical science pertaining to the collection, analysis, interpretation or explanation, and presentation of data [34]. As part of the foundation for pattern recognition, the importance of the study of statistics can't be overestimated. General study of statistics can be divided into three categories,

Descriptive statistics: This is the study of methods to describe or summarise a collection of data.

Inferential statistics: In this study, the distribution of patterns in the data is assumed to follow certain mathematical rules. Models are derived to account for randomness and uncertainty in the observation. Inference is drawn upon by analysing the model.

Mathematical statistics: This is the study of theoretical basis of statistics, based on exact probability statements.

Among these studies, inferential statistics is of great interest to our research of pattern recognition problems. The basic concepts related to our research will be addressed in the following context.

2.5.1 Hypothesis Tests

A statistical hypothesis test is a method of making statistical decisions using experimental data [35]. A statistical hypothesis test requires a pair of hypotheses, namely,

A null hypothesis H_0: a hypothesis made to be rejected, in the sense that, in order to prove the otherwise hypothesis, or the so-called alternative hypothesis, is right, one only needs to prove the null hypothesis does not stand. Of course, if the null hypothesis can't be rejected, the test fails to support the alternative hypothesis.

An alternative hypothesis H_a: This is the hypothesis to be proved.

Clearly, the two hypotheses need to be mutually exclusive, which means, if one of them is true, the other must be false, and vice versa. A general procedure for conducting a hypothesis test is explained in following steps,

- The first task is to state the hypotheses, in another word, to define the null hypothesis and the alternative hypothesis in such a way that they are mutually exclusive.

- Decide on the test method to use and the significance level. This part describes how to analyse the data samples and set the conditions for acceptance and rejection. Typically, the test method involves a formula of the test statistic, such as mean, difference between means and chi-square and a sampling distribution of the test statistic, such as normal distribution and T-distribution. Given the test statistic and its sampling distribution, one can derive the probability associated with the test statistic, specifically, the probability of observing a sample statistic at least as extreme as the test statistic by assuming the null hypothesis is true. In inferential statistic

studies, this probability is called a P-value. (The details about the P-value and methods to calculate the P-value will be explained in the next section.) If the P-value is less than the significance level, the null hypothesis is rejected. A significance level is a threshold used for such purpose. The null hypothesis is accepted only if a certain level of significance is reached in the statistical score. Typical values of significance levels are 0.01, 0.05 and 0.1.

- Conduct the test using the method defined in step 2 and derive the test score.
- Decide on acceptance or rejection based on the test score and the significance level.

2.5.2 Student's t-test

Student's t-test is a statistical hypothesis test method developed by William Sealy Gosset, a statistician with a pen name of "student". Basically, t-test is generally applied for deriving the confidence level associated with judgements made from small samples. Popular t-tests include,

- One-sample t-test.
- Slope of a regression line.
- Two-sample t-test with equal sample size and equal variance.
- Two-sample t-test with unequal sample size and equal variance.
- Two sample t-test with unequal sample size and unequal variance.

Here only the last test will be addressed in details. Interested readers are referred to [36,37] for further reading.

Basically, two sample t-test tests the hypothesis about the difference between two means in order to make judgements, for example, whether the two samples belong to the same characteristic group or they have the same criteria. The test is based on following assumptions,

- Random sampling is the sampling method and the samples are independent to each other.
- The population from which the sample is taken, is at least 10 times larger than the sample size.
- Each sample is drawn from a normal or near-normal population.

A typical procedure for conducting a t-test between two means μ_1 and μ_2 is as follows,

1. State the hypotheses.
   - The null hypothesis H_0: $\mu_1 = \mu_2$.
   - The alternative hypothesis H_a: $\mu_1 \neq \mu_2$
2. Choose the formula of the test statistic. For this test, a two-sample t-test formula is chosen,
$$t = \frac{|\mu_1 - \mu_2|}{SE}$$
where SE is the standard error given by,
$$SE = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$$
The standard deviation σ is calculated for each sample.
3. After obtaining the t-test value, the P-value needs to be calculated from the t-distribution, which is actually a probability distribution function of

the test value by random sampling. It is given by,

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

where Γ is the Gamma function given by,

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

Symbol v represents the number degrees of freedom in the two sample mean test, which is calculated by,

$$v = \frac{\left(\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}\right)^2}{\left(\frac{\sigma_1^2}{n_1}\right)^2/(n_1-1)+\left(\frac{\sigma_2^2}{n_2}\right)^2/(n_2-1)}$$

If the result of v is not an integer, it needs to be rounded off to the nearest integer. As mentioned above, the P-value is the probability of observing a sample statistic at least as extreme as the test statistic, assuming the null hypothesis is true. In this case, let us assume the two means are identical. Then the probability of the difference of the two sample means being as large as t is the cumulative of the t-distribution function over two tails,

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

Normally, the P-value is obtained from a pre-calculated t-distribution table, which is widely available from Internet.

2.6 Classification Methods

Classification is the final and mandatory step for pattern recognition problems. The process of classification is to assign the patterns, which have the similar properties to the same class. The simplest and most intuitive classification methods are based on the similarity of well established patterns, which can be classified by using a few prototypes.

General purpose classifiers can be divided into three categories, in terms of the design methodology [30],

- Based on the concept of similarity.
- Based on the probabilistic approach, such as Bayes decision rule, K Nearest Neighbour.
- Based on constructing decision boundaries directly by optimising certain error criterion.

As addressed before, the task of pattern classification has some degree of overlap with the feature extraction process. The aim of a classifier is to discriminate between different patterns by searching for the similarity as well as differences during the training process. The parameters of a classifier are adjusted so that patterns belonging to different groups are discriminated against each other. The information required for such adjustment is something related to the characteristic of each group, specifically, how the patterns distributed within the group and how close the neighbouring groups are. For well-separated two classes, one might just require a simple decision boundary to do the classification job. By either employing more powerful classifier or conducting complicated feature extraction process does not improve the performance significantly. For complex distribution

of data patterns, like commonly encountered in image processing or signal processing tasks, the search for discrimination information is the key to the success of classifications. Either the feature extraction process improves the pattern distribution by presenting the data to the classifier in a way that the class separation is maximised, or the classifier takes on the original data and searches for the subtle boundary between classes. Certainly, this is subjective to the capability of the classifier. No classifier can guarantee a 100% classification success. Generally speaking, if the level of discrimination can be measured, the more discrimination information extracted from the data, the less power is required from the classifier and vice versa.

As a general discussion of classifiers in this section, it is intended not to relate the classifier to the feature extraction functionality. As stand alone systems, different classification methods are described, including Minimum Distance Classifier (MDC), k nearest neighbour (kNN) classifier, Artificial Neural Networks (ANN), and Support Vector Machine (SVM). The aim is to provide a general overview of different systems, the strength and weakness of each method and the applicability.

2.6.1 Minimum Distance Classifier

Minimum distance is probably the simplest classification criterion. Basically, the method finds the centres of the clusters during the training process and measures the distances between these centres to each test sample. The centres are usually calculated by the class means,

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_i \mathbf{x}_i^{(k)}$$

where N_k is the number of training samples for class k and $\mathbf{x}_i^{(k)}$ denotes the ith pattern vector belonging to class k. The classification is determined purely by distance. The distance is defined as a measure of similarity so that the minimum distance indicates the maximum similarity. For example, a test pattern vector $\mathbf{x}_t$ can be classified to group k by satisfying following condition,

$$\min\left[d(\mathbf{x}_t, \boldsymbol{\mu}_i)\right] \qquad (i = 1, 2, \ldots, c)$$

Common functions for measuring distance between two vectors include:

The Euclidean distance Probably the most common and straightforward distance measure defined as,

$$d_M^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y})$$

When the Euclidean distance is used for the minimum distance classification, the boundary between any two classes can be described as a straight line perpendicular to the line connecting the two classes mean vector. Evidently, this equal treatment of distance to all classes causes problem when the sample variances are significantly different between classes.

The Mahalanobis distance The Mahalanobis distance takes into account the scale of the scattering of samples and hence is a "normalised" version of the Euclidean distance,

$$d_W^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T\mathbf{C}^{-1}(\mathbf{x} - \mathbf{y}) \tag{2.5}$$

where $\mathbf{C}$ is the covariance matrix calculated from the training samples from the class to be tested with.

Vector angle The angle between the two vectors can also be used as a measure of distance.

Other distance functions include, the Manhattan distance, which measures distance in a grid geometry, and the Hamming distance, which measures the number of steps required to switch from one class to the other. The latter is mainly used in the study of information theory.

Using the big O notation, the computational complexity is O(N) in the training process, and O(c) in the testing process for each test sample. Symbol N denotes the total number of samples and c denotes the total number of classes. Evidently, minimum distance classifier is a fast and easy-to-implement algorithm.

2.6.2 k Nearest Neighbours

Of all the classification methods, the k-nearest neighbour algorithm is probably among the simplest. It was first introduced by Fix and Hodges in [38].

It classifies a pattern based on the majority vote from its k nearest neighbours with known class labels. If the majority of its k neighbours are from class 1, the pattern will be classified as the same. For example, if among 7 closest neighbours, there are 4 samples belonging to class one and 3 examples belonging to class two, the data point being analysed will be labelled as class one. The nearest neighbours are typically determined by Euclidean distance between the test sample and all the training samples with known class labels. The value of k is always a positive integer, usually a small number.

There is no computation involved in the training process, apart from storing the data and class labels. For each sample test, the Euclidean distance between the test sample and all the training samples need to be calculated. Hence the computational complexity is mainly in the test phase with the order of O(N).

The selection of the value k is another important factor in the design of the classifier. Larger values of k helps reduce the noise effect on the classification. However, it makes boundaries between classes less distinct. A good value of k can be determined by various heuristic techniques, such as cross-validation. When $k = 1$, the pattern is simply assigned to the class of its nearest neighbour [22]. In binary (two class) classification problems, k is chosen to be an odd number to avoids tied votes.

A major disadvantage of using the nearest neighbours as the basis for classification is the possible bias towards the class with larger number of samples. Statistically, a new sample will "meet" these samples more frequently, compared to classes with less number of samples. Hence the new sample is more likely to be labelled as the class with larger number of samples. This tendency is a statistical bias. If the samples are well separated between classes and the distribution are far apart, this will not be an issue any more. However, in real-world problems, class distributions usually have overlaps because of noise and contamination. Obtaining a statistically sound solution is critical to achieving satisfactory performance.

Another problem related to using the algorithm is the sensitivity to noise, irrelevant features, and the distribution scale of patterns. Computing the mutual information between the training data and the training classes can help reduce the effect. In recent years, feature selection and scaling using evolutionary algorithms have been used to overcome such problem with success.

2.6.3 Artificial Neural Networks

Artificial neural network is a mathematical model derived based on the structure of biological neural networks. As explained in the name, it consists of a network of artificial neural interconnected to each other with a task of processing information

from the input nodes and providing decision information to the output nodes. It is an adaptive system that changes structures based on external or internal information flowing through the network during the learning phase.

The application of ANN covers a wide spectrum, including pattern classification [20], data mining (knowledge discovery) [39], signal processing (neural filtering) [40], system modelling [41], and many others. In this thesis, as a pattern classification tool, the basic idea and structure of ANN will be addressed. Popular types of networks for the pattern classification task will be described.

The mathematical model of the ANN can be formulated by the following descriptions. Basically, each node of the network can be described as a weighted sum function of the inputs x_i,

$$y = \psi\left(\sum_i \omega_i x_i\right) \tag{2.6}$$

where ω_i denotes the weighting coefficient for input x_i and $\psi()$ is a predefined decision function, which sends a message to all the connected nodes. Concurrently, the inputs are the outputs of other nodes and the output is one of the inputs of subsequently connected nodes. The interconnection of all the nodes constructs a network.

The ANN model has been studied and researched extensively in both theory and applications. There are a spectrum of well-developed ANN models and learning algorithms. Based on different criteria, ANNs can be divided into different categories. In terms of learning methodology, there are,

Supervised learning: In supervised learning, the target of learning is predefined. The learning algorithm knows exactly what to achieve. For example, for a pattern classification problem, the pattern class labels of the training samples are known. During the training process, the learning algorithm

adjusts the weighting values so that the mismatch between the output and the target decreases. This is done through a cost function, which measures the error between the output and the target. Mathematically, the goal of the learning algorithm is to minimise the cost function.

Unsupervised learning: In an unsupervised learning process, the cost function can be any function of the input data and the network output. For example, if a cost function,

$$\kappa = [x - f(x)]^2$$

is defined to be minimised, the result is, after the learning process, that the network output $f(x)$ equals the mean of the samples x. Of course, the cost function in reality is much more complicated. For the task of clustering, the cost function can be a measure of the class separability. In statistical modelling, it could be the probability of the model given the data.

In terms of information flow, ANNs can be divided into

Feed-forward networks: In this type network, the information flows in only one direction, specifically, from input nodes to the hidden nodes and the output nodes. There is no reverse or loop of information in the network.

Recurrent networks Contrary to feed-forward networks, recurrent networks allows bi-directional flow of information.

There are many variations of ANNs, such Radial Basis Function (RBF) networks [42], Kohonen self-organising networks [43], stochastic neural networks [44], modular neural networks [45], etc. They will not be addressed here for details. In this thesis, a very popular and powerful model of ANNs is described and proposed as the major tool for pattern classification, namely, Multi-Layer Perceptron (MLP).

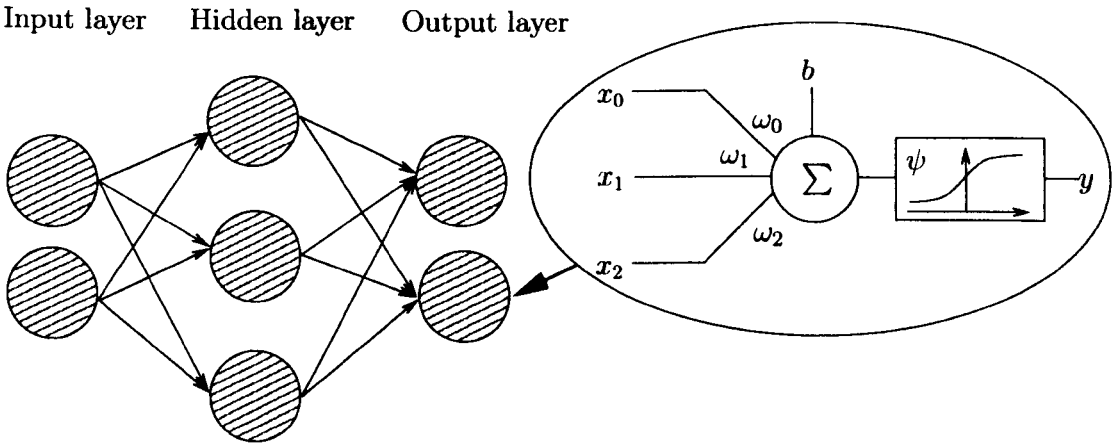


Figure 2.3: A typical structure of multi-layer perceptron network.

The concept of perceptron was originally developed by Frank Rosenblatt [46]. Considering the function ψ in Equation 2.6 to be a simple threshold function defined as

$$\psi = \begin{cases} 1 & \text{if } \sum w_i x_i + b > 0, \\ 0 & \text{else} \end{cases}$$

where w_i is the weighting coefficient for input x_i and b is a constant bias to offset the data. The output y has a binary value depending on whether the sum is larger than zero. This simple mathematical model is called a perceptron, which is capable of separating classes by a hyperplane. In another words, it is a linear classifier, which has the ability to handle linearly separable distributions.

A feed-forward network of perceptrons, which utilise a nonlinear activation function, is called Multi-Layer Perceptron (MLP). MLP is a such popular realisation of ANN for its simplicity in implementation and ability of handling nonlinearity in data. A typical structure of MLP can be illustrated by Figure 2.3.

Instead of using a simple threshold, MLP uses a nonlinear function to convert

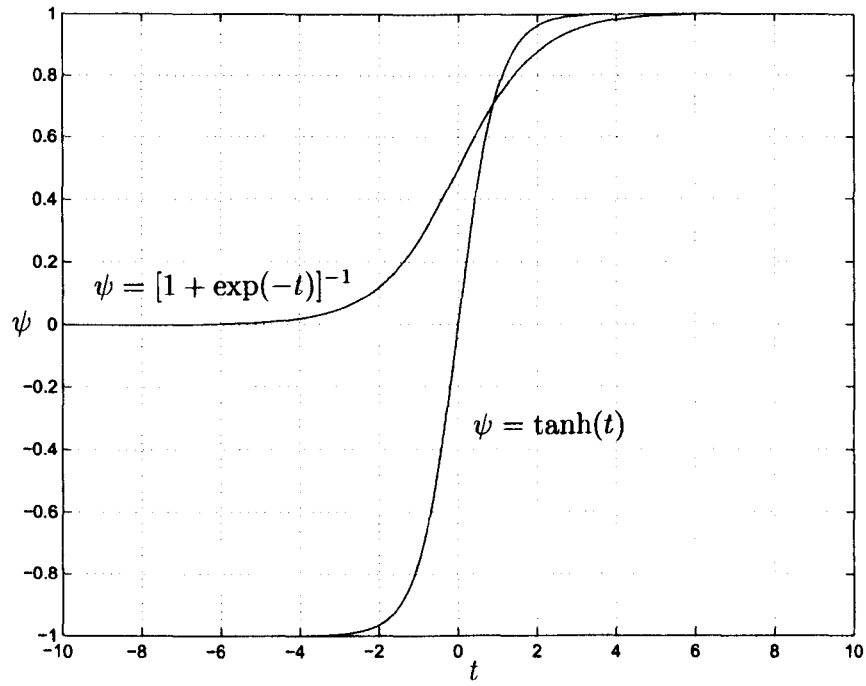


Figure 2.4: Sigmoids activation functions widely used in multi-layer perceptron networks.

the sum into a "decision". In the structure of MLP, this nonlinear function is called "activation function". In most of applications, they are sigmoids, or in another word, "S" shaped, as illustrated in Figure 2.6.3, where t is given by

$$t = \sum x_i \omega_i + b$$

Thanks to the multi-layer structure, the advantage of MLP over a linear perceptron for the pattern classification task, is the ability to handle nonlinearity in the data. As illustrated in Figure 2.3, it usually has three or more than three layers:

Input layer: This layer has the same number of neurons as the input features. Each neuron "reads" the information from each input feature and possibly

applies a weighting as well as a bias. Usually, the neurons in the input layer do not require an activation function, which means the information goes through the first layer as a linear transformation.

Hidden layers: There could be one or more than one hidden layers between the input layer and output layer, depending on the complexity of the problem and nonlinearity in the data. The ability of handling nonlinearity in the data are attributed to these layers. Generally speaking, large number of neurons in the hidden layer or large number of hidden layers will contribute to better nonlinearity handling capability.

Output layer: This layer outputs classification decisions. The number of neurons should be the same as the number of classes. For each class, it is expected that one of the neurons will produce a "high" output while all the rest produce "low".

For a supervised task, back propagation learning algorithm is widely used for training the MLP networks. The basic idea of back propagation technique can be explained in following steps.

1. Given a training sample to the network, compare the network output with the desired output and calculate the error for each neuron.

2. Adjust the weights of each neuron to lower the error.

3. "Back-propagate" the the local error to the previous layer of neurons, by "blaming" them as the cause of the error. The neurons with stronger weights will have greater responsibility for the error.

4. Repeat the previous step until the error has reached the first layer.

5. Repeat step 1 to step 4 until all the training samples are exhausted or the error has been sufficiently small.

Usually, data samples are divided into three groups, no necessarily in same size, but each one should fulfill the requirement in term of number of samples.

Training samples A data set for the learning purpose, which is to adjust the weighting coefficients to minimise the classification error.

Validation samples A data set used to tune the parameters of a classifier, for example, the number of neurons in the hidden layer.

Testing samples A data set used only to assess the performance.

2.6.4 Support Vector Machines

The basic idea of Support Vector Machines (SVM) is pretty straightforward. That is to separate two classes in a n-dimensional space by creating a hyperplane, which has the largest distance to the nearing points in both classes. These points are called support vectors because the hyperplane is only determined by these points rather than the whole data set. By finding this hyperplane, the samples of the two classes are completely separated on the two sides of the hyperplane, and of course, the larger the distance, the better generalisation of the classifier.

Clearly, this idea only works if the two classes are linearly separable, which means a hyperplane does exist between the two classes. The hyperplane is actually an optimised boundary between two classes. For nonlinear problems, the margin has to "softened" to allow some data points to cross the boundary. The "soft" margin tries to split the samples as cleanly as possible, while still maximising the distance to the nearest cleanly-split samples.

Kernel trick can be added to the logic to convert it into a "true" nonlinear machine. The maximum margin is achieved in the transformed space. Because of the nonlinear transformation, this is equivalent to finding an optimal surface separating the two class distribution with the maximum margin to each class samples.

Multi-class SVM can be achieved by converting the single multi-class problem into multiple binary problems, each of which can be solved by a binary classifier. When each binary classifier outputs a high/low value to indicates the preference towards class one or two, a overall decision need to be made based on the outputs of all binary classifiers. There could be two strategies in doing this,

One against all In this logic, the classifier with the maximum separation assigns the class.

Between every two The class label is assigned by counting votes from the classification between every two classes. The class with the maximum number of votes wins out.

2.7 Summary

This chapter has provided an overview of the structure of general pattern recognition systems, by addressing basic concepts, approaches to design feature extractors and classifiers. An overview of feature extraction/selection techniques are provided together with different statistical measures used in the thesis. An introduction to the general principals of linear, nonlinear independent feature extraction methods (PCA, FLDA, kernel-PCA and GDA) are presented. Furthermore the basic principals of four different classifiers: MDC, kNN, ANN and SVM are described. In the following chapters, they will be used as references to

evaluate the proposed feature generation systems.

Chapter 3

Evolutionary-based Algorithms

3.1 Introduction

All of the work contained in this thesis has been developed based on the Genetic Programming (GP) paradigm for the tasks of feature extraction and generation. The major purpose of GP is to optimise feature sets for the later pattern recognition tasks. As already addressed in the previous chapter, the process of feature selection/generation is to derive new features from the original feature set to reduce the computational complexity, and possibly to improve the classification efficiency and accuracy.

Evolutionary computation, as a subfield of artificial intelligence, has received considerable attention in the pattern recognition field, for the power of non-parametric searching and optimisation [47–50]. As a subset of evolutionary computation, evolutionary algorithms have a mechanism inspired by the biological evolution process, such as reproduction, mutation, recombination, natural selection and survival of the elitists [51,52]. Within a population, each candidate is a solution to the optimisation problem and the fitness is measured by a *fair* measure in order for the most competitive candidate to survive the natural selection.

This process is repeated in each generation until the satisfactory overcome or the maximum number of generations has been reached.

Genetic Programming (GP) is an evolutionary algorithm based methodology that finds computer programs to perform a user-defined task [53–55]. It is also a machine learning technique for the optimisation of computer programs according to a fitness measure, which determines the program's ability to perform [56, 57]. Due to the fact that GP is computationally intensive, it was mainly used to solve relatively simple problems in the 1990s. Nowadays, thanks to the exponential growth of CPU power, GP plays more and more important roles in quantum computing, electronic design, game playing, sorting, searching and many other areas. It has produced many novel and encouraging results. Although theoretically development of GP has been very difficult due to the non-parametric model, after a series of breakthroughs in the early 2000s, it has been possible to build exact probabilistic models of GP [58–60].

In recent years, applications of evolutionary learning algorithms for pattern recognition problems have become increasingly common. Evolutionary strategies [61], Evolutionary Programming [62], Genetic Algorithms (GAs) [16, 63], and GP [56,64] have been used to solve complex problems. The evolutionary learning algorithms have been used for feature selection/generation tasks to reduce the computational complexity and improve classificatoin accuracy. However, in most of the applications, GP was solely employed as a classifier based on manually developed features [65–68].

In this research work, we are trying to explore the capability of GP in feature extraction. Instead of the search for optimal classifiers, it is believed that GP is also capable of finding discrimination information in the data. The discrimination information can be extracted in a form of feature vectors. These feature vectors are further used for the classification task by a dedicated classifier. It is not

the first attempt for such task. Efforts have been made to use GP as a feature extractor [64, 69–71]. However, in most applications, a wrapper-type approach is adopted, which has considerable computational complexity. In order for a satisfactory solution to emerge in the evolution process, computer power and running time are still important factors. This compromises the optimisation capability of the evolutionary algorithm.

We try to address the same problem using a GP-based approach, but with a different design of fitness measure. Instead of using classification success as the fitness score, a discriminant-analysis based fitness measure is conducted for the solution evaluation. Exhaustive training, validating and testing of a classifier is not required in such a design. Due to the reduced computation time for the fitness value, GP is able to evolve faster and more efficiently for better solutions to emerge. Surely the ultimate goal of the feature extraction is to assist the classification. Hence the optimal criterion for measuring is the classification success. However, from a practical point of view, we would argue that, computation-intensive algorithms, such as GP, will benefit greatly by reducing the computation time and at same time maintaining reasonable accuracy. It is believed to be more efficient than searching for the perfect solution during a usually slow process. Following this idea, three systems are designed and tested for different pattern recognition problems in later chapters. This chapter will focus on the theory explanation and the design of the common part, which all systems share.

This chapter is organised as follows. Section 3.2 briefly explains the basic concepts and terms used in the study of biological evolution, for the purpose of gaining a basic understanding of the theory behind the evolutionary computation. Section 3.3 starts to talk about the principle of genetic search and general steps for conducting a genetic search. The basic structure of genetic algorithms will be

explained in Section 3.4, as a preparation for the later design and implementation of the proposed GP feature extractor in Section 3.5. Section 3.6 addresses some issues related to the implementation and testing of GP. This chapter is then summarised in Section 3.7.

3.2 Basic Concepts

The idea of simulating the evolution process for the benefit of computing is inspired by the biological natural selection process. It is observed that all species in the natural world have evolved over time and improved their certain ability in dealing with particular problems. Before the discussion of genetic based computation, it will be useful to briefly go through the basic concepts in the study of biological evolution. They are elementary to the understanding of the evolution process. Without the understanding of these concepts, one will not be able to simulate the natural selection process for real-world problems. In addition, gaining a good understanding of these concepts and finding out the underlying theory would be beneficial to the design of evolution-based computer systems. Certainly, only the content related to the analysis of evolutionary computation will be addressed.

Gene: gene is the basic unit of heredity. It specifies a trait. Many genes can coexist in a chromosome and pass on to the next offspring. One gene determines one particular attribute of an individual, such as appearance, ability to do something or a disease. A gene can be passed on unaltered for many generations.

Gene pool: A gene pool is the set of all genes in a population.

Chromosome: In biology, a chromosome is an organised structure of DNA and

protein in cells. It contains a lot of genes. It is the unique identification of an individual within a population.

Mutation: In biology, mutations are changes to the gene sequence. Mutations can be caused by copying errors, by exposure to radiation, chemical agent, or virus, etc.

Population: Population consists of a number of individuals, each of which has a unique chromosome hence has different characteristics in terms of competence.

Generation: Generation is a term used to describe the ongoing evolution process. In fact, it is a result of the evolution, as only part of the generation will survive to the next generation. The next generation is created based on the survivors from the previous generation.

Evolution: Evolution is a change in the gene pool of a population over time.

These concepts will be further explored in the following discussions about genetic search and evolutionary computation.

3.3 Genetic Search

Before discussing the power of genetic search, it is advisable to briefly explain what is a general search algorithm and what strategies involved in a search. In the domain of computer science, a search algorithm is a technique to locate a solution for a predefined problem, by evaluating a number of possible solutions. The set of all possible solutions to a problem is called the search space. In term of search strategy, there could be two approaches, namely uninformed search and informed search. Uninformed search or naive search, uses the simplest method by

going through the entire search space. Informed search algorithms use heuristic functions to apply knowledge about the structure of the search space to try to reduce the amount of time spent in the search.

In terms of scope of the search, there could be local or global search methods. The former only conducts the search within a local optima region. The search is stopped once a local optima is reached. Usually, this approach converges quickly after a few iterations. However, the solution is highly dependent upon the starting point. Different starting points may lead to different answers. Hence a global solution is better for most real-world problems. Compared to local search methods, global search is relatively slow and demands more computation power to evaluate the topology of the search space.

In most applications, genetic algorithms can be categorised as global search heuristics. The search is conducted through a simulation of biological evolution processes. New generation is created based on the survivors of the previous generation. Hence, as long as the evolution keeps going, the solution will get close to the optima.

The genetic search is an iterative process within populations of possible solutions. The search is carried out iteratively in the sense that the operation will be repeated by many cycles with the desired output results being improved steadily. Candidate solutions are presented in an encoded form, which is defined by size and shape in terms of the specification of the problem in hand. Estimation of encoded solutions is an issue involving rating the performance of each individual within the population. The steps for conducting a genetic search are,

1. An initial population with a set of randomly generated individuals is created. Each individual represents a possible solution to the problem.

2. Each individual in the population is ranked in terms of its performance to

the problem. It is evaluated by a component of GAs/GP, namely **Fitness Function**. The fitness function assigns a fitness value to each individual.

3. **Selection** is the process of producing an intermediate population by applying pressure for surviving. This pressure is in a form of fitness landscape. In order to explain how this landscape works, it might be a good idea to take some examples from the natural evolution process. One of the examples will be: deer with longer necks have a better chance to survive in an environment where leaves are usually above a certain height. The pressure is applied by setting a high standard for surviving. Of course, there are advantages and disadvantages. A too high standard results few survivors, hence reduces the diversity of the population. Low diversity limits the ability of the population for creating new genes and narrow the search angle. The search may ends up at a local optima. A low standard maintains good diversity. However, it allows too many weak genes to survive through the generation. Statistically, the elite candidates are not given enough opportunity to marry and produce offsprings, which carry parts of the good genes inherited from both parents. The evolution progress may be very slow and time consuming. Therefore, it can be said that a properly set standard is critical to the success of the genetic search.

   Another result of the natural selection is adaptation. Populations starting farming rather than hunting develop enzymes that can digest grains. This natural phenomenon cannot be explained by the survival-of-fittest theory as there is not surviving pressure. Gene adapts itself to suit the living or working conditions. As the biological theory and reasoning behind this is still under research, it will be really difficult to simulate such phenomenon without understanding how it happens.

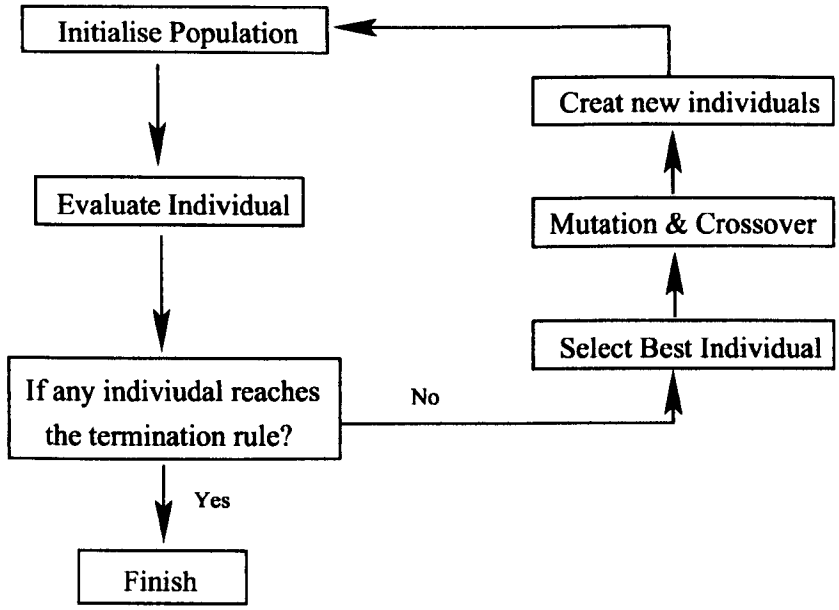


Figure 3.1: The basic structure of genetic search.

4. In the new generation, the survivors choose mates and produce offspring. The offspring share similarities but are not direct copies of the parents. Children inherit traits from parents but they may vary in their physical properties and behaviours. Based on the survivors, a new population is generated under common genetic operations, such as mutation or crossover.

5. Now the new population will go through the natural selection process again and produce offspring for the next generation, until the stopping criteria are met. The criteria could be a "good-enough" fitness value, a certain number of generations having been evolved or no significant improvement being made.

Flowchart in Figure 3.1 illustrates the genetic evolution search process.

3.4 Genetic Algorithms

Genetic algorithms (GAs) started to gain popularity after the developments by Holland [72] and his Ph.D student DeJong [73] in 1975. Most of the current evolutionary algorithms are derived from their work. GAs are now commonly seen as a generic stochastic search algorithm as they are able to find optimal solutions in high dimensional space. In recent years, many different types of genetic algorithms have been developed with the ability to control the selection operations. Genetic algorithms have been used to construct feature selection systems in [1, 16, 19, 63, 74].

In this thesis, GAs are used for the construction of a feature selection system. As mentioned in the previous chapter, feature selection is a process to reduce the dimensionality in the data by selecting only part of the features as a "good-enough" representation of the original data. In this research, a wrapper-type feature selector is constructed under the GA structure. Comprehensive explanation of GA can be found in [75, 76]. A brief description of Genetic Algorithm is given below.

As a method of evolutionary computation, GA simulates the evolution process using a binary string chromosome.

3.4.1 Chromosome

In GAs, a chromosome is defined as a set of parameters of a proposed solution to the problem in hand. The task of the GA is find the optimal solution for the problem by altering the chromosome during the evolution process. The chromosome is often represented as a simple string, although a wide variety of other data structures are also available. The encoding of the chromosome varies by the requirement of applications. Possibilities are, real or floating point numbers in

a binary string chromosome, string chromosome or tree chromosome, the variable length or fixed length chromosome, etc. However the chromosome encoding should have the capability of including full representation of all the possibilities and allowing the creation of new chromosomes that can be evaluated during the process.

Binary encoding is the most extensively used chromosome in evolutionary computation. A string with D binary digits can be used. Each binary digit denotes a feature with 1 representing the presence and 0 representing absence. For instance, chromosome 01001001 indicates that the second, fifth and eighth features are selected.

3.4.2 Fitness Function

A fitness function is an objective function used in the evolution process to quantify the optimality of each chromosome. By doing that, one particular chromosome may be ranked higher than others. Hence it is allowed to survive to the next generation for breeding. The employment of fitness function is critical in the design and implementation of a genetic algorithm. The selection is conducted mainly based on the fitness measure. An ideal fitness function correlates closely with the algorithm's goal, and yet may be calculated with a reasonably fast speed. Speed of execution is a very important factor in the evolutionary computation, as typically a reasonable solution only emerge after the evolution of a large number of generations.

In some designs, computation speed and size of the feature are considered to be combined in the score to evaluate a chromosome. This indicates that the desired quality in the solution is the overall performance rather than the fitness itself.

3.4.3 Selection Algorithm

In conventional GA search, the entire populations are replaced at one time. This is different to the natural world, since the birth and death of individual are a continuous process. A steady state GA [77] is applied in the research to determine the selection of individual in each population, rather than replacing the entire population.

Various selection functions, such as uniform, tournament, ranking, proportional, truncation selections, can be used in genetic algorithms to determine which individual in current population can be chosen to the next generation for conducting the operations of crossover and mutation [78]. For standard selection functions, parameters need to be carefully set to control the performance of selection on the individuals. In some cases, it can be only done through a process of experimentation. Many selection functions of GA systems have been developed by maintaining a certain degree of diversity.

Fitness-proportionate selection: Fitness-proportionate selection is also well-known as Roulette Wheel selection. The probability of an individual being selected is proportional to its fitness value. This simple method often leads to premature convergence.

Rank-based selection: A fitness threshold is calculated by a ranking equation. Then fitness-proportionate selection is conducted based on the threshold. The ranking formula can be linear scaling, exponential scaling or any other scaling. Rank-based selection preserves diversity and has relatively low selection pressure.

Tournament selection: The fittest individuals in a tournament group go into intermediate population or the new population. It is less computationally

intensive as not all the individuals are evaluated.

Uniform selection: For the measurement of fitness on each candidate, a random number f is uniformly generated within the interval $[f_{min}, f_{max}]$, where f_{min} and f_{max} are the maximum and minimum fitness values in the current population. The individual with the fitness closest to the number f is selected for the next generation. Because the probability of selecting the random fitness number is equal, the minorities will have better chance to survive through to the next generation. This design preserves a great deal of diversity with a cost of slow convergence.

Elitism: A certain number of fittest individuals are copied into intermediate or next-generation population. The advantage of using elitism selection scheme is that once a good solution is found, it will not be lost until better solutions are found.

3.4.4 Genetic Operators

Crossover and mutation are two main genetic operators used in GAs. They are fundamental to the operation of GA. These operations alter one or more gene values in the chromosome to produce offspring. Genetic operation is one of the most important research topics in the study of genetic algorithms. Different methods can alter the performance of the GA dramatically. Crossover and mutation are two most common operations.

Crossover

The crossover operation is to create new members based on the combination of the elements of two parent chromosomes. The newly created offspring will inherit certain characteristics of both parents.

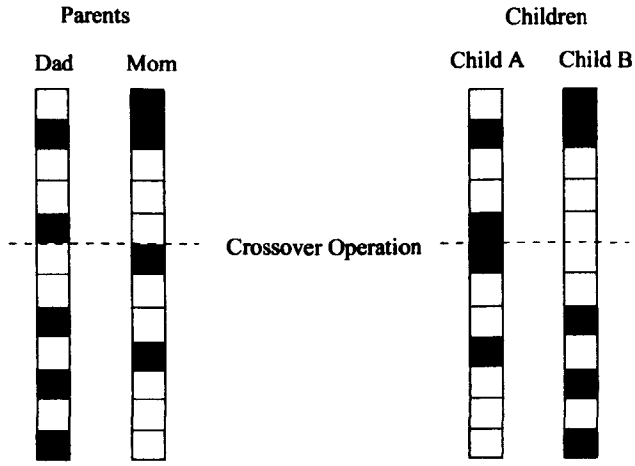


Figure 3.2: The crossover process of genetic algorithm

The operation of crossover can be illustrated by Figure 3.2. For a fixed length, the crossover is performed by randomly selecting a same cutting point in each of two parent chromosomes, and swapping the portions. In addition, a cutting point may be broken in different places and recombined to perform the crossover operation on variable length parents. The probability of crossover P_c is used to control the occurrence of crossover operation in given chromosomes.

Mutation

Mutation alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible. Figure 3.3 illustrates the operation of mutation.

Mutation occurs during the evolution process based on a pre-defined mutation probability P_m, which is usually a fairly low value ($P_m = 0.01$) in GA problems. Otherwise, the search will turn into a primitive random search [79].

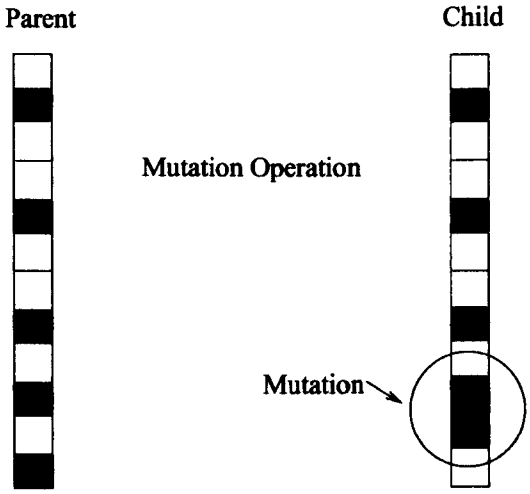


Figure 3.3: The mutation process of genetic algorithm

3.5 Genetic Programming

Analogue to the GAs, GP is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a user-defined task. It differs from GAs by that it uses a tree representation to emulate computer programs. Each individual in the GP generation is actually a computer program that can be used for a computation task. GP, as a machine learning method, is used to optimise a population of computer programs under a fitness landscape, which determines the program's ability to perform.

In the tree structure, each node is a mathematical operator/function, which obtains the input values from the sub-branches, calculates the result, and provides an output to the upper-level node. This tree structure is a typical computer routine/subroutine structure. Thus any programming language that uses tree structures can be evolved by GP.

Originally developed by Koza [56] in 1990s, GP was considered to be computationally expensive and mainly used to solve simple problems. More recently, with the exponential growth of CPU power, GP has attracted a lot of attention for its

flexility and powerful search competence, compared to other artificial intelligence systems. Many novel results have been achieved with outstanding performance in many areas, such as stock market modelling [80], circuit design [81], game playing [82], etc. In fact, in Koza's original report [56], it has been demonstrated that GP has the potential for solving complex optimisation problems, by either evolving decision trees for input classification or evolving a play strategy.

A good example of the application of GP is data classification/partitioning problem [66,83–89]. Unlike the *divide and conquer* approach of standard machine learning algorithms, GP algorithm concentrates on finding the right combination of attributes and decision tree shape. As a comparison, greedy algorithms like C4.5 [90,91] are aimed at locally optimising a decision tree during construction. Generally speaking, GP performs a more global search through the space of possible solutions.

GP, as a form of evolutionary computation and an extension of genetic algorithms, is proposed as the paradigm for the construction of a feature extraction/generation system in this research.

In the GP paradigm, individual is implemented in a tree form, which consists of internal nodes and leaf nodes. The leaf is defined as a **Primitive Terminator** and the internal node is known as a **Primitive Operator**. The tree-based structure determines that GP has the ability to create new solutions rather than the pure selection of features using GA. As the representation changes, the structure of GP is changed by the selection methods, the genetic operators, the fitness function and the primitive sets. Although the selection methods used in GP are the same as those used in GA, the genetic operators need to be redefined for conducting the operations on the tree-based chromosome.

If treated as a black box, this GP-based system extracts features from the original feature set with a goal to improve the classification success rate. It has

the following advantages compared to classical feature extraction methods and GAs,

- Features of each generation are created automatically, thus avoiding human influence or bias.
- GP has the ability to extract features from the original feature set without prior knowledge of the probabilistic distribution of the data.
- GP exhibits a certain level of artificial intelligence by deciding whether to perform feature extraction or feature selection during the evolutionary process. Whereas in GAs, the optimal solution is only "chosen" from a large number of candidates.

Since both GA and GP are utilised in this research, it will be useful to compare these two methods. The major difference between GP and GA approaches is the representation of chromosome. GA optimises the solution by "selecting" features while GP tries to "generate" features by "integrating" original feature set. The results of this integration is the extraction of discrimination information from the data for the classification task.

Generally, the length of string presentation is fixed in GA. The length of chromosome in GP can vary based on the problems. With a GA based solution, the basic form of the solution is predefined. The task of GA is to optimise parameters of the solution, but not the actual structure of the solution. GP by comparison has control over both the structure and the parameters of the solution to the problem.

The following sections will provide a description of the GP-based feature extraction system designed in this research.

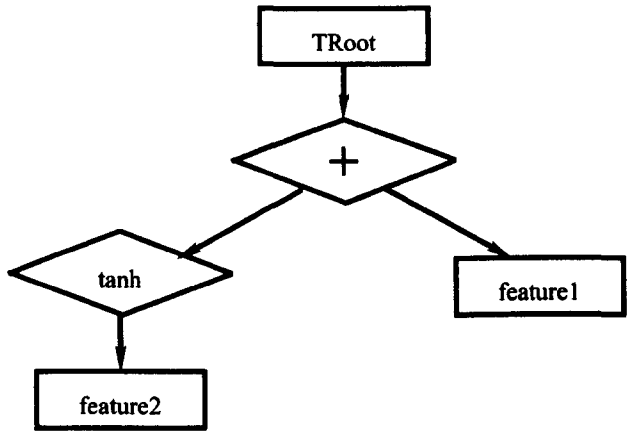


Figure 3.4: Tree representation

3.5.1 The Representation of Chromosome

Although non-tree representations were suggested in some literature and had successful stories [66], in our design, the conventional tree structure is used to construct the chromosome as it can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. In addition, it has stronger ability to handle the nonlinearity in the data by carefully choosing the mathematical operators.

In fact, the mathematical operation carried out within the chromosome can be formulated as a polynomial expression consisting of the functions listed in the operator pool. For instance, a formula $TRoot = tanh(feature1) + feature2$ corresponds to the tree shown in Figure 3.4.

3.5.2 Primitive Operators and Terminators

The operator set and terminator set are the mathematical basis of the GP operations. The terminal set consists of variables and a number of constant values used by the program. For the case of feature extraction, each terminator connects

Symbol	No. of Inputs	Description
+, -	2	Addition, Subtraction
*, /	2	Multiplication, Division
square, sqrt	1	Square, Square Root
sin, cos	1	Trigonometric functions
asin, acos	1	Trigonometric functions
tan, tanh	1	Trigonometric functions
reciprocal	1	Reciprocal
log	1	Natural Logarithm
abs, negator	1	Absolute, Change Sign

Table 3.1: The operator sets for GP

to one element of the feature vector. Hence the number of required terminators will be the same as the number of dimensions in the original feature space. The constant values are randomly generated at the construction cycle of new individuals. These numerical values can be either integer or floating point numbers, both ranging from 1 to 100.

The functions stored in the operator pool are mathematical, logical or probabilistic operators that perform operations on one or more inputs. Table 3.1 lists the mathematical functions used as operators in the design. Note that any invalid input to an operator will result in a false flag being assigned to the fitness value in order to filter out individuals who cannot complete the mathematical transforms. This will effectively exclude them from further consideration during the evolution process.

3.5.3 Primitive Operations

Primitive operations are such important functions of GP as they determine how a brand new generation is established based on the survivors from the previous generation. The word "primitive" here refers to the simulation of elementary functions of creating offspring in the natural world. Mathematically, it does not

mean "simple". On the contrary, the underlying mathematical model of the primitive operations are fairly complicated to derive because these operations change the mathematical structure of the operation function by "selecting" and "combining". The resultant chromogen cannot be written as an analytic function of the input chromosomes. However, since the purpose of this research is more focused on the application of GP rather than theoretical analysis, we are not going to investigate further into the mathematical derivations. We are more concerned with applying the methodology with suitable configurations and observing/analysing the performance.

In our design, three "primitive operations" are conducted in the early phase of a generation,

Crossover

In the GP paradigm, one of the most important operations is the crossover operation. In the crossover operation, two solutions are sexually combined to form two new solutions or offspring. The parents are chosen from the population by evaluating the fitness function. Two new individuals are generated by selecting compatible nodes randomly from each parent and swapping them. GP carries out a crossover operation to create new individuals. The crossover operations with one cut point and two cut points are introduced here.

As illustrated in Figure 3.5, a random subtree is selected from each parent, which then exchanges the subtrees at the corresponding node location.

GP can also perform a crossover operation with two cut points. To create new individuals, two compatible cut nodes are randomly selected and are labelled from each parent. The subtrees are swapped based on the label from two, as illustrated in Figure 3.6.

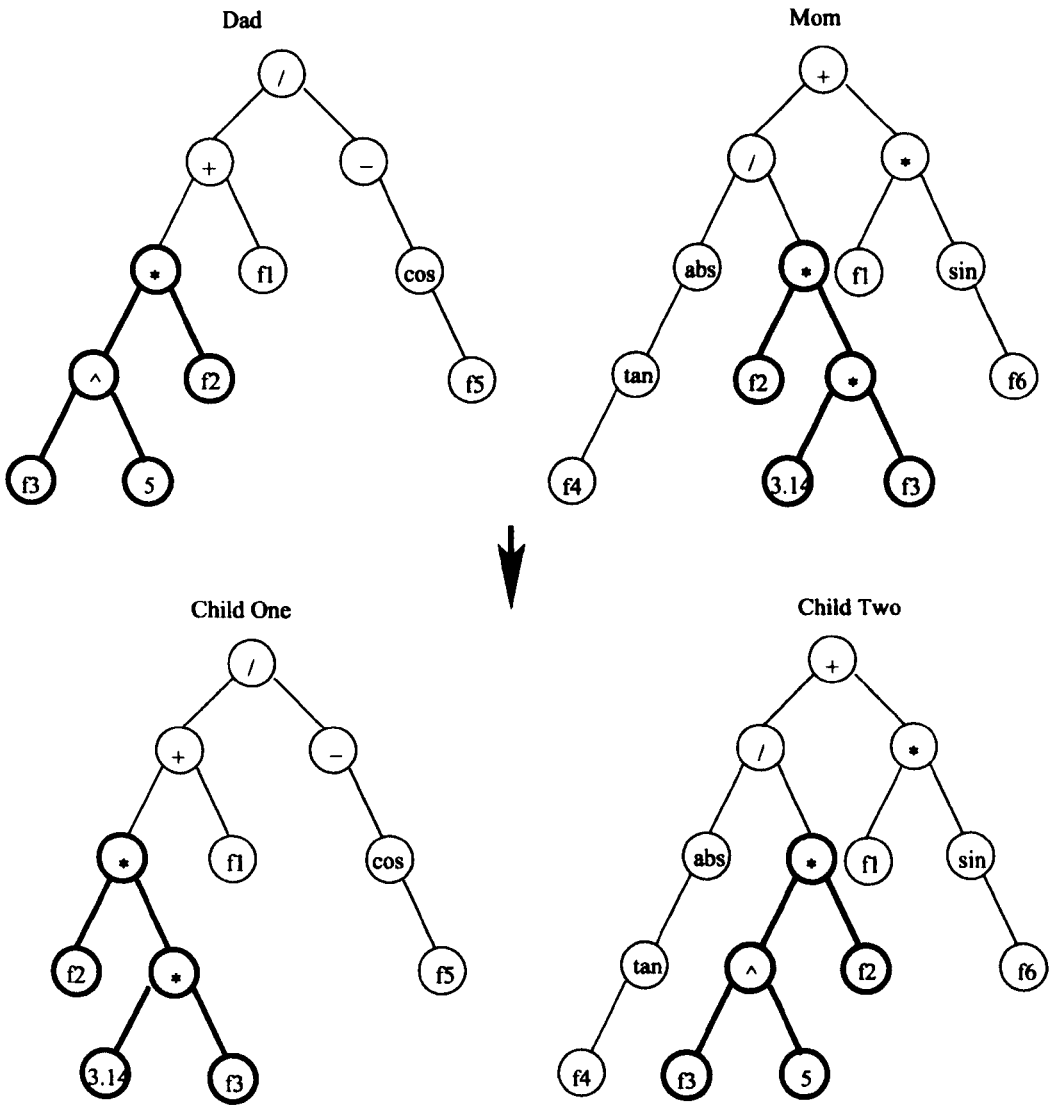


Figure 3.5: An example of crossover operation with one cut point

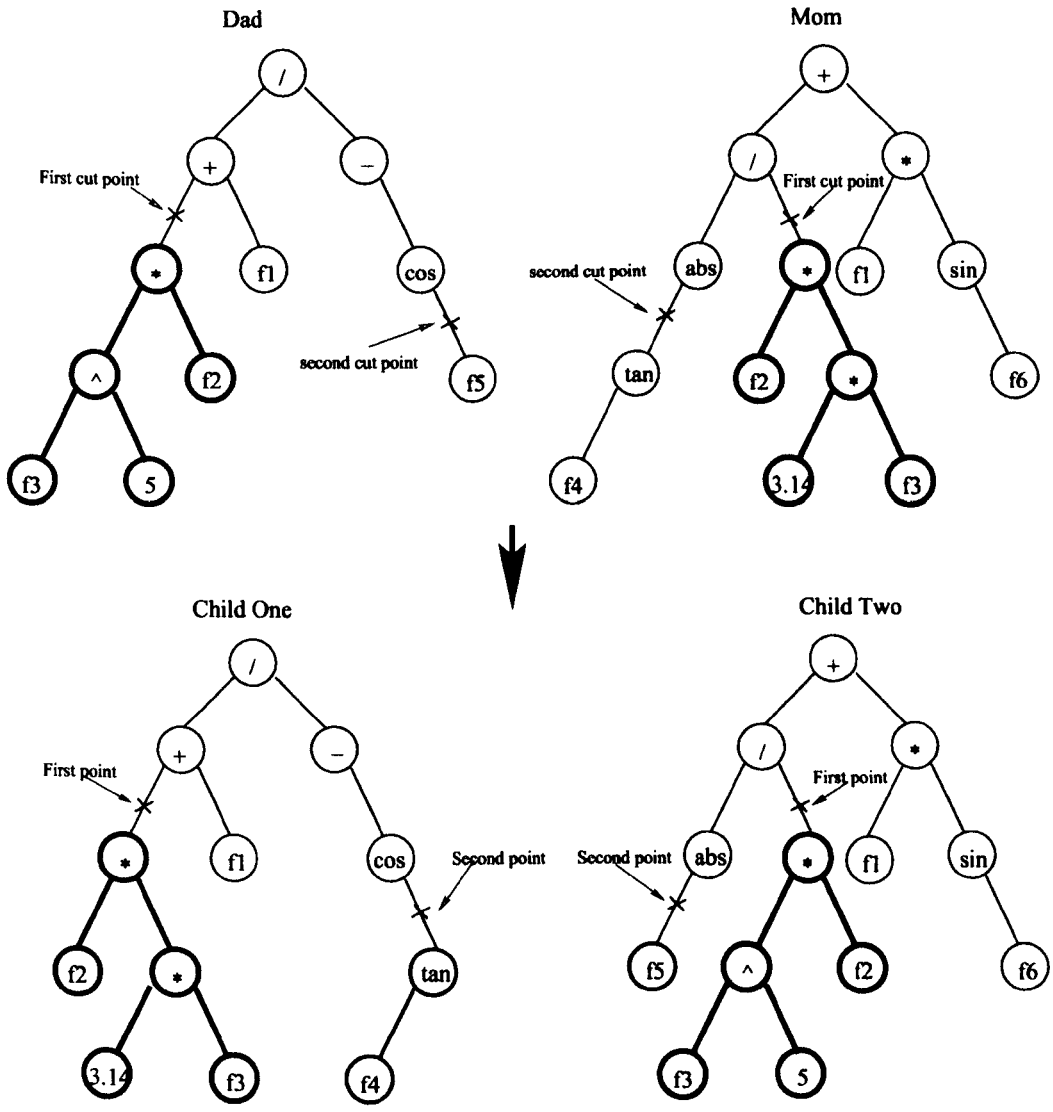


Figure 3.6: An example of crossover operation with two cut points

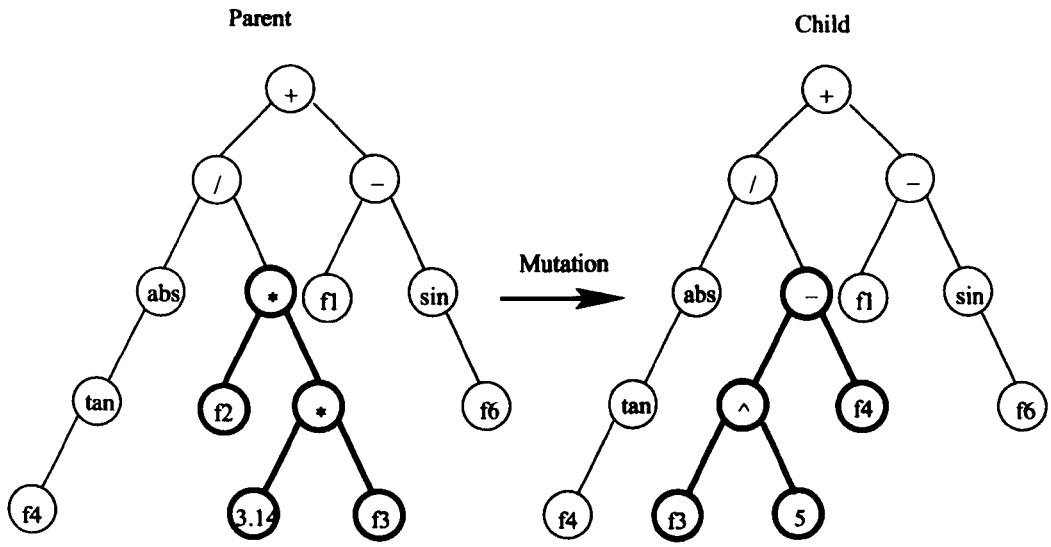


Figure 3.7: An example of mutation operation with one cut point

Mutation

The mutation operation is performed by the creation of a subtree at a randomly selected nodes. For one cut point mutation, there is an index assigned to each node for the identification to a given parent. A random index number is generated to indicate the place where the mutation occurs. Once the node is located, the tree downstream from this node is deleted and a new subtree is generated from the node (see Figure 3.7), exactly in the same way as growing the initial population.

For two cut points mutation operation, two random mutation points are selected among the stored links (mutation point selection), with a uniform probability. The two subtrees below the common mutation points are deleted and new subtrees are created as growing new trees (see Figure 3.8).

Reproduction

The reproduction operation is performed by copying individuals to the next population.

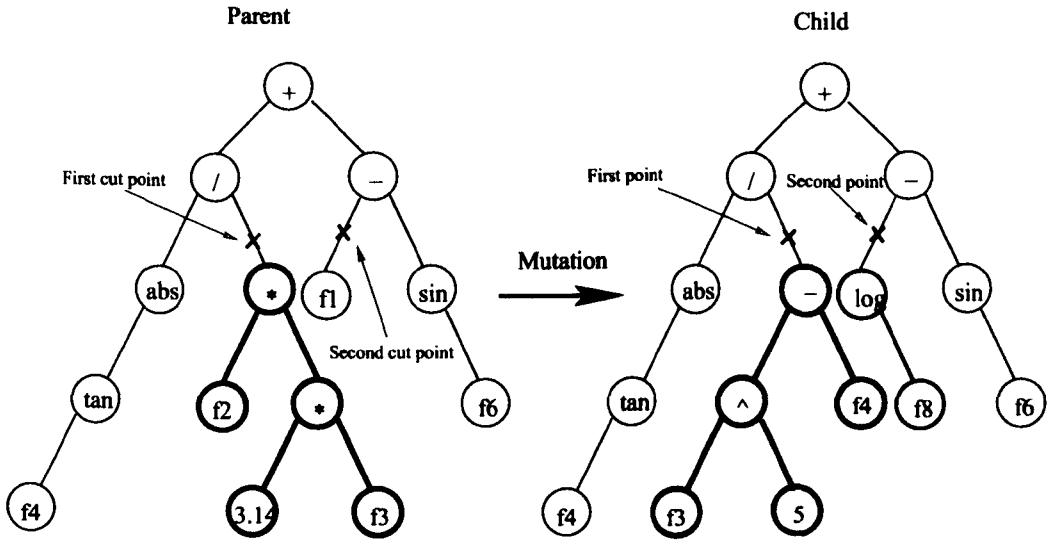


Figure 3.8: An example of mutation operation with two cut point

All three operations take place in the early phase of a generation on a random basis. The probability of occurrence follows,

$$P_{c1} + P_{c2} + P_{m1} + P_{m2} + P_r = 1 \tag{3.1}$$

where P_{c1} is the probability of one-point-cut crossover, P_{c2} is the probability of two-point-cut crossover, P_{m1} is the probability of one-point-cut mutation, P_{m2} the probability of two-point-cut mutation, and P_r is the probability of reproduction.

3.5.4 Fitness Function

Same as the genetic algorithm, the most difficult and the most important concept of genetic programming is the fitness function. The degree it correlates with the common evolution goal determines how successful the solution can be. At the same time, speed is an issue as in evolutionary computation, optimality of the solution also depends on the time consumed in the computation of generations.

Generally speaking, evolutionary-algorithms based methods require relatively expensive computation power and resources. Hence, a not "so-good" fitness function with relative low computation demand may end up better solution than a "perfect" fitness function which demands a lot of computation, because many more generations have been evolved in the former case. In this regard, fitness function can be incorporated with other factors, such as speed of learning, depth of trees.

In this research, GP is proposed as the paradigm for designing feature generation systems for different tasks. The fitness functions are designed based on requirements. In all cases, fitness function consumes most of the computation power in the evolution process. It calculates classification errors, sum squared error or other factors to determine the performance of each individual. However the performance achieved using these measures sometimes do not necessarily match the computation cost. For example, with a less computationally demanding, but effective measure, GP is able to evolve more generations or larger populations and hence explore a larger feature space, within the same period of time. Therefore, from a cost-effective point of view, a good design of fitness measure is a trade-off between the speed and efficiency.

For the problem of pattern classification, intuitive choice of fitness measure will be classification success. There are examples of that in the literature [64,69]. However, classifier is a computationally-demanding method. Training a ANN or SVM to obtain a fitness value of an individual can be very time-consuming, considering that the training, validation and testing of the classifier has to be conducted for each individual in the population and repeated many times through the evolution.

In order to reduce the computation complexity, in our design, the classifier is not used in the network . The score of the fitness function is not the classification accuracy. The details of the fitness measure will be addressed accordingly in each

chapter.

3.6 Implementation & Testing Issues

3.6.1 Code Bloat

Code bloat is defined in GP context as the production of unnecessarily long, slow, or resource-wasting codes during evolution [92, 93]. In particular, it refers to the nodes that have no effect on the individual's output. The unnecessary growth of programs produced by genetic programming is a well documented phenomenon [94, 95]. Quite often, it occurs in fitness-based search techniques which allow variable length solutions.

Apart from the redundancy introduced by code bloat, the harm it might cause is to decrease the likelihood of finding improved solutions. So far no benefit has been identified in code bloat for real-world applications. Hence it is necessary to limit the occurrence of code bloat during evolution. Typical solutions to code bloat in GP include,

1. Setting a fixed limit on the size or depth of the programs. The technique is effective but needs prior domain knowledge to determine a reasonable limit on the depth or size.

2. Dynamic limit on the size or depth of the programs. The evolution starts with an initial limit on the program size. A child may only exceed this limit if its fitness is better than the best fitness obtained so far. In this situation, the child is moved into the next generation, within which the program size limit will be increased to the size of the child.

3. Parsimony pressure. Parsimony pressure is adopted as a penalty to complex

solutions. As a result of that, least complex solutions will be favoured. A basic realisation of parsimony pressure is the complexity factor in the fitness function. In case of tree structure of programs, typically, the complexity factor is calculated by counting the total number of nodes.

In this research, a fixed limit of tree depth is used and empirically proved to be effective in avoiding code bloat for each specific application.

3.6.2 Robustness and Generality

A common generalisation issue of machine learning methods is over-fitting or over-training. For example, a typical problem of generality occurs when we try to use a certain algorithm to train a classifier for the separation of two classes. The algorithm is able to separate the two classes correctly after a considerably long process of training. However, when the test data is in use, the algorithm seems to classify the patterns into wrong groups. This problem often happens in the evolutionary computation. We consider the evolved programs are not robust enough.

The concept of robustness of the evolved programs is the ability to work correctly on general data sets rather than a particular one it was generated on. The desired performance of robustness solution should be obtained in an environment similar to the one within which it was evolved. In conventional learning methods (such as ANNs, SVM), the generalisation problem is well understood. Extensive research have been conducted in this area [96–98]. However, research dedicated to the generalisation issue in GP is limited to a few papers [99–101] in the published literature.

In recent years, the ability of generality and robustness of GP has obtained an increasing amount of attention from researchers. The system with the ability to

generalisation has been recognised in the study of GP. Many researchers report that solution to learning problems may result in non-robust programs [48, 102–105].

In [105], a series of methods used for improving the ability of robustness are discussed. Three major approaches are introduced to deal with non-robust solutions: modifying fitness function, improving the selection of fitness cases, using co-evolution, and two or more populations being evolved at the same time. Hooper and Flann [106] show that the size of programm in a GP search tends to increase the redundancy. The large sub-expressions are replaced by a much simpler, smaller expression which could produce similar results. An expression simplifier is applied to identify and eliminate the redundant expressions and to rewrite the bloated expression into a simpler, but equivalent expression. They further state that when training and testing phases are adapted, improving the fitness function to prefer simpler expressions may help avoid over-fitting and obtain better predictive accuracy during the test phases. In [107], the test set is used to calculate the fitness of individuals. The method of measuring generalisation performance of the system reveals that the measure is actually based on the testing data. At the end of each generation, the performance of individual on the test set is recorded. After each GP run, the individual with the highest recorded performance is chosen as the solution to the problem at hand. However, the test set used in his paper is also used in the learning process. As mentioned by the authors, the testing data is no longer a test set, it has become part of the training data.

In this research, generality is not the major focus as no classification results are used during the evolution process. Instead, statistical tests play a major role in the fitness measure. Hence discussion about generality will be addressed in the final stage of feature extraction process–the validation and testing of relevant

classifiers using GP-evolved features.

3.6.3 Choosing test data sets

In order to test the robustness and generality of the proposed GP methods, two industry problems, one medical problem, one telecommunication problem, and a series of machine learning data sets available from the public domains, are chosen for the testing purposes. Specifically they are,

1. Bearing fault detection (dual class problem)
2. Machine condition monitoring (six class problem)
3. Breast cancer detection (dual class problem)
4. Automatic digital modulation recognition (ten class problem)
5. Type of Iris plant (three class problem)
6. Balance of the scale (three class problem)
7. Contact lenses for different patients (three class problem)
8. Zoo animal classification (seven class problem)
9. Lung cancer (three class problem)

Problems 5, 6, 7, and 8 are relatively simple problems. The idea to test on these date sets is to demonstrate the functionality and the strategy of searching. Problem 1 and problem 2 are from industry applications, where competitive performance is desirable, as well as the implementation cost. Problem 3 and problem 9 are from medical domain, where reliability is an important issue. Problem 4 requires the recognition of different signal patterns from noise-contaminated environment. Hence accuracy is the main focus.

3.7 Summary

This chapter provides an introductory description to the GA/GP paradigm and the implementation issues related to the design of the program for the feature extract tasks. The concept of evolutionary computation has been reviewed, in terms of genetic search algorithms, natural selection schemes, genetic operations, and the functionality of fitness measures. Furthermore, issues related to the implementation and testing of GA/GP have been discussed, including the code bloat issue, generality issue and choice of data sets.

Chapter 4

Multi-Feature Generation for Multi-Class Problems

4.1 Introduction

Multi-class pattern recognition is the study of classification methods for the separation of more than two classes in the class distribution. It has a many of applications, such as hand-written digit recognition, text categorisation, voice recognition [4,5], etc.

However, many powerful classifiers are originally developed for binary class problems, such as SVM and Perceptron. A classification function works in a way that an optimised threshold determines the boundary between the two class distributions. In order for a binary classifier to separate multiple classes, the most popular approach is to decompose the problem into multiple two-class classification problems. The results from all the binary classifier are then combined together to provide an overall estimation of the class assignment.

There are a number of ways to decompose a c-class pattern classification problem into binary-class problems [12,16,19,22]. In terms of decision making,

there could be two approaches,

Decision making by voting This is to apply a voting mechanism, where each classifier votes for (or against) a certain class. The pattern is assigned to the class with the highest number of votes. By doing this, the classifier is required to output a binary answer.

Decision making by winning The other way is to assign the class with the most confident result. Each classifier outputs a confidence value, rather than a binary answer. The highest confidence level indicates the maximal likelihood of the answer. Clearly this requires a classifier to produce a continuous confidence value.

Decision making by error-correction In addition, error-correcting can be introduced to change the class assignment by learning.

In terms of decomposition, there could be two possibilities,

One against all For a $c(c > 2)$ class problem, a total of c classifiers are designed for the separation of each class against all other classes. c number of results will be obtained and the class assignment is based either on "voting" or "winning".

Between every two For a $c(c > 2)$ class problem, one classifier is required to separate between every two classes. Therefore, there will be altogether $\frac{c!}{(c-1)!2}$ classifiers. In this scenario, a voting mechanism can be implemented for the assignment of class labels.

In our design, the idea of the second approach has been adopted, but not using any classifier and not in the classification stage. Rather, in the feature extraction stage, the fitness measure is mainly based on this idea. The detailed design will be addressed in the later section.

For multi-class pattern recognition problems, machine learning techniques have been extensively used for the automatic generation of features. For instance, Raymer *et al.* present an approach to feature selection using a genetic algorithm, which optimises a weighting vector used to scale the individual features [19]. A masking vector is also employed to perform simultaneous selection of a subset of the features. This technique is employed in combination with the k-NN classifier. It is further compared with the results from classical feature selection and extraction techniques. The results show that the combination of GA and k-NN is more effective than most of other approaches in comparison, and require less number of features for the same level of classification success. GP is applied to a dual-class pattern classification problem in [56]. A single expression is evolved in one run of GP. In [108] the feasibility of applying GP to multi-class pattern classification problem is studied. It is the first time that GP is applied to a multi-class problem. A genetic programming classifier expression (GPCE) is evolved as a discriminant function for each class. They also discuss the various issues that arise in the implementation of GP-based classifier, such as the creation of training sets, the role of incremental learning, and the choice of function set in the evolution of GPCEs, as well as the conflict resolution for uniquely assigning a class.

In [67], GP technique is used to develop a decision support system for vehicle dispatching. It is conducted by evolving a population of utility functions that evaluate candidate vehicles for servicing requests. GP is tested in a medical diagnosis problems of six classes [109]. The results are compared with those obtained by neural networks. It is worth noting that, however, in all of the above applications [67, 108, 109], GP is employed solely as a classifier based on manually developed features.

In [64], as a data preprocessor, GP-based feature extraction is conducted for

a medical problem. It performs intelligently by deciding whether to conduct feature extraction or feature selection. Unfortunately, the system is unable to sample adequately the search space for high-dimensional problems. The main disadvantage of the method is the computational complexity. Kotani *et al.* [110] conduct feature extraction using GP with a k-NN classifier on one artificial task and one acoustic diagnosis problem. It concludes that the genetic programming is an effective tool for the feature extraction task.

In this chapter, genetic programming is applied for the purpose of multi-feature generation for multi-class pattern recognition problems. In order to test the performance of proposed GP-based feature extraction system, a data set is carefully chosen for the task. A well-known industry problem is used as a benchmark. Using a novel fast method to evaluate the difference among classes, this method alters the distribution of each class based on the well-known Fisher criterion, instead of using classification results to determine the fitness of a feature, resulting in a more economic and fast solution. Different combinations of features and different configurations of classifiers are tested to fully examine the capability and robustness of the proposed method. The strength and advantages are evaluated against other popular methods.

This chapter is organised as follows: The data preparation using vibration signals for condition monitoring is addressed in Section 4.2. Parameters and developed fitness function for GP Multi-feature generation model are described in Section 4.3. Based on the model, a series of feature-extraction experiments for the classification of vibration signals are conducted in Section 4.4. Section 4.5 provides discussions based on experimental results. Advantages and limitation of the GP-based feature-extraction method are addressed in the conclusion part.

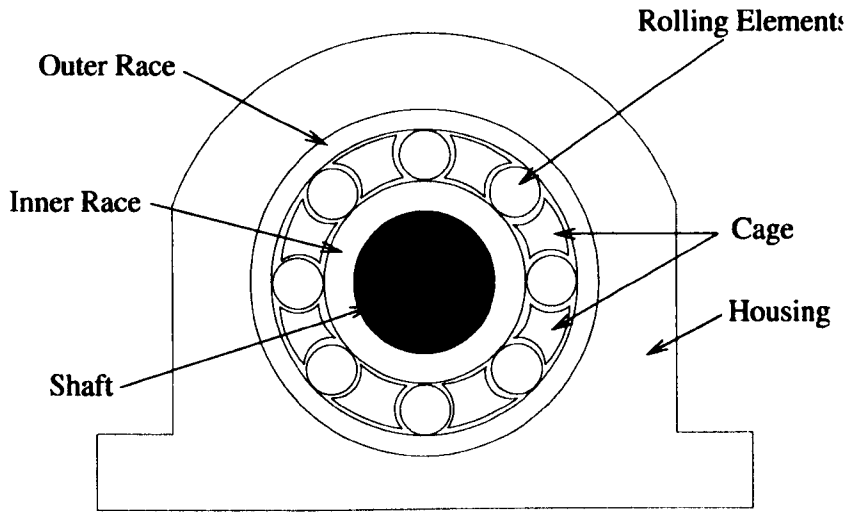


Figure 4.1: A typical roller bearing, showing different component parts [1]

4.2 MCM problem

Machine Condition Monitoring (MCM) is of vital importance in the manufacturing industry. Maintenance costs can be reduced significantly by monitoring health of machinery. Potentially disastrous faults can be detected in early stages, whilst enabling the implementation of condition based maintenance rather than periodic or responsive maintenance.

Rolling element bearings (see Figure 4.1) are probably among the most widely used rotating machine components. It is of vital importance to be able to detect accurately the existence and severity of faults in machinery in certain areas of industry, as in many cases the machine may be safety or emergency related.

In literature, a number of conventional methods are used to analyse the bearing vibration signals in order to extract effective features for bearing fault detection. These include probabilistic analysis [111, 112], frequency domain analysis [113], time-domain [114] and finite-element analysis [111, 112]. In recent

years, evolutionary learning algorithms for machine condition monitoring applications are studied. Genetic algorithm based feature selection is carried out in [1] for the classification of bearing faults using vibration signals. In [115], Chen *et al.* presents a GA-based method to automatically generate symptom parameter functions from rolling bearing data for the diagnosis of machinery operating conditions. Zhang *et al.* [68] applies GP for fault detection in the field of MCM.

The six bearing conditions are,

1. Normal Bearing (NO)
2. Worn Normal Bearing (NW)
3. Inner Race Fault (IR)
4. Outer Race Fault (OR)
5. Rolling Element Fault (RE)
6. Cage Fault (CA)

Each of them has own characteristics in terms of mechanical status. The normal bearing (NO) condition is to indicate a brand new bearing, which has been run in, but is in otherwise perfect condition. The worn normal bearing (NW) is in good condition, however has been running for some period of time and serves as an example of a bearing that has seen some usage. The inner race (IR) fault was created by removing first the cage, moving the elements to one side of the bearing, and removing the inner race. A groove was then cut in the raceway of the inner race, using a small grinding stone, and the bearing was reassembled. The outer race (OR) fault was created by removing the cage, pushing all the balls to one side, and then inserting a small grinding stone. The rolling element (RE) fault was induced by using an electrical etcher to mark the surface of one of the

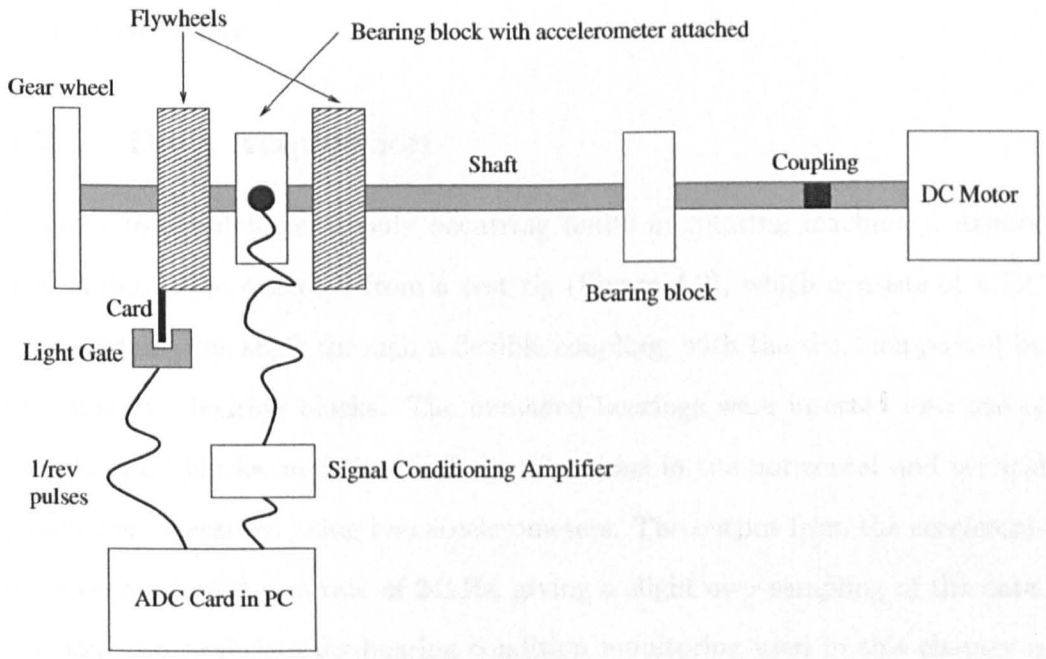


Figure 4.2: Machine test rig used in experiments

balls, simulating corrosion. The cage (CA) fault was simply created by removing the plastic cage from one of the bearings, and cutting away a section of the cage, so that two of the balls were free to move, and not held at a regular spacing, as would normally be the case.

The normal conditions are easily sampled as data can be acquired from working machinery with known bearing conditions. However, the faulty conditions are less easy to resemble, as in the real situation, the machine structure differs and the design of bearing differs. Hence the source of faults could be many. Here, attempt is made to divide the faults into four groups. This is done from an mechanical engineer's point of view. The major sources of bearing faults have been covered. It is a realistic categorisation of roller bearing conditions. The major goal here is not to conduct a full analysis of machine vibration characteristics, but to test the proposed method for feature extraction and classification. Interested readers are referred to [111, 112] for further detailed discussion on roller bearing

machine conditions.

4.2.1 Data Acquisition

In order to simulate commonly occurring faults in rotating machinery, experimental data were collected from a test rig (Figure 4.2), which consists of a DC motor driving the shaft through a flexible coupling, with the shaft supported by two plummer bearing blocks. The damaged bearings were inserted into one of the plummer blocks, and the resultant vibrations in the horizontal and vertical planes were measured using two accelerometers. The output from the accelerometers were sampled at a rate of 24kHz, giving a slight over-sampling of the data. The experimental data for bearing condition monitoring used in this chapter is collected by the machine set, which is a small vibration test rig loaned to the University of Strathclyde by Weir Pumps Ltd. of Cathcart, Glasgow.

4.2.2 Time Domain Characteristics

By examining the actual vibration plots on a time series basis (Figure 4.3), some characteristics can be found. First, signals from four conditions, including NO, NW, CA and OR, look similar with amplitude not exceeding ± 100, while the other two conditions have periodic strong pulsations. Apparently, these two groups can be differentiated easily by examining the amplitude. The two normal conditions look similar, though the signal from worn condition is a bit noisier than that from a brand new bearing. The outer race fault, and cage fault display little difference to the normal condition in terms of magnitude and noise level. Therefore, it will be difficult to identify these conditions solely by time series inspections.

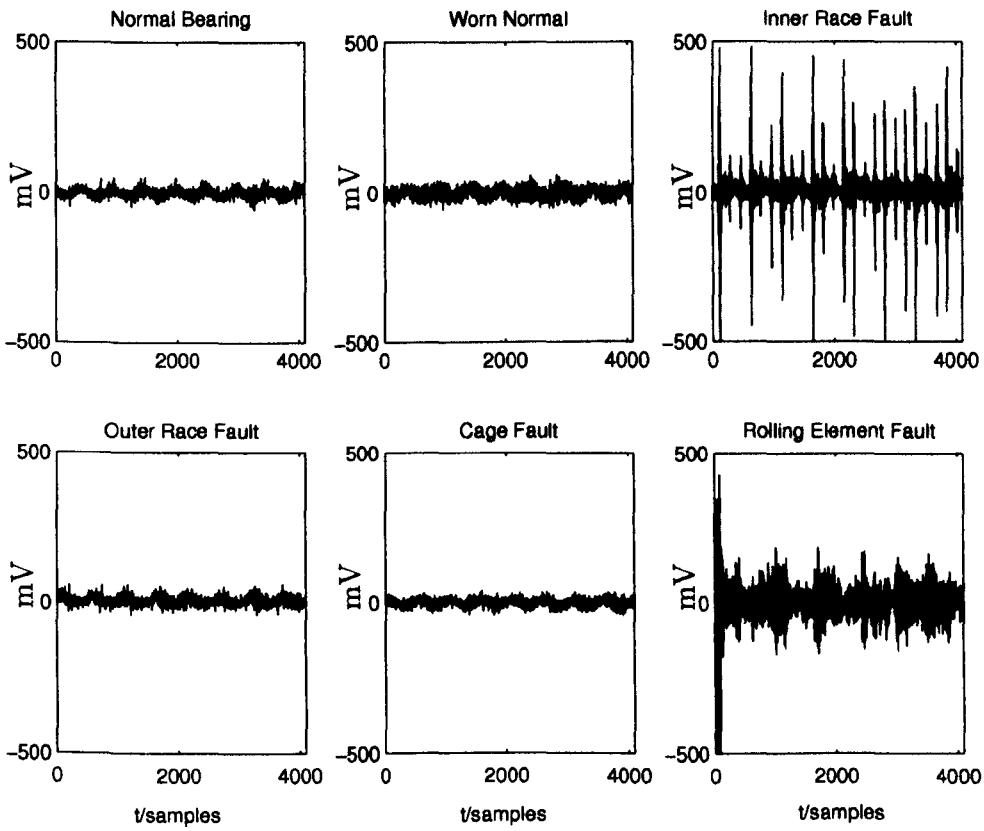


Figure 4.3: Typical vibration signals for six bearing conditions [1]

4.2.3 The Raw Data Set

Experimental data sets were formed by running the machine (see Figure 4.2) over a series of sixteen different speeds and taking ten examples of data at each speed. Each example consists of 2000 data samples. This gives a total of 160 examples of each condition, and a total of 960 raw data examples over six conditions to work with. The full input data set creates a 960×2000 matrix as the training data set. The other two 960×2000 matrices are the validation data set and the test data set respectively. For each given vector in the input raw vibration data sets, a corresponding vector was created in a matrix containing the target information used during the experiment.

4.3 GP-based feature extractor

A GP-based system has been designed and implemented for the task of feature extraction from the above-mentioned roller bearing vibration data. The extracted features are furthered used in various classifiers to evaluate the quality in terms of discrimination information.

The benefit of doing this is twofold. From a researcher's point of view, the general feature extraction capability of GP can be examined by designing such system and conducting relevant experiments. It will be interesting to evaluate the performance against other conventional methods. From an engineer's point of view, manually developing features is a such a tedious and time-consuming work. In many industry problems, generated features should have the ability to identify subtle or complex relationships within large data sets where the mapping from data to class labels is often obscure or difficult for human to identify. Most of time, the result quality heavily relies on expert knowledge and experience.

The overall structure of GP-based system has been established in Chapter

3. The following section will address the design variations related to terminator pool, operator pool, and the fitness function.

4.3.1 Preparation of Terminator Set

The design of the terminator pool is the most application-specific job. It is closely related to the characteristic of the problem. The terminator pool contains all the elementary processing functions on the raw data. The purpose of doing this is to provide a large collection of possibly useful features in the pool, by which GP is able to select, combine and produce features. The terminator can be directed connected to the raw data to avoid such preparation phase. However, it is found that it is much less efficient than simply using the features that already proved to be relevant and useful in the field. This step does not require expert knowledge as there is no filtering or ranking on the choosing of features. As long as it is relevant, it can be thrown into the pool. The job of selection and extraction is left to GP to accomplish.

Nevertheless, knowledge about the problem in hand is still mandatory as basic features are required to be listed and prepared to construct the terminator pool. Specific to our chosen application, MCM, machine vibration signals contains vital information about the condition of the machine. Traditionally they are recorded by one or more than one sensors. By analysing the vibration signal, it is found that the signal energy, or the magnitude of a particular frequency due to a fault are indication of the condition changes. In the following discussion, conventional features useful for MCM are listed. All of them will be packaged and put into the terminator pool.

Conventional Measures

There exist a number of statistically-based performance indicators, which provide single figure assessments of the condition of rolling element bearings. These give an indication of whether a bearing is in a state of distress, or within normal operating parameters, and show the degree of distress that a bearing is under. Conventionally, the three most common measurements used are shock pulse, crest factor, and kurtosis [1].

Shock Pulse (SP) method is a signal processing technique used to measure impact and noise caused by metal to metal contact in the bearing. It is much more refined than other high frequency measurements. Shock pulse analysis relies on a specialised transducer, which has a resonant frequency of $32 - 36$KHz. The amplitude of the shock pulse is relative to the velocity of the impact. For bearing conditions, Carpet Value and Max Value are two readings of the shock pulse, thus can be used as two components of four conventional features.

Carpet Value Metal impact on metal always occurs in rolling element bearings, even a brand new bearing under normal operating conditions. When there is no damage to the bearing, the metal-to-metal contact creates a background noise of shock pulses. This is referred to as the Carpet Value [116]. Carpet Value decreases in the state of high lubrication of the bearing. When damage occurs on the bearing there will be more metal to metal contact, which is reflected by an increasing Carpet Value. By examining the Carpet Values in the signal, information can be gleaned about the likelihood of the existence of a defect.

Max Value Max Value is another conventional parameter used to identify the damage in the rolling element bearings. However, it is not capable of distinguishing different fault conditions of bearings. When a fault occurs within a

bearing element, it is periodically hit by the rolling elements in the bearing. These create a high amplitude burst of shock pulse. The Max Value increases as the bearing damage develops further. This peak from the carpet of background shock signal can be used to detect the damage in applications of bearing condition monitoring.

Crest Factor The Crest Factor (CF) is a commonly used measure for the detection of bearing faults. CF is equal to the peak amplitude of the waveform divided by the root mean square (RMS) value. The analysis of the CF can give an idea of how much impact occurs in time domain. The impacting is associated with the rolling bearings. The CF is relatively high due to the amount of the impact occurring within the bearing and it works well while the fault develops. However, as the degree of damage goes up, the high frequency component of a vibration signal increases. Hence the RMS value increases with the result that the CF value decreases.

Kurtosis

Kurtosis method is one of the vibration signal analysis methods in the category of time-domain analysis techniques. It is a commonly used measure of damage. The definition of kurtosis is given by:

$$kurt = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{x_j - \overline{x}}{\sigma}\right)^4 \tag{4.1}$$

where x_j represents a vibration sample, N is the number of samples, $\overline{x}$ is the average of N vibration samples and σ is the standard deviation. The kurtosis value emphasises the length of the "ails" of a distribution. Signals show a lot of sharp impacts when the rolling elements of a bearing strike a defect. Consequently the value of kurtosis will be high. The kurtosis value will be low while signals

have little or no spike content.

Plain Statistics

It is well known that vibration signals mainly depend on the resonant frequencies of different parts of the machine. If the machine condition varies due to wear or damage, the resonant frequencies, and hence the vibrations, will change. In addition, it is generally not possible to classify the condition based upon an individual sample of the vibration, thus some transformation of the recorded vibration time-series is required to extract time-invariant features. These are statistical moments and cumulants. For example, as the machine's condition deteriorates, the energy (mean square value) in the vibration signal is expected to increase. A number of different statistical features were generated using moments and cumulants of the vibration data. The nth-order moment is denoted by a pair of straight brackets in the subscript as in Equation 4.2. The four plain statistical features used here are the four (first to fourth order) moments. These are stored in a matrix of size 4×960.

$$m_x^{[n]} = \frac{1}{N} \sum_{i=1}^{N} x_i^n \tag{4.2}$$

Signal Difference and Sums

Differences highlight the high-frequency components in the signal, and the sums of the signal emphasise the low-frequency portions. The numerical derivative of each vibration signal was calculated by Equation (4.3). The four plain statistical features were calculated from the derivatives. The results were saved in another 4×960 matrix. The numerical integral of vibration signal were given by Equation 4.4. Using the same process, this creates another 4×960 matrix.

$$d(n) = x(n) - x(n-1) \tag{4.3}$$

$$i(n) = \{x(n) - m_x^{[1]}\} + i(n-1) \tag{4.4}$$

High- and Low- Pass Filtering

The four plain statistical features were calculated on data filtered using an eighth-order Butterworth IIR high pass filter with a cut-off frequency at 129 Hz; this gave another 4×960 matrix. A low-pass filter with the same cut-off frequency were used on the same data sets, and gave a 4×960 matrix.

Normalisation

The importance of normalisation to both the efficiency and accuracy has been demonstrated [20]. The normalisation in experiments is based on Equation 4.5,

$$\mathbf{f}_i' = \frac{\mathbf{f}_i - \mathbf{m}_f}{\boldsymbol{\sigma}_f} \tag{4.5}$$

where $\mathbf{m}_f$ is the vector mean and $boldsymbol\sigma_f$ is the vector standard deviation.

4.3.2 Operator Pool

The operator pool is the same as designed in Chapter 3.

4.3.3 Fitness Function

As mentioned in the previous chapter, the classification results are traditionally used as the fitness value for multi-category classification problem. However, the computational demands are relatively high in training and validating a classifier for each individual. To avoid such complexity, Fisher criterion is adopted for the solution of the feature extraction problem based on the advantage of the maximisation of inter-class scatter over the intra-class scatter. GP as an evolutionary

method is proposed to maximise the degree of difference between two classes, analogous to the Fisher criterion, but in an iterative process. The following expression is obtained based on the Fisher criterion as a class separation measure between class p and class q,

$$\rho^{(p,q)} = \frac{\|m^{(p)} - m^{(q)}\|}{\sqrt{[\sigma^{(p)}]^2 + [\sigma^{(q)}]^2}} \tag{4.6}$$

where m and σ are the class mean and standard deviation respectively. They are given by,

$$m = \frac{1}{N}\sum_{i=1}^{m} f_i$$

and

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(f_i - m)}$$

where f represents the feature output of each individual (note that the output of the tree-structured chromosome is always a scalar).

Based on this, the c-class ($c > 2$) Fisher criterion can be decomposed into k ($k = \frac{c!}{(c-1)!2}$) two-class Fisher criteria. In order to solve the worst case in k two-class problems, the fitness value is defined to be determined predominantly by the minimum one of k two-class Fisher criteria, each of which measures the distribution of inter-class scatter over the intra-class scatter for any two classes. Considering the overall distribution of c classes, the weighted mean of k two-class Fisher criteria will contribute to the fitness function by selecting the best feature for c classes. The fitness measure can defined as,

$$F = min(\rho_i) + \lambda \cdot \frac{1}{k}\sum_{i=1}^{k}(\rho_i) \tag{4.7}$$

where parameter λ is an empirical factor that accepts the contribution from mean

value and at the same time diminishes the effect of too large mean values. The purpose of taking the average is to take into account the distribution of conditions rather than the worst-separated two classes. Consequently, the feature with a few large criteria values but small minimum value cannot compete with the one with average criteria values but relatively large minimum value. On the other hand, if the minimum values for two features are similar, the one with the largest average of values will survive. Specifically, λ is chosen equal to 0.001. Overall, the individual having high fitness value means that difference between any two conditions, even the closest classes, is large.

In this design, only the elitist will survive the natural selection. This mechanism allows the feature to evolve in a direction towards the best classification performance, thus achieving the automatic generation of features.

4.4 Results

4.4.1 Feature Extraction Results

For illustration purposes, two examples of GP-extracted features with different stopping criteria are described in this section.

GP-Generated Feature One

Feature 1 was extracted by GP after evolving 1000 generations using the raw vibration data as the input. The maximum tree depth was chosen as five. In the evolution process, there is always a possibility that two chromosomes in one population are identical and the probability generally increases with the size of the population. Consequently the population size does not need to be large for only four terminators and in this experiment is chosen as 10 to avoid unnecessary

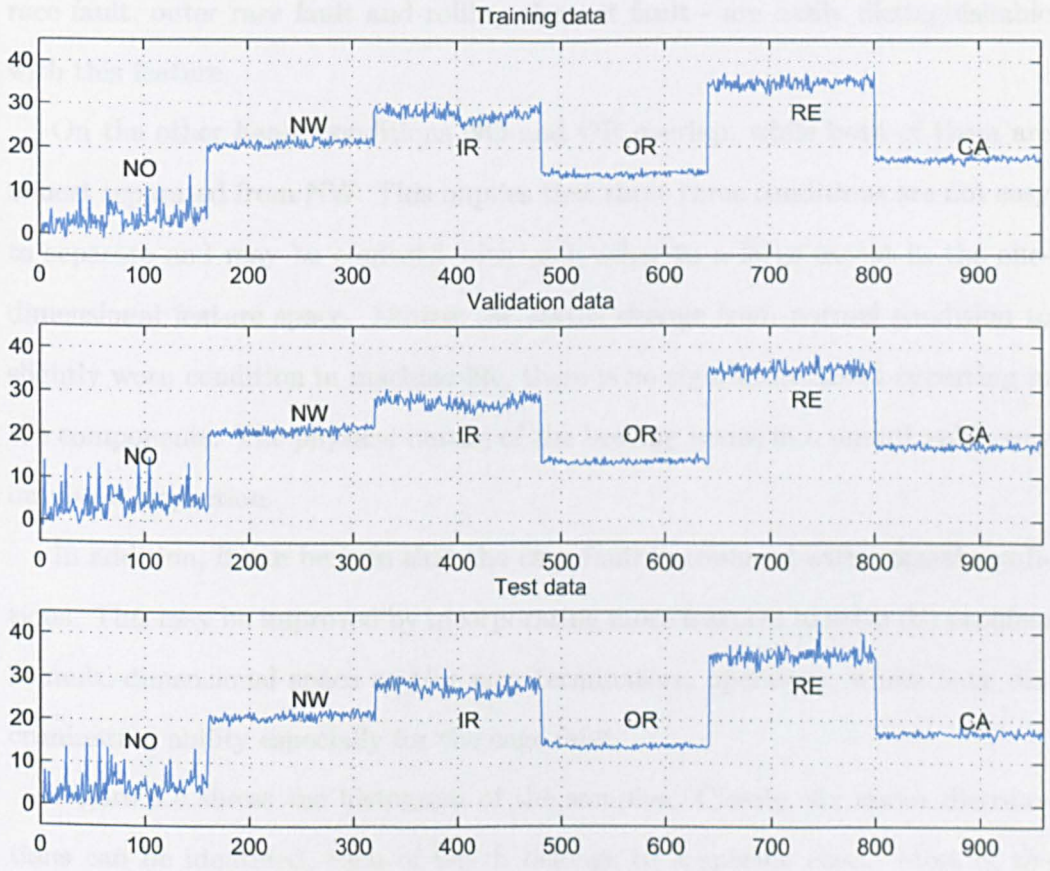


Figure 4.4: The output of first feature generated by GP for six bearing condition monitoring problem (for simplicity, the value of the feature output is unit-less as the actual unit will vary for each feature due to the tree structure.)

computations. The total time for computation is about five minutes for running 1000 generations. The extracted feature is formulated as,

$$f1 = log[(m_x^{[2]})^4] - [log(m_x^{[4]}) - m_x^{[1]} + \frac{m_x^{[1]}}{m_x^{[4]} - m_x^{[3]}}] \tag{4.8}$$

Figure 4.4 shows the feature-processed data for six different conditions. There are altogether 960 examples from the six conditions, with 160 examples for each class. Evidently, class IR, OR and RE are well separated from each other, and from classes NO, NW and CA as well, meaning that three faulty conditions - inner

race fault, outer race fault and rolling element fault - are easily distinguishable with this feature.

On the other hand, conditions NO and OR overlap, while both of them are almost separated from NW. This implies that these three conditions are not easy to separate and may be confused with each other to a large extent in the one-dimensional feature space. During the status change from normal condition to slightly worn condition in machine life, there is no significant defect occurring in the components. The physical nature of the bearing varies in a unnoticeable way on visual inspection.

In addition, it can be seen that the cage fault is confused with normal conditions. This may be improved by incorporating more features to solve the problem in multi-dimensional space and/or new terminators, operators, which have discriminating ability especially for the cage fault.

Figure 4.5 shows the histogram of the samples. Clearly, six major distributions can be identified, each of which belongs to a specific class. Most of the samples from different classes are well separated from each other. Most of the misclassification happens between condition NO and OR, which is also evident in Figure 4.4.

GP-Generated Feature Two

Feature 2 is generated by GP after 10,000 generations with the population size of 14 and the maximum depth of 10. It took about one hour for computation. The formula of the feature is given by:

$$f2 = \cos(log(\frac{m_x^{[2]}}{m_x^{[4]}}) - \tan|(m_x^{[3]})|) - \tanh(m_x^{[3]})$$

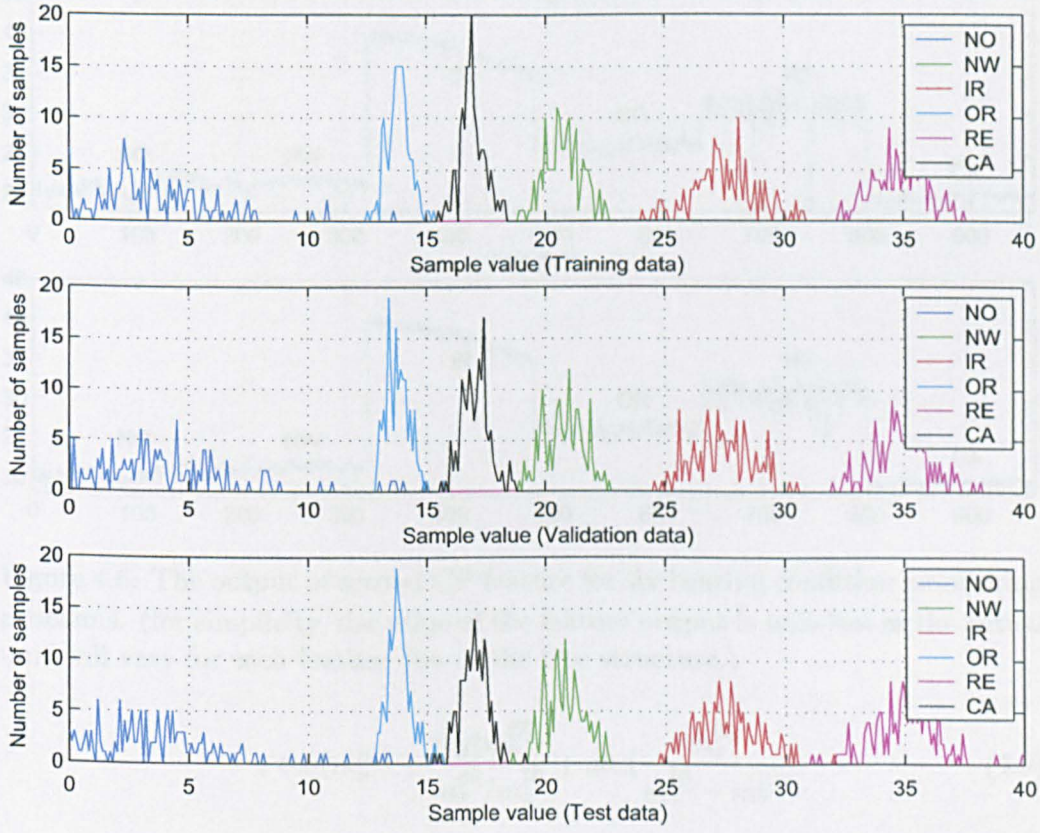


Figure 4.5: The histogram of first feature generated by GP for six bearing condition monitoring problem

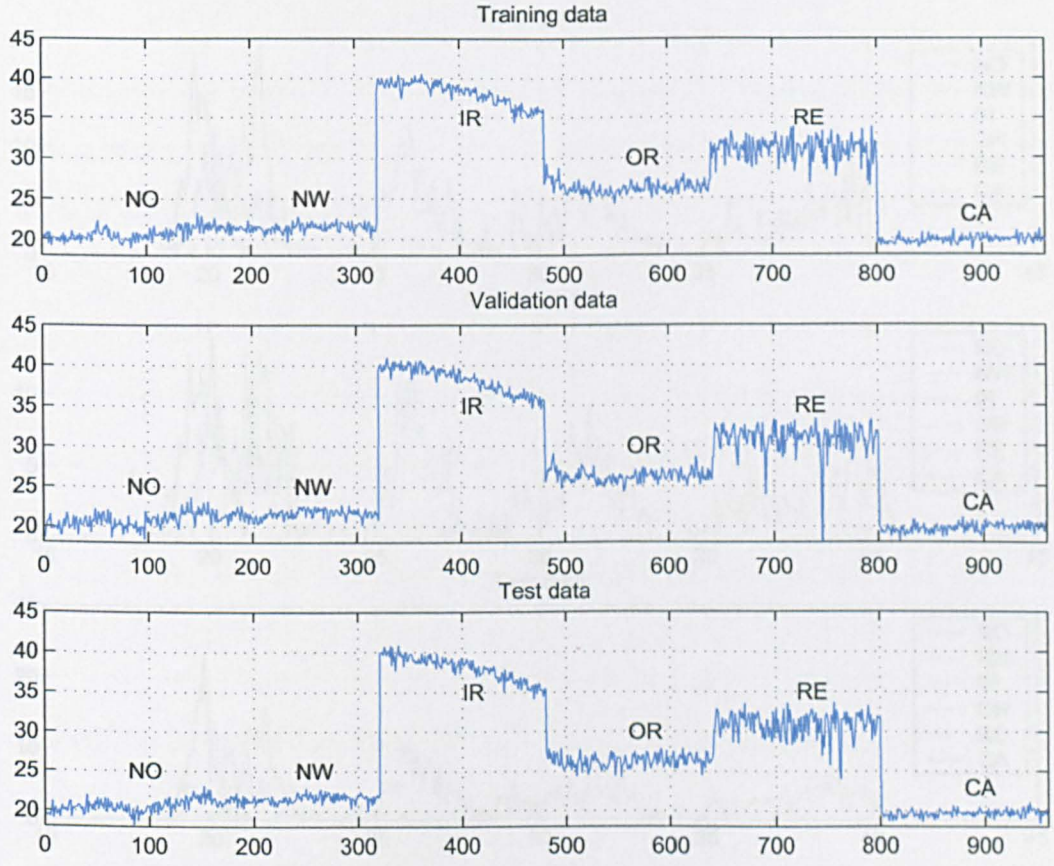


Figure 4.6: The output of second GP feature for six bearing condition monitoring problems. (for simplicity, the value of the feature output is unit-less as the actual unit will vary for each feature due to the tree structure.)

$$+\cos(m_x^{[1]}) + \frac{log(m_x^{[2]})}{m_x^{[2]}/m_x^{[4]}} + abs(\frac{m_x^{[1]}}{m_x^{[4]} - m_x^{[1]}}) \tag{4.9}$$

Figure 4.6 demonstrates that, condition IR, OR and RE are well separated, although condition RE does not have as good separation as that by feature 1. This is mainly due to the fitness algorithm, which uses the smallest Fisher criterion value as the fitness, thus gives the feature with better discriminating ability for closest classes, specifically condition NO, NW and CA, more chance to survive. This can also be seen as a compensation among all classes in order to give the best

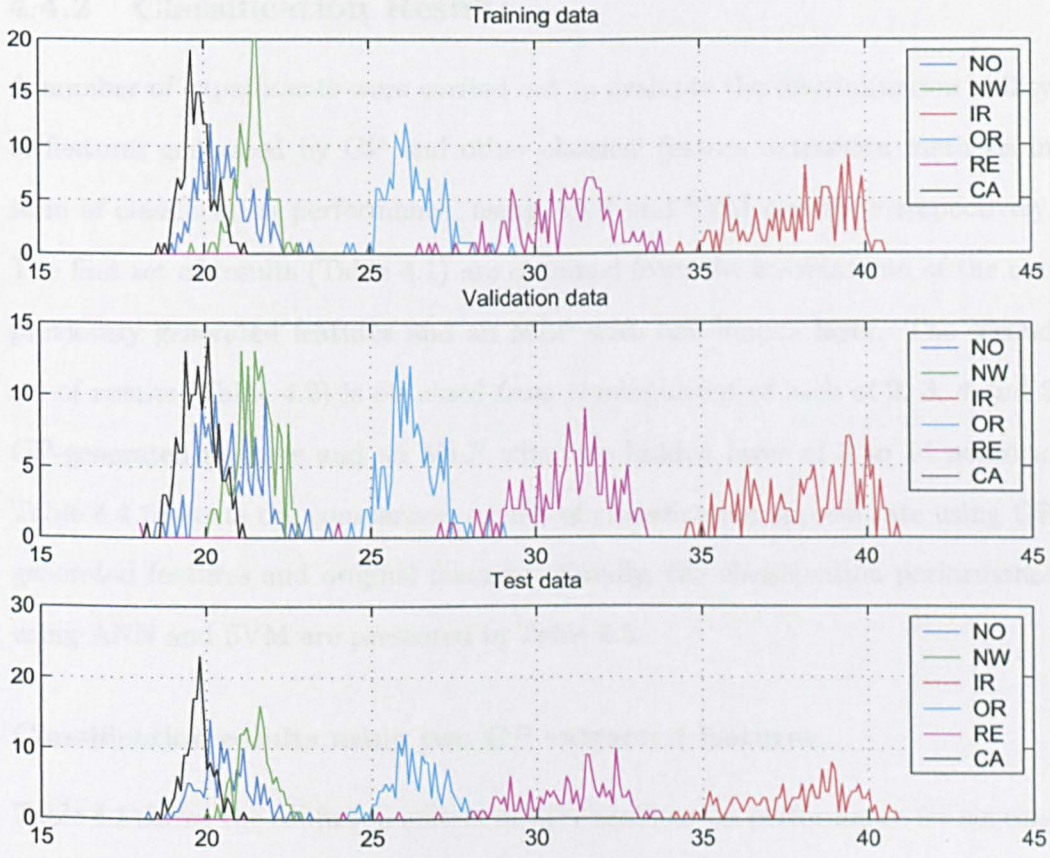


Figure 4.7: The histogram of second feature generated by GP for six bearing condition monitoring problem

overall performance. Clearly, classes NO (example $1-160$) and NW (example $161-320$) are better separated than feature 1.

Figure 4.7 illustrates the histogram of the samples from plotting the feature 2 in the one dimensional space. It can be seen that this feature is not as good as feature one in terms of separation of classes. Condition NO, NW and CA have considerable overlaps in the feature space, which makes it difficult to classify between the three. Moreover, class RE and OR have overlapping sample values.

Overall, the GP-based feature extractor performs well by separating different classes without explicit knowledge of the statistical distribution of the data.

4.4.2 Classification Results

A number of experiments were carried out to evaluate the discrimination ability of features generated by GP and other classical feature extraction methods in term of classification performance, using ANN and SVM classifiers respectively. The first set of results (Table 4.1) are obtained from the combination of the two previously generated features and an MLP with one hidden layer. The second set of results (Table 4.2) is obtained from combinations of each of 2, 3, 4 and 5 GP-generated features and an MLP with one hidden layer of 3 to 14 neurons. Table 4.4 presents the comparison results of classification success rate using GP generated features and original features. Finally, the classification performance using ANN and SVM are presented in Table 4.5.

Classification results using two GP extracted features

Table 4.1 shows the confusion matrix of the classification performance for six conditions, using only two features extracted by GP. Each row of this table shows the associated classification results made by the MLP for a given condition. Each entry in the row shows what the perceived classification is expressed as a percentage of the total number of cases for the condition. From Table 4.1, it can be observed that conditions IR, OR and RE manage to achieve 100% accuracy, conditions NW and CA achieve 95.6% success. 10% of samples from Normal condition are misclassified with condition CA. It is also clear from the Figures 4.4 and 4.6 that most of the misclassifications occur between classes NO and CA.

	NO	NW	IR	OR	RE	CA
NO (160 in total)	**85**	5	0	0	0	10
NW (160 in total)	0.6	**95.6**	0	0	0	3.8
IR (160 in total)	0	0	**100**	0	0	0
OR (160 in total)	0	0	0	**100**	0	0
RE (160 in total)	0	0	0	0	**100**	0
CA (160 in total)	3.1	1.3	0	0	0	**95.6**

Table 4.1: Classification performance (%) for GP/ANN, using two features extracted by GP.

Classification result using different number of GP extracted features and neurons

Table 4.2 shows the percentage of classification success for six bearing conditions using 2 to 5 GP extracted features and MLP, which consists of one hidden layer with 3 to 14 neurons. It is clear that the GP/ANN classification success rate is always more than 95%, with the lowest being 95% (for 8 neurons with 2 features or 8 neurons with 3 features) and the highest being 96.7% (for 14 neurons and 5 features). It can be seen that using the GP generated features the classification results are fairly independent of the number of neurons. The improvement of classification success by increasing the number of neurons is fairly small, ranging from 0.9% to 1.3% for each feature set.

The correlation coefficients of five GP extracted features are given in Table 4.3. It can be seen that all absolute values of coefficients by feature 1 are 0.4 or less. Feature 2 to 5 are fairly correlated, sometimes positively and sometimes negatively. Two high negative correlations occur between feature 2 and 4, feature 3 and 5. These features are highly correlated. In addition, the lowest coefficient among feature 2 to 5 is 0.7729, which demonstrates the similarity between these features. It can be seen that the information content within feature 1 is independent from the other four features, which are positively correlated to varying

	2 features test perf.(%)	3 features test perf.(%)	4 features test perf.(%)	5 features test perf.(%)
3 neurons	95.3	95.9	96.0	95.9
4 neurons	95.2	95.7	95.8	96.1
5 neurons	95.5	95.6	95.9	96.3
6 neurons	95.3	95.9	96.1	96.3
7 neurons	95.1	95.9	96.1	96.1
8 neurons	95.0	95.0	95.2	95.7
9 neurons	95.5	96.1	96.3	96.4
10 neurons	95.8	95.9	96.3	96.4
11 neurons	95.4	96.0	96.3	96.3
12 neurons	96.0	96.0	96.3	96.1
13 neurons	95.9	96.1	96.5	96.3
14 neurons	95.8	96.1	96.4	**96.7**

Table 4.2: classification success (%) with 3 to 14 neurons in one hidden layer of ANN, using 2 to 5 different features extracted by GP.

	feature1	feature2	feature3	feature4	feature5
feature1	1.0000	-0.4084	0.4018	0.3576	-0.2709
feature2	-0.4084	1.0000	-0.8754	-0.9388	0.8182
feature3	0.4018	-0.8754	1.0000	0.8099	-0.9206
feature4	0.3576	-0.9388	0.8099	1.0000	-0.7729
feature5	-0.2709	0.8182	-0.9206	-0.7729	1.0000

Table 4.3: correlation coefficients of five features extracted by GP

degrees. The classification results obtained using two to five GP extracted features are stable and independent of the number of neurons in the MLP (see Table 4.2).

Classification results using ANN with the GP generated features and four plain statistical features

The percentages of correct classification are listed in Table 4.4 for the comparison of performance between four normalised plain statistical features and GP-extracted features with the variation of number of neurons used in the MLP hidden layer. Each classification success rate (%) of GP/ANN is obtained by

averaging over experiments using 2, 3, 4 and 5 GP-generated features. The classification success rate using GP-extracted features are higher than those using four plain statistical features, with improvements ranging from 1.9% in the case using 13 neurons to 17.4% in the case using 3 neurons. It is interesting to see that the increase in the number of neurons does not seem to help the classification much in cases using GP-extracted features while the original features require more neurons to achieve better performance, with only a 78.4% success for 1 neurons and a maximum 94.3% success when using 13 neurons. However, only 79.4% classification success is obtained when 14 neurons are used. This can be explained by the fact that the GP-based evolutionary process has extracted the most useful information from the data and has expressed these in a suitable way for classification, while the four plain statistical features have to rely on the classifier to reduce the effect of unwanted interference. In addition, the range of variance of test success obtained by using GP generated features is from 0.1% to 0.5%.

Among the total 960 number of test data values, the misclassification varies from 1 to 5 samples on an average basis. It can be concluded that, in terms of classification success, the GP-evolved features are more robust than the plain statistical features.

4.4.3 Comparison to Classical Methods

The classification performance results displayed in Table 4.5 were obtained using features generated by GP and classical methods. Both ANN and SVM classifiers are utilised in order to examine the capability of different feature sets over different classification algorithms. In this experiment, features are directly used as the

	GP-Generated Features/ANN		Four Plain Stat. Features/ANN
No. of Neurons	Test success(%)		Test success(%)
	average	stdev	best
3 neurons	95.8	0.3	78.4
4 neurons	95.7	0.4	81.1
5 neurons	95.8	0.4	81.9
6 neurons	95.9	0.4	89.5
7 neurons	95.8	0.5	92.3
8 neurons	95.2	0.3	94.2
9 neurons	96.1	0.4	94.1
10 neurons	96.1	0.3	88.5
11 neurons	96.0	0.4	93.4
12 neurons	96.1	0.1	92.2
13 neurons	96.2	0.3	94.3
14 neurons	96.3	0.4	79.4

Table 4.4: classification success (%) with 3 to 14 neurons in one hidden layer of ANN, using 2 to 5 different features extracted by GP.

inputs to classifiers without normalisation. As listed in Table 4.5, among classical methods, conventional features achieve the best classification performance for both ANN classifier with success rate at 91.5% and SVM classifier with success rate at 92%. The four low-pass-filter features perform the worst with around 16.7% success in SVM classifier. The failure is mainly due to large variation in values in the un-normalised data. It is interesting to see that the difference of performance between ANN and SVM classifiers is significant when using filter features and difference features. The performance disparity is fairly small in other features, because the SVM classifier is sensitive to large values within the input data. When GP-extracted features are used, there is improvement in both ANN and SVM classifiers.

4.4.4 Comparison to Genetic Algorithms

The classification performance results displayed in Table 4.6 for six bearing conditions were obtained by ANN using 18 plain statistics feature sets [1], GA with

Features Type	ANN Best Perf.(%)	SVM Best Perf.(%)
4 Plain stat.	90.8	91.2
4 High pass Filter	85.4	85.0
4 Low pass Filter	87.1	16.7
4 Difference	45.4	35.6
4 Sum	65.1	90.5
4 Conventional	91.5	92
4 GP features	96.5	97.1

Table 4.5: classification success using the ANN and SVM classifiers with the different features.

Pattern Recognition Systems	Original Feature/ANN	GA/ANN	GP/ANN
No. of features	18	12	5
Perf.(%)	84.4	95.4	96.7

Table 4.6: Classification success (%) for Original feature set with ANN, Feature set selected by GA with ANN, Feature set generated by GP with ANN.

best ANN using 12 statistics feature sets [1], and GP with best ANN using 5 feature sets, respectively. The 18 plain statistics feature sets were taken based on the higher order statistics, which have been found to be useful in the identification of different problems in condition monitoring. The comparison results demonstrate that GP with ANN has the best classification performance of 96.7%, which is slightly more than that by GA with ANN, and about 12% more than that by the straight ANN. In addition, GP reduces the number of required inputs for the classifiers. The superiority of GP-based feature extractor against other methods is clearly demonstrated through the overall classification successes

Summarising the results from above experiments, it can be said that, when classical feature extraction methods are employed, the classification performance changes by a large extend with the variation of classification method and data, while GP-extracted features maintain a constantly high level of success. The superior performance of GP against other methods is clearly demonstrated through the overall classification successes.

4.5 Discussion

Based upon the experimental results, it can be said that using features generated by GP, both the ANN and SVM classifiers see improvement in classification accuracy and robustness, compared with those using classically developed features. GP derives feature selection from GA and also extends the abilities to implement feature extraction and generation, which combine different nonlinear functions to achieve the discrimination capability. The available feature space in GP is much larger than that in GA, where the one-dimensional feature space is pre-defined before the evolutionary process and the possible combination is constructed by randomly selecting candidates from the feature set. Rather than pure selection, GP produces new features by combining terminators and operators.

Generally speaking, a useful feature must have some discrimination characteristic, but at the same time, inevitably contains some noise and interference. Features may have some overlapping information, which introduces redundance in the feature space. During the natural selection process of GA, classification-useful information is retrieved in the form of a minimum feature set, but without changing the property of individual features. GP substantially alters the process by reconstructing features. During the evolutionary process, the features with maximum discrimination ability but minimum interference will eventually turn out to be the survivors. The maximisation of discrimination information in each feature results in the further reduction of dimensionality. In addition, in scenarios where the original features are not well prepared for the problem, GP has the advantage of generating useful features directly from the raw data, rather than the selection of pre-defined features in the GA.

In terms of classification success, there is a 12.3% increase from original feature set with ANN to GP/ANN and a slight increase from GA/ANN to GP/ANN.

This is due to two reasons. First, interfering features are filtered out during the evolution by the fitness measure. Only features with discrimination ability are remained for the classification thus reducing the pressure of the classifier to handle interference. From Table 4.6, it can be observed that the number of features required for the problem sees a reduction to 5 in GP from 18 in straight ANN and 12 in GA/ANN respectively. Furthermore, the powerful searching capability of GP is demonstrated by the smaller number of features required for achieving similar or even better classification success rate (see Tables 4.2 and 4.4). The reduction in dimensionality required to describe the problem indicates an improvement in multi-category classification performance.

In terms of computational complexity, the time consumed in computing is reduced by using a fast fitness measure based on the Fisher criterion, compared with those using classification results as the fitness.

4.6 Conclusion

In this chapter, a GP-based feature generator is proposed for the generation and extraction of multiple features from the raw vibration data for the problem of bearing condition classification. Based upon these results, it can be said that the discrimination capability of GP-generated features assists the ANN classifier to enhance the classification performance in the problem of bearing condition monitoring. More importantly, major faulty conditions are well separated in the feature-processed data, and provides an improvement from original feature set with ANN to GP extracted features with ANN. In most of experiments the classification success in GP/ANN is higher than that in GA/ANN using similar number of features.

GP is a powerful and efficient tool for the automatic feature generation. Using

features extracted by GP, the ANN and SVM classifiers see an improvement in classification results, compared with those using features extracted by classical methods. It is also shown from the extraction results that GP is not only capable of enhancing the classification performance, but also reducing the dimensionality required to describe the problem. Thus it can be a more economical implementation to the problem, compared to GA. The reduction is significant, compared with the number of original features and GA selected features. Furthermore classification performance obtained from GP-extracted features are reasonably robust. GP produces results in a direct tree representation, which allows an understanding of how it works. Generally speaking, longer evolved and more complex (larger terminator pool, operator pool and bigger tree depth) features have better overall discrimination capability.

Due to the fact that ANN is a required component in the implementation of GA-based feature selection, it needs to be conducted every time when individual is being evaluated. It requires a large amount of computation for calculating the classification success. Comparing the computation demands in the experiments for GA/ANN and GP/ANN respectively, it can be said that GP with this configuration for bearing condition monitoring problems is less time-consuming than that using GA with the same configuration. Together with the reduction in dimensionality, it makes GP a more practical realisation for the problem.

Although GP can successfully generate features for improving the accuracy of bearing condition monitoring, the method proposed in this chapter still requires several runs of GP to obtain multiple features for the multi-class problem. The computational demand is larger than that by a system needing only one run for multi-class classification, which will be addressed in the later chapter.

4.7 Summary

This chapter proposes a GP-based framework for the generation of multiple features for an industrial pattern recognition problem–bearing condition monitoring. The method for data acquisition and characteristics of the signal due to different mechanical conditions are described. Experiments are conducted, in order to evaluate the performance of the feature generation and classification algorithm. The performance of the conventional feature set, GA selected feature set and GP generated feature set are evaluated using two classifiers, namely MLP and SVM. The GP generated features demonstrate superior performance over other methods. Furthermore this chapter has addressed the advantages and capability of proposed GP algorithm for the application.

Chapter 5

Feature Generation for Dual-Class Problems

5.1 Introduction

The chapter is aimed at designing a GP-based feature extractor for dual-class pattern recognition problems. Before jumping into the main framework proposed for the problem, it will be useful to address the issues related to traditional feature extraction techniques, such as Fisher Linear Discriminant Analysis (FLDA) [28]. For many years, FLDA and its variants are among the most popular methods for linear discriminant analysis, initially designed in the domain of statistic analysis and later adapted for pattern recognition problems [29, 117, 118]. It is a powerful and efficient tool in linear feature space, where parametric functions are sufficient to distinguish between clusters. The algorithm only requires the calculation of mean and variance hence it can be very efficient during long evolutionary processes. This could be beneficial to the feature extraction because the computation time consumed in each generation can be an important factor for satisfactory results.

In this chapter, a modified version of FLDA is designed to improve certain aspects of Fisher criterion. The FLDA-based methods will play an important role in the GP based feature extraction system, which will be addressed in the following context. GP as an evolutionary computation method, provides a learning paradigm for feature generation/extraction. In this design, a GP-based system is constructed to generate a single feature for breast cancer diagnosis and bearing fault detection. A modified Fisher criterion is developed to optimise the class distribution during the evolution process.

This chapter is organised as follows. Two data sets used in the experiments are described in Section 5.2. A classical linear feature extraction measure (Alternative-FLDA) is addressed at the beginning of Section 5.3. Inspired by the Alternative-FLDA, a novel feature extraction measure, namely Modified-FLDA, is designed in the same section. Furthermore, a feature generation system based on GP and various Fisher criteria is described. In Section 5.4, a number of experiments using the two data sets are conducted. The comparison of classification performance using features extracted by linear feature extraction measures (PCA, FLDA A-FLDA and Modified FLDA), nonlinear feature extraction measures (KPCA and GDA) and GP-generated features based on different Fisher criteria (FLDA, A-FLDA and Modified FLDA) are given. Finally, the discussions and conclusions based on the experimental results are presented in Section 5.5.

5.2 Experimental Data Sets

Two data sets, Wisconsin Diagnostic Breast Cancer (WDBC) data set from the UCI Machine Learning repository [119] and the Bearing Fault vibration data set, are used in all experiments.

5.2.1 Breast Cancer Diagnosis

Image Preparation

The Wisconsin diagnostic breast cancer (WDBC) data set is created by Wolberg *et al.*, at University of Wisconsin [120]. The diagnosis procedure begins by obtaining a small drop of fluid from a breast tumour using a fine needle. The image for digital analysis is generated by JVC TK-1070 colour video camera mounted atop an Olympus microscope and the image is projected into the camera with a 63× objective and a ×2.5 ocular. The image is captured by a ComputerEyes/RT colour frame grabber board (Digital Vision, Inc., Dedham MA 02026) as a 512 × 480, 8-bit-per-pixel Targa file.

Data Preparation

An active model located in the actual boundary of cell nucleus is defined as a snake. Ten different features from the snake-generated cell nuclei boundaries are extracted by following techniques:

- ***Radius***: The radius of an individual nucleus is measured by averaging the length of the radial line segments defined by the centroid of the snake and the individual snake points.
- ***Perimeter***: The nuclear perimeter is defined by calculating the total distance between the snake points.
- ***Area***: The nuclear area is defined by counting the number of pixels on the interior of the snake and adding one-half of the pixels in the perimeter.
- ***Compactness***: The $perimeter^2/area$ is used as the compactness of the cell nuclei.

- ***Smoothness***: The smoothness of a nuclear contour is quantified by measuring the difference between the length of a radial line and the mean length of the lines surrounding it.

- ***Concavity***: Concavity is defined as the severity of indentations in a cell nucleus. For a line connecting any two non-adjacent snake points, if the actual boundary drops inside the line, an indentation occurs and the distance to the line is a measure of the severity.

- ***Concave Points***: This feature is similar to Concavity but measures only the number, rather than the magnitude of contour concavities.

- ***Symmetry***: The length difference between lines perpendicular to the major axis to the cell boundary in both directions is defined as symmetry.

- ***Fractal Dimension***: The fractural dimension is an indication of the regularity of the nucleus. Higher values of the downward slops of the coastlines correspond to less regular contour and vice-versa.

- ***Texture***: The texture of the cell nucleus is defined by finding the variance of the gray scale intensities in the component pixels.

The mean value, the maximum value and the standard deviation of each feature are computed for each image. A set of 569 images has been processed, yielding a database of 30-dimensional points. Here, we randomly select, without replacement, 100 samples for the benign case, and 100 samples for the malignant case respectively. Two 30×200 matrices are obtained for training and testing purposes. A third matrix is created based on the target information. A corresponding 2×200 target matrix records the actual conditions, benign or malignant. Each column has two rows corresponding to the class labels. For example, sample 1

is collected from a benign case. The corresponding column in the target matrix will be $[1,0]^T$. Otherwise, the target column should be $[0,1]^T$.

5.2.2 Bearing Fault Detection

The bearing condition monitoring data set is the same as the one mentioned in the previous chapter. The data set includes a total of 160 examples of each condition, and a total of 960 raw data examples over six conditions, which include four faulty conditions: Inner Race (IR) fault, Outer Race (OR) fault, Rolling Element (RE) fault and Cage (CA) fault; and two normal conditions: Brand new (NO) and slight worm but normal (NW) bearing. In this chapter, we only deal with dual-class problem. Hence the aim of the classification is to distinguish between normal conditions and faulty conditions.

In terms of the pre-processing of the raw data, in addition to the basic statistical methods used in the previous chapter, spectral data are calculated as an indicator of rotation-related faults. This is based on the observation that many of the machines being monitored are rotational and a large number of faults are frequency related. The spectral data has been one of the most effective indicators used in bearing fault monitoring.

The experimental data are processed by taking a simple 32 point Fast Fourier Transform (FFT) of the raw vibration data for each of the two channels sampled. A total of 33 values are obtained for each channel. These are then stored as a column vector of 66 values, which are used as the input data set. Two 66×960 matrices are obtained for training and testing, with 640 examples in faulty conditions and 320 in normal conditions. A corresponding 2×960 target matrix records the actual condition of the machine. Each column has two rows corresponding to the class labels. For example, sample 1 is collected from a normal condition. The

corresponding column in the target matrix will be $[1,0]^T$. Otherwise the target column should be $[0,1]^T$.

5.3 Feature Extraction Methods

In the preliminary chapter, the basic principle of FLDA has been addressed in the domain of feature extraction. As a classical feature extraction method, FLDA has evolved extensively into many varieties, among which Alternative-FLDA (A-FLDA) [118] has been considered to exhibit more power than conventional FLDA. To this regard, A-FLDA will be described first in this chapter and a modified version of FLDA is further developed.

5.3.1 Alternative-FLDA

A-FLDA is an improved version of FLDA for binary-class problems by replacing the original between-scatter with a new scatter measure [118]. This new scatter is described in the following text.

For c classes, the ith observation vector from class k is defined by $\mathbf{x}_i^{(k)}$, where $1 \leq k \leq c$, $1 \leq i \leq N_k$, and N_k is the number of observations from class k. The within-class covariance matrix is defined by

$$\mathbf{S}_W = \sum_{k=1}^{c} \mathbf{S}^{(k)}, \tag{5.1}$$

where

$$\mathbf{S}^{(k)} = \sum_{i=1}^{N_k} \left[\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)}\right] \left[\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)}\right]^T \tag{5.2}$$

The between-class covariance matrix is defined by

$$\mathbf{S}_B = \sum_{k=1}^{c} N_k \left[\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right] \left[\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right]^T \tag{5.3}$$

where $\boldsymbol{\mu}^{(k)}$ is the vector mean of class k, and $\boldsymbol{\mu}$ is the global mean.

For a dual-class problem, the new between-class covariance matrix of A-FLDA is defined by,

$$\mathbf{S}'_B = \frac{1}{N_1} \sum_{i=1}^{N_1} (\mathbf{x}_i \mathbf{x}_i^T) - \frac{1}{N_2} \sum_{j=1}^{N_2} (\mathbf{x}_j \mathbf{x}_j^T) \tag{5.4}$$

where $\mathbf{x}_i$, $\mathbf{x}_j$ denotes the ith, jth observation of class 1 and class 2, and N_1 and N_2 are the numbers of observations of class 1 and class 2 respectively. The projections $\mathbf{w}$ for the dual-class problem using new between-class scatter measure can be obtained by maximising

$$J(\mathbf{w}) = \frac{|\mathbf{w}^T \mathbf{S}'_B \mathbf{w}|}{|\mathbf{w}^T \mathbf{S}_W \mathbf{w}|} \tag{5.5}$$

where $\mathbf{S}_W$ is defined by Equation (5.3). The details of the algorithm and the theory behind the design can be found in [118].

5.3.2 Modified-FLDA

The limitation of FLDA lies in the within-class scatter which is not able to measure the distribution of observations accurately when the data clusters have significant overlaps. Although the computational demand of FLDA is relatively low, its drawback becomes obvious when it is used as a measure of class separation.

A large ratio of between-class scatter over the within-class scatter may be due to that the two classes do not overlap each other, which is desirable. However, it can also be a result of overlapping classes with a large separation of cluster centres.

Generally speaking, overlap is the major reason for misclassifications. The higher ratio of between-class scatter over the within-class scatter is not always helping the classification. In some cases, the classification performance may even drop, as observed in many experiments.

To overcome the limitation of FLDA and measure the genuine distribution of each pattern, we develop a new within-class scatter that uses the distance between any two patterns belonging to the same class instead of the variance.

For two-class problems, the new within-class scatter is defined by Equation 5.6,

$$\mathbf{S}'_W = \sum_{i=1}^{N_1}\sum_{j=1}^{N_1}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T + \sum_{i=1}^{N_2}\sum_{j=1}^{N_2}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \tag{5.6}$$

where N_1 and N_2 denote the numbers of samples in class 1 and class 2. Vector $\mathbf{x}_i$ denotes the ith observation. With Equation (5.6) employed as the within-class scatter, the sum of all the distances will act as an indication of degree of separation within one class.

By using this measure, a feature with all the observations concentrated within an area will be more competitive than those with a distribution of most of samples concentrated within a smaller area but a few scattered away from the centre. The desired goal of this scheme is to force the algorithm to search for more robust solutions rather than those having only good performance in part of the data.

The modified-FLDA is defined to maximise the objective function

$$J(\mathbf{w}) = \frac{|\mathbf{w}^T\mathbf{S}_B\mathbf{w}|}{|\mathbf{w}^T\mathbf{S}'_W\mathbf{w}|}. \tag{5.7}$$

5.3.3 A GP-based System

In order to reduce the computational complexity in classifiers, the design of fitness measure in this chapter, again, intends to look for answers in statistical analysis. Under such perspective, three measures, Fisher criterion, Alternative Fisher criterion and a Modified Fisher criterion, are utilised and designed for the evaluation of statistical distribution by testing the between-class scatter over the within-class scatter. The discriminant analysis is incorporated into the GP paradigm for the task of fitness measure. During the evolutionary process, the between-class scatter is maximised and the within-class scatter is minimised. For simplicity, the fitness function used for each system has been reformulated for dual-class problems. They are all Fisher criterion based. Because of the single feature evolving structure of GP, these functions only need to work in a one dimensional feature space. Hence all the variable involved are scalars. This design simplifies the calculation and improves the computation speed.

Original Fisher criterion for GP (F-GP): Based on the discussion of LDA in Chapter 2 and Equation 2.3, the Fisher criterion for dual classes can be simplified as,

$$f^{(p,q)} = \frac{\|\mu^{(p)} - \mu^{(q)}\|}{\sqrt{\sigma^{(p)} + \sigma^{(q)}}} \tag{5.8}$$

where μ is the mean of each class, σ is the variance of each class. The result $f^{(p,q)}$ is a scalar, which is the ratio of the between-class separation and within-class concentration.

Alternative Fisher criterion(AF-GP): The alternative Fisher criteria for two-class is denoted by

$$f^{(p,q)} = \frac{|\psi^{(p)} - \psi^{(q)}|}{\sqrt{\sigma^{(p)} + \sigma^{(q)}}}, \tag{5.9}$$

where the new between-class scatter ψ is defined by

$$\psi = \frac{1}{N}\sum_{i=1}^{N} x_i^2.$$

.

Modified Fisher criterion(MF-GP): The modified Fisher criterion is defined by

$$f^{(p,q)} = \frac{\|\psi^{(p)} - \psi^{(q)}\|}{\sqrt{\phi^{(p)} + \phi^{(q)}}}, \tag{5.10}$$

where ϕ is defined by $\phi = \sum_{i=1}^{N}\sum_{j=1}^{N}\|x_i - x_j\|$.

Apart from the fitness function, the overall design of the GP-based feature extraction system follows the same idea presented in Chapter 3, including the design of terminator/operator pools and the genetic operations. The terminator pool is application-specific hence requires the knowledge to the problem. The related parameters for each problem have been selected and formulated in the previous dicussions.

5.4 Experiments and Comparisons

In this section, experiments are conducted for the feature extraction task of two dual-class problems. In these experiments, MLP, k-NN and MDC classifiers are utilised to examine the ability of different features. First, the Modified-FLDA is experimentally compared with some linear feature extraction methods (PCA, FLDA, A-FLDA). The first set of results present the comparison of classification success rates using one to five linear feature sets extracted by PCA, FLDA, A-FLDA and Modified-FLDA. Second, the feature generated by the GP-based system using Modified Fisher criterion (MF-GP) is compared with the original

features and extracted features using conventional methods, namely KGDA and KPCA in terms of classification success. In addition, the proposed system is compared with the same GP-based system, but using conventional Fisher criterion and A-FLDA as the fitness measures.

In order to compare the performance, as well as to examine the reliability, two data sets are chosen, the Wisconsin diagnostic breast cancer (WDBC) data set [119] and the bearing fault data set. A series of experiments are conducted based on these two data sets.

5.4.1 Feature Extraction Results

Results for Breast Cancer Diagnosis

The information in Figure 5.1 is obtained after running the GP-based system with population size 100, maximum tree depth 10 and terminating after the number of generations reaches 5000. Figure 5.1 shows the output of a single feature, generated from the original feature set with 30 dimensions, for the training data set and test data set respectively. There are 200 examples in total from two conditions, with 100 examples in the benign case and 100 examples in the malignant case. It can be seen from Figure 5.1 that the two conditions are very well separated from each other in training data set, and three examples misclassified in test data set.

Results for Bearing Fault Detection

Figure 5.2 is obtained for bearing fault detection problem by the GP-based feature generator with population size 100, maximum tree depth 5 and max number of generations equal to 2000. Figure 5.2 shows the output of the feature values after processing the original 66 feature sets. There are a total of 960 examples from two

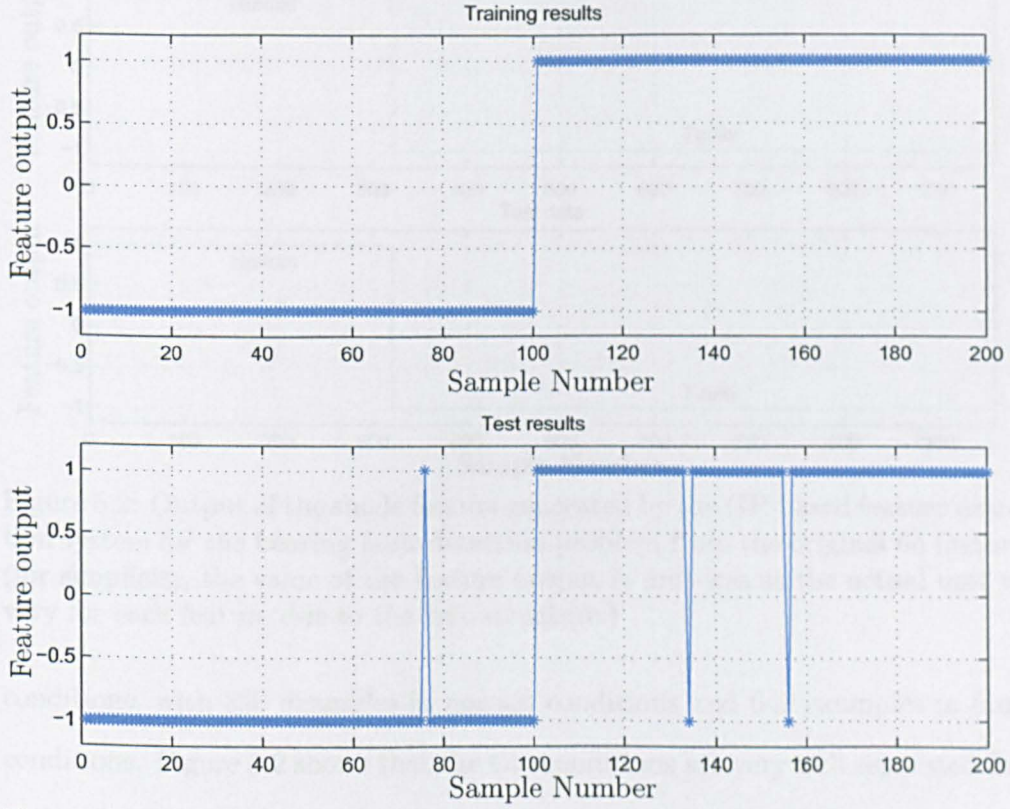


Figure 5.1: Output value of a single feature, generated by GP from the original 30 dimensional breast cancer feature set, showing 200 examples in each of the training set and test set (for simplicity, the value of the feature output is unitless as the actual unit will vary for each feature due to the tree structure).

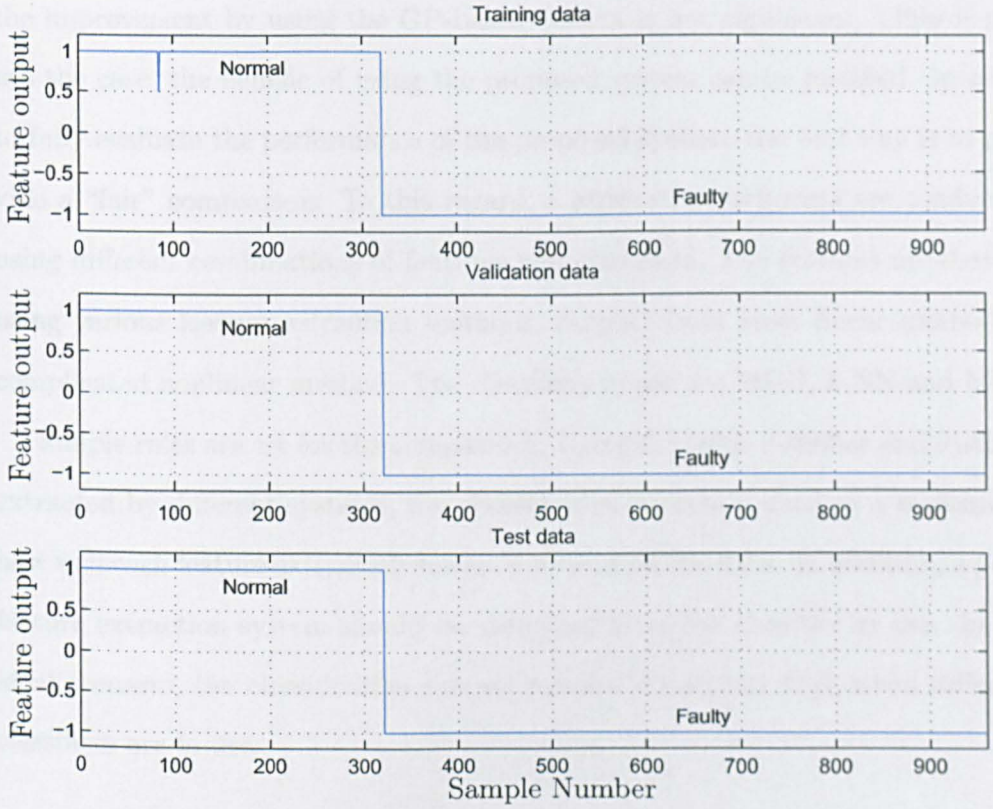


Figure 5.2: Output of the single feature generated by the GP-based feature extraction system for the bearing fault detection problem from the original 66 features. (for simplicity, the value of the feature output is unit-less as the actual unit will vary for each feature due to the tree structure.)

conditions, with 320 examples in normal conditions and 640 examples in faulty conditions. Figure 5.2 shows that the two conditions are very well separated from each other by one GP generated feature in the test data.

5.4.2 Classification Results

Feature extraction is just the preprocessing of data before the classification. The ultimate goal in pattern recognition is classification. Although the feature extraction results presented in the previous section demonstrate that the two classes are well separated from each other, it is not necessarily the case in the original feature space. If the classes can be easily separated in the original feature space,

the improvement by using the GP-based system is not significant. Only if it is not the case, the benefic of using the proposed system can be justified. In order to fully evaluate the performance of the proposed system, the best way is to provide a "fair" comparison. To this regard, a series of experiments are conducted using different combinations of features and classifiers. The features are derived using various feature extraction methods, ranging from basic linear method to complicated nonlinear method. The classifiers in use are MDC, k-NN and MLP.

Simple rules are set for the comparison. Using the same classifier and features extracted by different systems, the classification success is used as a measure of how well each feature extraction system performs on the data. In addition, a good feature extraction system should be independent of the classifier in use. In the ideal scenario, the classification success remains constantly high when different classifiers are in use.

Using features derived from linear systems

Four linear feature extraction systems have been developed, namely PCA, FLDA, A-FLDA and Modified-FLDA. The performance achieved by each classifier is listed in details in Table 5.1. The first column lists the pattern recognition systems in use. The second column reads the number of features in use to achieve the classification accuracy for the breast cancer detection problem. The third column shows the number of features and classification accuracy for the bearing fault detection problem. For the breast cancer detection problem, the results show that, using the four features extracted by Modified-FLDA as the inputs to MDC, the performance can be achieved at 89.5%. The classification accuracy of using three features extracted by FLDA is 89.0%, which is the same as that of using two A-FLDA features. The classification accuracy achieved by PCA is the lowest among those nonlinear feature extraction measures for both the breast cancer

Pattern Recognition System		Breast Cancer Detection		Bearing Fault Detection	
Feature Extraction by	Classifier	No. of Feature	Perf.(%)	No. of Features	Perf. (%)
PCA	MDC	2	88.5	1	66.6
FLDA		3	89.0	1	67.0
A-FLDA		2	89.0	1	78.8
Modified-FLDA		4	89.5	1	81.5
PCA	kNN	3	89.0	4	98.9
FLDA		4	88.5	6	97.5
A-FLDA		3	89.5	4	98.0
Modified-FLDA		3	89.5	3	98.0
PCA	MLP	5	90.0	4	96.7
FLDA		5	89.5	6	88.8
A-FLDA		3	90.0	4	93.7
Modified-FLDA		4	90.5	5	97.3

Table 5.1: The classification success rate (%) using various linear feature extraction systems and classification methods, for two pattern recognition problems, namely breast cancer diagnose and bearing fault detection.

diagnosis problem and the bearing fault detection problem. Using one feature extracted by Modified-FLDA with MDC as the classifier achieves the highest classification success of 81.5% in the data.

When kNN classifier is in use, it can be seen that the classification accuracy from A-FLDA feature set is the same as that using the feature set extracted by Modified-FLDA. The classification success is achieved at 89.5% for the breast cancer data and 98% for the bearing data. The classification success using FLDA with kNN is the lowest among all the pattern recognition systems in use. It is 88.5% for the breast cancer detection problem in a four dimensional feature space and 97.5% for the bearing fault detection problem in a six dimensional space.

It can be observed that using features extracted by Modified-FLDA with MLP leads to higher classification accuracy, with the highest performance achieved at 90.5% by the four features extracted from the breast cancer data and 97.3% by the five features extracted from the bearing data.

Using features derived from nonlinear systems

A range of nonlinear feature extraction systems have been put into place for comparison, including KPCA, GDA, FLDA combined with GP, A-FLDA combined with GP and Modified FLDA combined with GP. A number of experiments are carried out to evaluate the performance of different systems. The same line-up of classifiers are used for the classification task. Specifically, they are MDC, kNN and MLP. For a complete comparison, classifiers using the original feature sets are also registered in terms of classification performance. Table 5.2 lists all the classification results in details.

All the experiments involving the GP-based feature extractor or MLP classifier are repeated 50 times to account for the randomness in the systems, for example, the initial weighting coefficients of the neurons. The best, average and standard

Pattern Recognition System		Breast Cancer Detection (%)				Bearing Fault Detection (%)			
Feature extraction by	Classifier	No. of Feat.	Best	Avg.	Std.	No. of Feat.	Best	Avg.	Std.
Original Features	MDC	30	84.0	-	-	66	66.6	-	-
KPCA		1	94.5	-	-	1	71.2	-	-
GDA		1	93.5	-	-	1	100	-	-
F-GP		1	98.5	97.4	1.6	1	100	97.8	2.6
AF-GP		1	98.5	97.3	1.4	1	100	98.2	1.3
MF-GP		1	99.0	97.5	1.6	1	100	98.5	1.6
Original Features	kNN	30	87.5	-	-	66	99.2	-	-
KPCA		1	85.5	-	-	1	69.2	-	-
GDA		1	93.0	-	-	1	100	-	-
F-GP		1	97.5	96.6	1.1	1	100	98.5	2.3
AF-GP		1	97.0	96.4	1.5	1	100	98.8	1.5
MF-GP		1	97.5	96.8	1.2	1	100	98.6	1.7
Original Features	MLP	30	97	96.2	1.7	66	97	96.7	2.4
KPCA		1	90.0	88.6	4.3	1	84.4	82.8	1.1
GDA		1	93.5	93.0	2.5	1	100	99.6	3.8
F-GP		1	98.5	93.6	6.4	1	100	98.7	3.3
AF-GP		1	98.5	94.3	7.2	1	100	98.4	2.5
MF-GP		1	99.0	94.6	5.6	1	100	98.1	2.2

Table 5.2: The classification success (%) using various nonlinear feature extraction systems and classifiers for the breast cancer detection problem and the bearing fault detection problem.

deviation of the classification success rates are calculated to represent the overall performance of each system.

When MLP is used as the classifier, the best classification success by using the feature extracted by the hybrid system of GP and A-FLDA is achieved at 99.0%, which is the highest for breast cancer diagnosis problem. The average performance is also relatively high among all systems. For the same classifier, the best classification accuracy by using the feature generated by the hybrid system of GP and FLDA is achieved at 98.5%, which is the same as that of using one feature generated by the hybrid system of A-FLDA and GP. The classification accuracy of using one feature extracted by KPCA is the lowest with only 90% success. The average performance follows the same trend. It is interesting to see that the original feature set achieves the highest average performance and exhibits the smallest variation in repeating experiments.

For the bearing fault detection problem, using one feature generated by FLDA + GP, A-FLDA + GP, Modified FLDA + GP and one KGDA-extracted feature all achieve very high classification success, in both average and the best results. Directly using 66 original features achieves 97% classification success. The classification result using one GDA-extracted feature is the lowest among all with only 84.4% success.

When kNN classifier is in use, compared with the classification result using original features and nonlinear features for breast cancer diagnosis, the classification accuracy of using features generated by the GP-based systems (FLDA + GP, A-FLDA + GP and Modified-FLDA + GP) are remarkably higher, with the highest being 97.5% obtained by using a single feature generated by the hybrid systems of GP + FLDA and GP + Modified-FLDA. The best accuracy that A-FLDA + GP system achieves is 97.0% which is the lowest among all three GP-based systems. However, it is still higher than that using 30 original features

when MLP classifier is in use. The average performance is fairly close to the best performance, meaning that the GP-based systems perform consistently.

The classification accuracy of using KPCA with kNN is the lowest, 85.5%, among all pattern recognition systems. For the bearing fault detection problem, the classification successes achieved by one GDA extracted feature, one F-GP extracted feature, one AF-GP extracted feature and one MF-GP extracted feature are the same at 100%. Using 66 original features achieves 99.2% classification success. The classification result of using one KPCA-extracted feature is only 69.23% success, which is the lowest among all the pattern recognition systems in use.

With no doubt, the MLP and kNN are more powerful classification tools than the simple MDC classifier. However the simplicity of MDC can be an indirect measure of the quality of the features in terms of class separation. For the breast cancer diagnosis task, using one feature generated by the Modified-FLDA + GP system achieves the best classification result of 99% success. The classification accuracy of using FLDA + GP system is 98.5%, which is the same as that of using A-FLDA + GP system. KPCA achieves 94.5% as the best classification result, which is 1% higher than that of using one feature extracted by KGDA. The original 30 features produce the lowest classification success of 84% among all the pattern recognition systems in use. Using 66 original features for the bearing fault detection problem achieves only 66.6% classification success. One KPCA-extracted feature also performs not very well with only 71.15% classification success. In contrast, the classification results by GP-based systems provide very satisfactory results of 100% success, which is same as that using one GDA-extracted feature.

5.5 Conclusion

In this chapter, MLP, kNN and MDC are applied to examine the performance of feature sets extracted/generated by various feature extraction algorithms. As an improved version of FLDA and A-FLDA, Modified-FLDA overcomes the limitation of FLDA and has better generalisation ability than both of them for breast cancer diagnosis and bearing fault detection problems. In addition, linear feature extractors with MLP perform better than those using kNN and MDC. Nonlinear feature extractors: KGDA, FLDA + GP, A-FLDA + GP and Modified-FLDA + GP achieve 100% accuracy for the bearing fault detection problem in all three classifiers. For the feature set generated for breast cancer diagnosis, GP provides more than 97% classification success on all three algorithms (FLDA + GP, A-FLDA + GP and Modified-FLDA + GP). Overall, Modified-FLDA + GP system performs the best among all GP-based systems. Nonlinear feature extraction methods perform better than linear feature extraction methods.

Generally speaking, in pattern recognition problems, there is a reliance on the classifier to identify the discrimination information from a large feature set. In this work, GP as a machine learning method is proposed for nonlinear feature generation. A modified Fisher criterion (Modified-FLDA) is developed to overcome the limitation of original Fisher criterion, and applied to help GP select the best members among a large number of candidates. During the natural selection process, the feature is optimised to maximise the between-class scatter and minimise the within-class scatter of pattern vectors. Hence, this approach is able to learn directly from the data just like classical feature extraction methods (such as PCA, FLDA, A-FLDA), but in an evolutionary process. Under this framework, an effective feature can be formed for pattern recognition problems without the knowledge of probabilistic distribution of data. From the experimental results it

can be seen that the Modified-FLDA outperforms other linear classical feature extraction methods. With the support of a simple classifier MDC, GP combined with Modified FLDA outperforms the other two GP-based feature extractors (FLDA + GP and A-FLDA + GP), as well as the thirty original features with MLP. This indicates an advantage of the Modified-FLDA + GP system over classical methods for these data sets. Apart from the improvement in classification success and robustness, one important aspect of this approach is the reduction of dimensionality required to describe the problem compared with classical feature extraction/selection methods

5.6 Summary

This chapter first introduces two recently developed linear feature extraction methods based on FLDA, namely A-FLDA and Modified-FLDA. Based upon these two techniques, a GP-based feature extractor is constructed for generating single feature to solve general dual-class pattern recognition problems. Two experimental data sets are utilised for evaluating the performance of different feature sets. GP-generated features based on the proposed algorithm are obtained for breast cancer detection and bearing fault detection problems respectively. A series of experiments are conducted to evaluate the system performance. The Modified-FLDA is demonstrated to be performing well among other linear feature extraction methods, namely FLDA, A-FLDA and PCA for dual-class problems. Apart from the classification success, it exhibits superior robustness and good performance independent of classifiers.

Chapter 6

Single Feature Generator for Multi-Class Problems

6.1 Introduction

As mentioned in the previous chapter, traditional treatment of multi category pattern classification problems is to split it into multiple dual-class problems and derive a decision based on the integration of all dual-class results. Figure 6.1 shows the breakdown of a typical multi-category classification system. The role of the feature extraction block is to provide useful features, which contain discrimination information, to the function block. Each function in the function block consists of the feature inputs of each dual-class problem and a classifier for the associated problem. The output of each function can be either binary or continuous output depending on the design of the decision making block, which makes final decision on the class label based on the outputs of all the dual-class results. There exist different ways of splitting a multi-category classification problem, such as the one-against-all scheme and the between-every-two scheme.

The decision can be made based on voting and/or winning depending on the

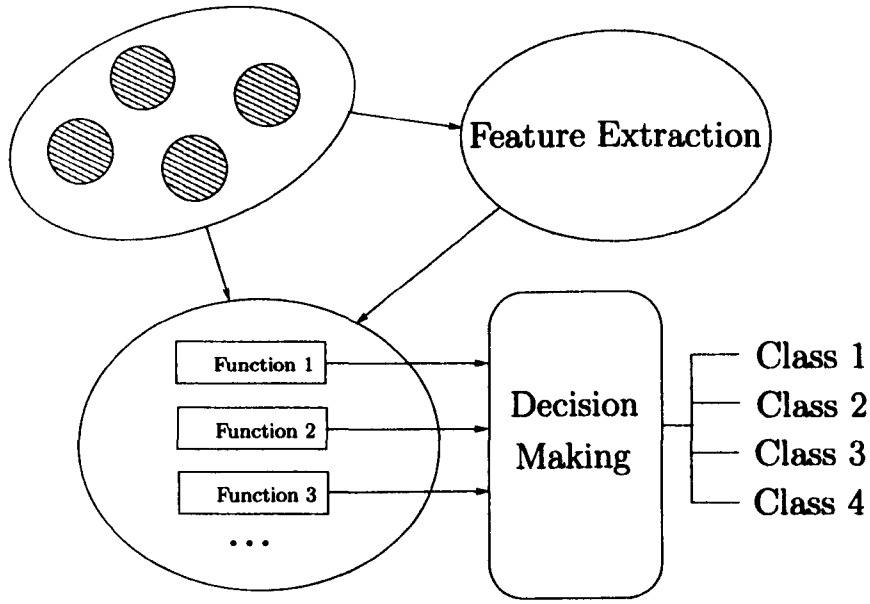


Figure 6.1: Breakdown of a general multi-category pattern classification problem.

output of the dual-class results. In case of one-against-all scheme, ideally, only one of the outputs from the dual-class classifiers should be positive while all the others remain zero or negative. However there are scenarios where multiple classifiers signal positive, which results in a conflict in the class assignment. In case of between-every-two scheme, ideally, the candidate with the maximum number of votes wins. However, the binary output of classifiers can not help when two candidates hold the same number of votes. These are inevitable when combining multiple outputs to provide a single class assignment.

These conflicts are usually resolved under a resolution, which requires an additional output from the classifiers. This output should be a normalised continuous value indicating the level of association to the class or a level of confidence. When the class assignment can not be solved by voting, conflict resolution is activated. There are schemes that only rely on the continuous output of each classifier for decision making. However they require that all the classes are well-balanced so that each classifier output floats within the same range and the same value from

different classifiers should indicate the same level of association.

Genuine multi-category classifiers are not uncommon, such as kNN and MDC, which require the assumption of statistical distributions. Taking the MDC for instance, the class centres are statistically estimated by averaging the training data set. The class assignment is conducted by a simple class associate level, specifically the distance to the class centres. The shortest distance wins. In fact, it can be considered as a classifier which only outputs distance and above-mentioned one-against-all voting scheme.

kNN is an typical example of classification by voting. The class assignment is conducted within a group of voters, which are believed to be closely related to the candidate based on the measurement of distance. The maximum number of votes determine the class label of the candidate. Again, it has to be assumed that the pattern from one class are distributed close to each other. Large overlap of distribution will result significant errors in the classification.

Both methods rely on distance as a kind of measure of the class association. Broadly speaking, all methods for multi-category classification problems use a kind of measure of class association to test likelihood of being each class and a decision-making scheme for the final class assignment. However, it should be borne in mind that distance measure is not always the true reflection of class association. Feature extraction of dual-class distribution usually provides more powerful and robust solutions. However, a decision making mechanism is required to integrate all the dual-class results. This may introduce discrepancy in the final output because all the classifiers should be well-balanced in terms of output. It is very difficult to maintain such balance because all the classifiers are separately designed and work independently for each dual-class problem. Also they work on different features. The output of each classifier by no means follows a universal standard that the decision making body can trust.

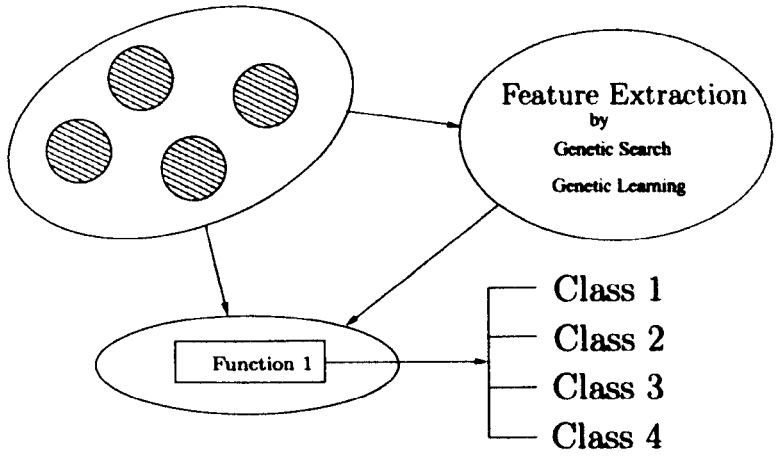


Figure 6.2: Breakdown of the proposed multi-category pattern classification system.

What we are trying to do here, is to get around of the decision making problem by searching for a "genuine" measure of class association. Ideally, it outputs a continuous output to indicate the likelihood of being each class. Class assignment is purely based on one particular measure. The idea is similar to MDC, simply the shortest wins. But the likelihood is not measured based on distance. Rather, the optimal measure is searched through a very large feature space by the genetic search algorithm. The solution can be illustrated by Figure 6.2.

The benefit is obvious. There is no need for decision making. The class association is measured in a one dimensional space. The likelihood of being one particular class can be told directly by looking at the distributions. This will be demonstrated in the later section. The challenging job of finding such good features has been conveniently allocated to the evolutionary process. Certainly this requires a dedicated fitness measure to evaluate the goodness of each feature and allow those discrimination genes to emerge early. It is believed that such design will be particularly interesting to industry problems, where usually fast and effective solutions are desirable. When problem occurs, it is easy to diagnose and solve. Complicated classifiers might be able to perform slightly better. From

a cost-effective point of view, this solution has the ability to provide a direct probe to the problem and the associated classifier is far more easier to implement.

GP is not the first time applied to multi-category pattern recognition problems. Kishore *et al.* first use GP for c-class ($c > 2$) pattern classification by modelling c number of dual-class problems [108]. A Genetic Programming Classifier Expression (GPCE) is evolved as a discriminant function for each class. A conflict resolution is set to resolve issue arising from multiple assignment of class labels from the dual-class classifiers. A decision-making logic is implemented to deal with conflicts. Analogy to Kishore's method, Chien *et al.* employ GP to generate dual-class discriminant functions using arithmetic operations with fuzzy attributes [121] and apply Z-value measure as the objective function of GP for multi-category classification problems [122]. An effective discerning mechanism is required to integrate the results.

The investigation of these GP-based methods reveals another benefit of our design. Generally, for c-class pattern recognition problems, a total of c features, in case of one-against-all scheme, or $\binom{2}{c}$ features, in case of between-every-two scheme, are required for all the dual-class classification tasks. Therefore, the running time for a c-class problem is c times or even $\binom{2}{c}$ times the single run. As an evolutionary algorithm, GP is very computationally intensive. Usually, it requires many thousands of runs for a good solution to emerge. A method of step-wise learning to design a multi-tree classifier, which consists of c trees for c-class pattern classification using only a single run of GP, has been proposed in [123, 124]. Under our design, only one run is required for any multi-category classification problem.

This chapter is organised as follows: Section 6.2 presents the proposed feature extractor using genetic programming based on Fisher Criterion. In Section 6.3, a number of experiments for multi-class pattern classification problems are

conducted. The comparison of classification performance using GP-extracted features against those by KGDA, PCA, and GA is presented. Finally, based on the experimental results, the advantage and limitation of GP-based feature extraction method is concluded in Section 6.4.

6.2 System Design

In this chapter, a new fitness measure based on Fisher criterion is developed within the GP paradigm to reduce the dimensionality required to describe the problem and to achieve the improvement of classification efficiency. As a result of the design of the fitness measure, the number of required features is reduced to one, allowing the simple and fast implementation of classifiers. It is believed that this is particularly beneficial to industrial applications, where cost-effective solutions are desired.

GP as a form of evolutionary algorithm is proposed as the primary method to extract nonlinear features based on Fisher criterion. The GP-based feature generator extracts the information from the real-valued parameter vector to create features based on the evolutionary algorithm. The survived features from the feature generator is used to provide the solution to the multi-class pattern recognition problems. At each stage of the evolutionary process, GP generates a population of features. Fisher criterion based fitness function evaluates the effectiveness of each feature in helping GP select the best solution. This approach provides a solution to obtain a single tree/feature by only a single run of GP. In other words, the method has the capability of transforming the feature vector with high dimensions into a single feature to reduce the computational complexity.

6.2.1 Fitness Function

The fitness function plays a key role in the system performance. It must have the ability to rate the performance of each population member effectively and accurately. At the same time it offers the stronger candidates higher chance of surviving. Ideally it does not require a large amount of computation as the increase of population size and the number of generations slows down the evolution significantly. Within a fixed period of time, a slow fitness measure may provide excellent assessment, however only a limited number of generations have been reached hence the feature space may not have been fully explored. The features derived may be far from optimisation. To this regard, the design of the fitness function is realised in this work as a tradeoff between accuracy and efficiency.

There have been some success [64, 110] in using the misclassification error as the fitness measure for multi-class problems. As these belong to a wrapper type approach, the computational cost depends on the structure of the classifier in use and the time consumed in training and testing the data. Indeed, classification error is one of the best measures in terms of accuracy since the ultimate goal of feature extraction is to improve the classification success. However, once efficiency is considered as a criterion in the evolution process, we should always explore broader literature for those measures providing descent accuracy as well as efficiency.

In this work, a Fisher-criterion based fitness measure is designed for the task. It tests the rate of between-class scatter over the within-class scatter for two adjacent classes in the histogram. Theoretically, the smaller rate, the better separability of classes within the feature space. The design is detailed in the following text. Please note that all the calculation is conducted in a one-dimensional space, hence all variables are scalars because the required number of features has been

reduced to "one" by the feature extractor.

For any two classes (p and q), the Fisher criterion can be defined by

$$f^{(p,q)} = \frac{|\mu^{(p)} - \mu^{(q)}|}{\sqrt{\sigma^{(p)} + \sigma^{(q)}}}, \tag{6.1}$$

where μ and σ are the class mean and class variance respectively.

It is well known that the Fisher criterion measures the distribution of between-class scatter over the within-class scatter. During the evolutionary process in search for candidates with larger fitness values, the between-class scatter is maximised and at the same time the within-class scatter is minimised. The treatment for the multi-class ($c > 2$) problem is more complicated by incorporating an individual-saturation mechanism. The evolutionary algorithm does not have the intelligence to take care of the simultaneous improvement of all classes. When an individual has a relatively high fitness value, it indicates that difference between any two classes is large since the magnitude of Fisher criterion value determines the degree of separation of two classes.

The proposed fitness function for c-class ($c > 2$) can be defined in following steps. Given a set of individuals or trees of GP $\{I_1, I_2, \ldots, I_r, \ldots, I_n\}$, where n is the total number of individuals/trees in each generation and I_r is the rth individual/tree, a corresponding fitness value F_r is assigned to rth individual/tree.

1. **Loop** $r = 1, 2, \ldots, n$

   (a) For individual I_r, calculate the mean of samples from each class. $\mu_1, \mu_2, \ldots, \mu_c$ are obtained.

   (b) Sort mean values in descending/ascending order to obtain sorted index $i = 1, 2 \ldots, c$.

   (c) Calculate $f^{(i,i+1)}$ $(1 \leq i \leq (c-1))$ for each adjacent pair of classes

based on Equation (6.1).

(d) Set $F_k = 0$;

(e) **Loop** $i = 1, 2, \ldots, (c-1)$;

i. **If** $f^{(i,i+1)} > T$, $F_k = F_k + 1$;

Else $F_k = F_k + f^{(i,i+1)}/T$;

End Loop i

End Loop k

2. $F_{max} = \max\{F_r, k = 1, \ldots, n\}$

3. **If** $F_{max} < (c-1)$, *I_r is put into the next generation.*

Else if $F_{max} == (c-1)$, *GP is terminated.*

The fitness function follows the procedure to evaluate each individual and to search for the best individual during the learning process of GP. The advantage of the fitness function is that only $(c-1)$ dual-class Fisher criteria values $f_{i,(i+1)}$ need to be calculated. The fitness function is a measure of separation, designed in such a way that the contribution of the Fisher criterion value $f_{i,(i+1)}$ is the same once it is larger than a certain threshold T.

The computational cost of this design is reduced significantly compared to the standard classifier approaches (such as MDC and kNN). The Fisher-criterion based fitness measure merely requires $2N$ sums and N multiplications.

6.3 Experiments and Results

In order to provide a fair and comprehensive comparison to the nonlinear feature extraction method, a series of experiments are conducted using different

experimental data sets. This section starts with the description of the feature generation and experimental data collection. The GP-based feature extraction results using different data sets are presented. These features are later employed by three different classifiers, specifically MDC, kNN and MLP, to examine the performance under different conditions.

6.3.1 Experimental Data sets

In order to demonstrate the performance and robustness of above-mentioned GP-based feature generation approach, eight data sets are chosen. They are listed in Table 6.1, which includes the details of the number of classes for each problem, original feature dimensions, the number of training examples and the number of test examples.

The Machine Condition Monitoring (MCM) data set is the same as the one used in Chapter 4. It is obtained by running a roller bearing machine over a series of sixteen different speeds and taking ten examples of data at each speed. This gives a total of 160 examples of each condition, and a total of 960 raw data examples over six conditions (IR, OR, RE, CA, NW NO) to work with. The experimental data are further processed by taking a simple 32 point FFT of the raw vibration data for each of the two channels sampled. A 66×960 matrix forms the terminator set to the GP. For each given matrix in the experimental data set, a corresponding target matrix is to record the actual condition of the machine. The Bearing Fault Detection (BFD) data set is a dual-class version of the MCM data set. It is only required to distinguish between normal conditions (NO, NW) and faulty conditions (IR, OR, RE, CA).

The ADMR (automatic digital modulation recognition) data set is chosen from the domain of telecommunications. Ten data patterns are simulated, including

Dataset	Number of classes	Number of features	Number of examples		Domain
			training	test	
Balance	3	4	313	312	Balance Scale Weight& Distance Database
Iris	3	4	75	75	Iris Plants Database
Lenses	3	4	11	13	Database for fitting contact lenses
BFD	2	66	960	960	Bearing vibration data for fault detection
MCM	6	66	960	960	Bearing conditions classification
Lung cancer	3	56	15	17	Lung Cancer Data
Zoo	7	16	49	52	Animal classification
ADMR	10	17	10,000	10,000	Telecommunication

Table 6.1: The experimental data sets used for the evaluation of the performance of pattern recognition systems.

ASK2, ASK4, BPSK, QPSK, FSK2, FSK4, QAM16, V29, V32 and QAM64. The details of the data generation is addressed in [74]. For this experiment, each modulation scheme provides 1000 examples. The modulated signals are then added with white Gaussian noise, giving SNR at -5 dB. Combining the 5 spectral features and 12 statistical features, three 17×10000 sets of data are obtained for training, validation and testing purposes. Because this ADMR is synthesised, it will be only used for feature extraction experiment to demonstrate the idea. It will be not used for any further classification task.

The other six data sets listed in Table 6.1 are openly available from the public domain. They cover a spectrum of different problems. Five data sets are chosen from the UCI (University of California Irvine) repository of machine learning database [119].

Table 6.2 lists the parameter σ and T value used in the fitness function in each experiment. Two conventional nonlinear feature extraction methods, KPCA and KGDA, are utilised for the purpose of comparison.

Three classifiers, MDC, k-NN and MLP are used for the further classification task. Results of classification success of each experiment will be compared.

6.3.2 Feature Generation Results

Figure 6.3 is presented here as examples for demonstrating the feature generated by GP for the MCM data. Figure 6.3 illustrates the feature-processed data. There are altogether 960 examples from six conditions with 160 examples for each class. It is extracted by the GP-based feature extractor with a population size of 100, a maximum tree depth of 15 and the threshold $t = 4.5$. It can be seen from Figure 6.3 that six conditions are well separated from each other by a single GP feature derived from the original set. Figure 6.4 illustrates the histogram of the sample

Dataset	KPCA		KGDA		GP	
	No. of features	Value of σ	No. of features	Value of σ	No. of features	Value of t
Balance	2	15	2	15	1	15
Iris	2	5	2	100	1	4
Lenses	3	4.5	2	5	1	15
BFD	1	8000	1	5000	1	50
MCM	5	230000	5	20000	1	4.5
Lung cancer	2	7	2	4	1	5
Zoo	6	3.5	6	5	1	50

Table 6.2: The $(c-1)$ features extracted by KPCA, KGDA and the single feature generated by GP, and the value of parameter σ of KPCA and KGDA, t value of GP in all the experimental data sets

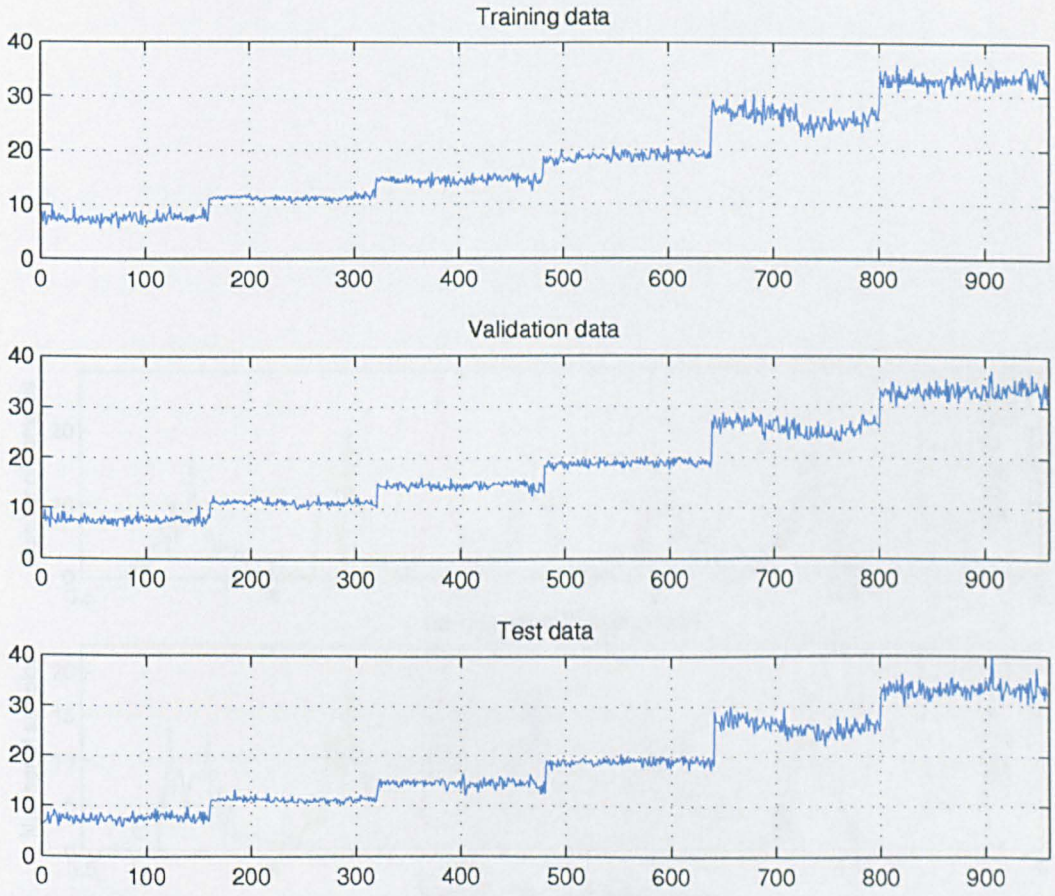


Figure 6.3: Example of GP generated feature using MCM data set. (for simplicity, the value of the feature output is unit-less as the actual unit will vary for each feature due to the tree structure.)

data.

For ten digital modulation schemes, one non-linear feature is generated at each SNR. Those features are extracted by GP with a population size of 100, a maximum tree depth of 15 and the threshold $T = 4.5$. There are altogether $10,000$ examples with each modulation providing $1,000$ examples. Figure 6.5 illustrates the distribution of values of the feature extracted by GP for ten modulation types at SNR $= -5$ dB. It can be seen from Figure 6.5(b) that class 1 to 6 are well separated even though this is not clearly visible from Figure 6.5(a). Figure 6.6 illustrates the histogram of the feature output. This experiment is repeated by

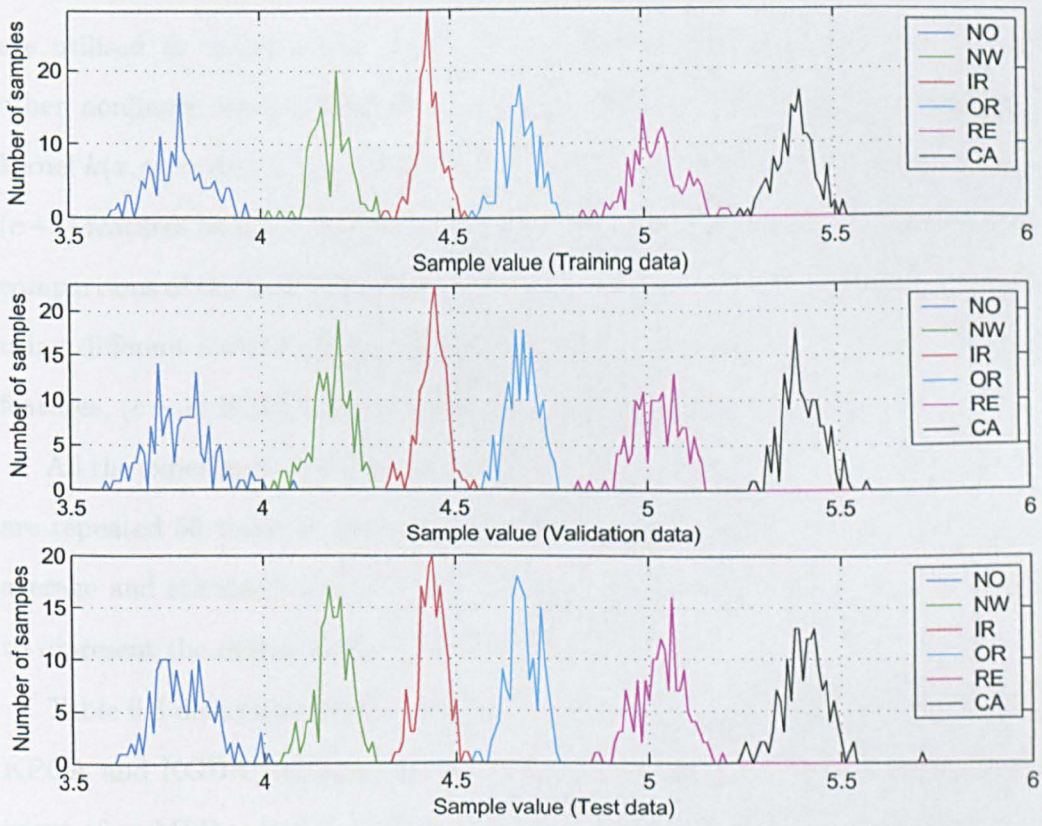


Figure 6.4: The histogram of GP generated feature for MCM data set.

generating a new feature by GP and similar level of distribution in the results are obtained.

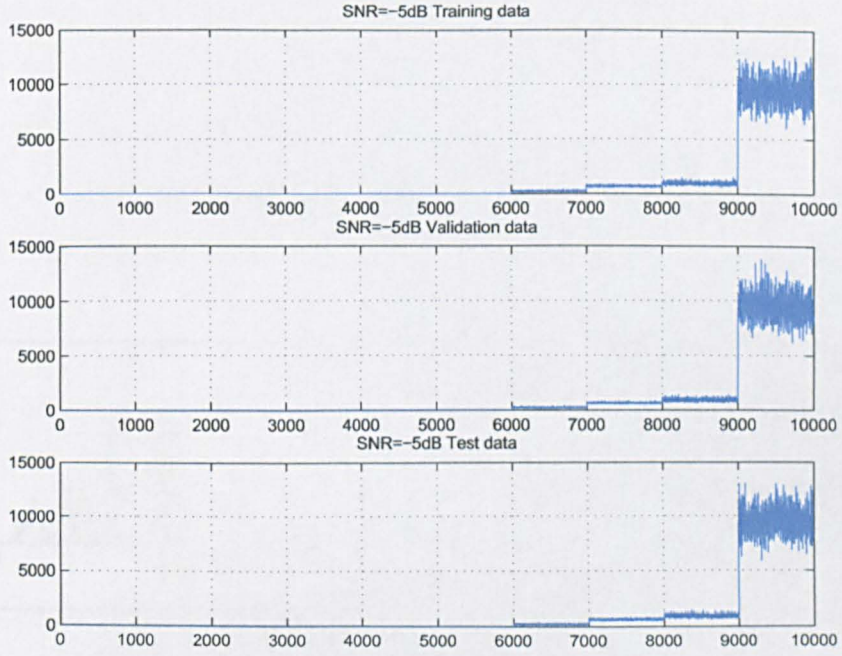
6.3.3 Classification Results

A number of experiments are carried out to evaluate the discrimination ability of features generated by GP. In these experiments, MLP, 1-NN and MDC classifiers are utilised to examine the ability of different features generated by GP and other nonlinear feature extraction methods (KPCA and KGDA). A Gaussian kernel $k(x, y) = exp(-\|x - y\|^2/2\sigma^2)$ is employed in KPCA and KGDA to extract $(c-1)$ features based on each set of original features. Tables 6.3, 6.4 and 6.5 show comparisons of the best classification results of MLP, 1-NN and MDC respectively using different feature sets including original features, $(c-1)$ KPCA extracted features, $(c-1)$ KGDA-extracted features and one GP generated feature.

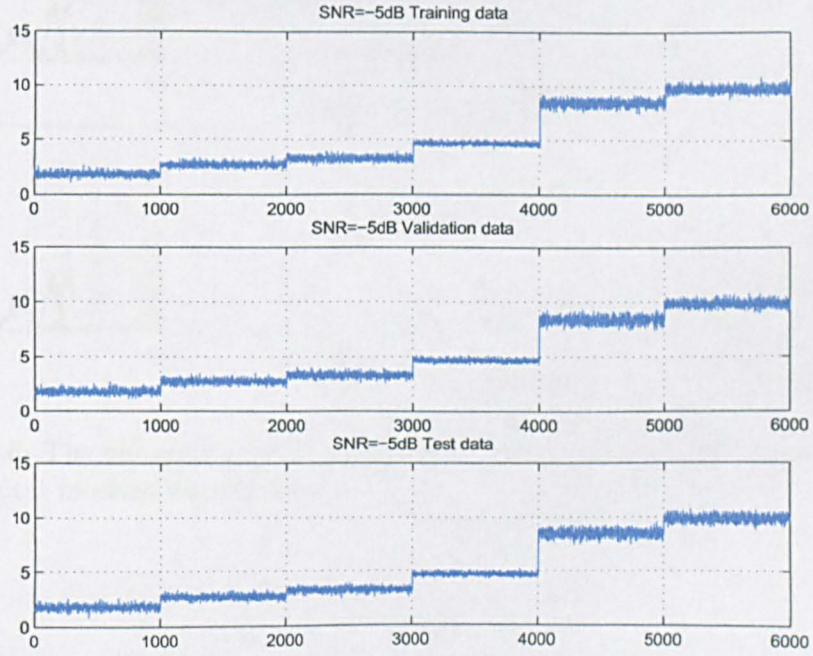
All the experiments involving the GP-based feature extractor or MLP classifier are repeated 50 times to account for the randomness in the systems. The best, average and standard deviation of the classification success rates are calculated to represent the overall performance of each system.

Table 6.3 demonstrates the comparison of the best classification results using KPCA and KGDA extracted feature sets and one GP generated feature as the input of an MLP on seven data sets. The results show that the feature generated from the GP algorithm offers the highest classification success, both in averaged and the best results, compared with other nonlinear feature extraction algorithms for all data sets. For the fault detection problem in BFD data set, the average classification accuracy of using the feature extracted by KGDA is 99.6%.

The classification accuracy, when using original feature sets as the input to an MLP, is comparable to that using the feature set extracted by KPCA.



(a) The distribution of the feature output value from all ten modulation schemes



(b) The distribution of the first six modulation schemes with a close-up range on y axis.

Figure 6.5: The output value of the GP evolved feature for the automatic recognition of ten digital modulation schemes, specifically, ASK2, ASK4, BPSK, QPSK, FSK2, FSK4, QAM16, V29, V32 and QAM64, with SNR=−5dB (for simplicity, the value of the feature output is unit-less as the actual unit will vary for each feature due to the tree structure.)

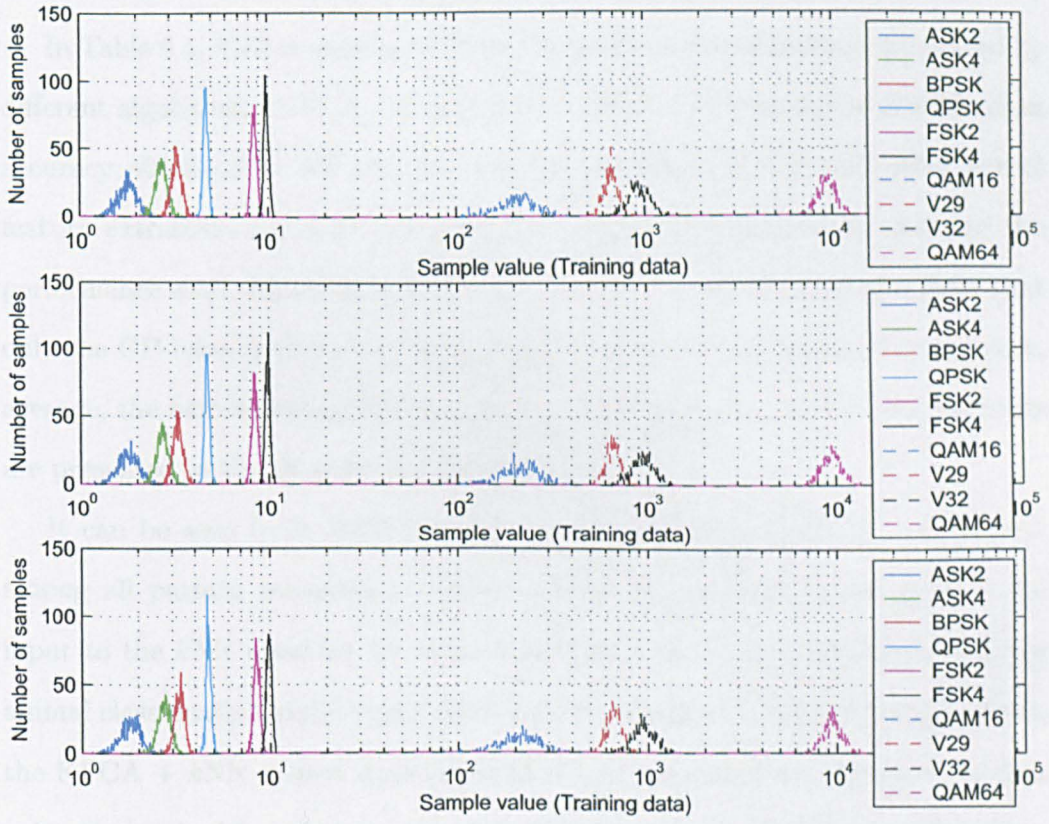


Figure 6.6: The histogram of GP generated feature for the automatic recognition of 10 digital modulation schemes.

It can be seen from Table 6.3 that KGDA/MLP performs better than the original features with MLP on BFD and lung cancer data sets. For the animal classification problem and fitting contact lens problem, original features and KGDA-extracted features achieve similar success. Remarkably GP-generated feature with MLP performs best on these data sets with the largest improvement of over 17% classification success compared to the original features.

In Table 6.4, kNN is used to evaluate the performance of features generated by different algorithms (KPCA, GDA and GP). Table 6.4 presents the classification accuracy obtained by different feature sets generated by GP and other kernel feature extraction methods using the kNN classifier employed to examine the performance and robustness of GP-based feature extraction approach. Note that only the GP-based system has randomness in terms classification success, hence, average, the best and standard deviation of the results in repeating experiments are presented in the table for the GP-based system.

It can be seen from Table 6.4 that the classification accuracy is the lowest among all pattern recognition systems when original features are used as the input to the kNN classifier for both the Balance and Lenses data sets. For the animal classification and bearing condition classification (dual and six) problems, the KPCA + kNN system does not achieve any improvement compared to that using the original feature sets with kNN. Moreover, single GP feature as the input to the kNN achieves the best classification success compared to other pattern recognition methods, with the largest improvement of over 21.7% compared to that achieved by original features in Lenses data set.

To further examine the dimensionality reduction capability of the proposed method, MDC as a simple classifier is employed for seven different classification problems. In these experiments, the same group of feature sets prepared by different feature extraction algorithms (KPCA, KGDA and GP) are used as the

Dataset	Orig. feat./MLP (%)			KPCA/MLP (%)			KGDA/MLP (%)			GP/MLP (%)			Diff. *
	Best	Avg.	Std.	Best	Avg.	Std.	Best	Avg.	Std.	Best	Avg.	Std.	
Balance	84.9	72.4	4.28	84.3	79.7	3.2	79.8	73	3.3	**98.4**	90.7	4.3	13.5%
Iris	96.0	90.6	10.1	97.3	92.3	7.3	94.7	91.0	6.9	**100**	95.9	8.3	2.7%
Lenses	84.6	54.9	19.1	61.5	45.9	8.7	84.6	58.8	17.9	**100**	81.9	10.6	15.4%
BFD	97.0	96.7	2.4	84.4	82.8	1.1	**100**	99.6	3.8	**100**	99.8	0.3	0
MCM	97.0	87.2	8.7	86.9	72.8	15.1	94.6	80.8	18.7	**99.9**	96.4	1.2	2.9%
Lung cancer	66.7	43.0	11.1	64.7	50.8	5.7	82.4	42.3	11.1	**100**	54.6	20.5	17.6%
Zoo	**98.1**	87.8	10.3	96.2	86.0	9.3	**98.1**	89	10.2	**98.1**	85.5	10.8	0

Table 6.3: The best classification accuracy (%) using original features, $(c-1)$ KPCA-extracted features, $(c-1)$ KGDA-extracted features and one GP-generated features respectively, with a MLP classifier on all the experimental data sets.

Dataset	Orig. feat./kNN	KPCA/kNN	KGDA/kNN	GP/kNN			Difference *
	Best classification accuracy (%)			Best (%)	Average (%)	Std. (%)	
Balance	65.7	71.5	77.9	**97.4**	89.4	5.7	19.5%
Iris	94.7	96.0	94.7	**97.3**	97.3	96.1	1.3%
Lenses	38.5	46.2	69.2	**90.9**	73.9	11.4	21.7%
BFD	99.2	79.7	100	**100**	98.7	0.4	0
MCM	98.4	85.7	92.8	**99.9**	96.2	1.5	%
Lung cancer	52.9	58.8	47.1	**60.0**	51.3	10	1.2%
Zoo	94.2	88.5	94.2	**98.1**	93.1	3.4	3.9%

Table 6.4: The best classification accuracy (%) using original features, $(c-1)$ KPCA-extracted features, $(c-1)$ KGDA-extracted features and GP-generated features respectively, with a kNN classifier on all the experimental data sets.

inputs to the MDC.

Table 6.5 illustrates the classification results of MDC using original features, KPCA extracted features, KGDA extracted features and a single GP-generated feature. From Table 6.5 it can be seen that the classification accuracy using the features extracted by KPCA and KGDA with MDC classifier is lower than that using original feature sets with MDC in almost all data sets, except BFD and zoo data sets. For the BFD data and zoo data sets, KGDA features outperform the original features. The overall conclusion is that GP generated feature with MDC performs the best on these data sets, with the largest improvement of over 11% compared to original feature in BFD data set.

In order to provide a fair comparison, Table 6.6 lists the results from the other researchers using the same data sets, but different pattern recognition systems. The first column is the data set in use. Column two and column three list the different pattern recognition systems used by different researchers and their best results. The last three columns present the classification success achieved by the GP-based systems.

The classification result of using "Balance" data set was obtained by using a REFNE (Rule Extraction From Neural network Ensemble) approach proposed by Zhou *et. al* [125]. A neural network ensemble is a set of neural network models taking a decision by averaging the results of individual models. REFNE utilises the trained ensembles to generate instances and then extracts symbolic rules from those instances, thus breaks the ties made by individual neural networks in prediction. The highest success is achieved at 93.6%, which is 0.3% higher than that using GP/MDC, 4.8% lower than that using GP/MLP and 3.8% lower than that using GP/kNN.

When a simple quantum neural network (QNN) [126] is in use, the classification result for "Iris" data set is 98.2%. This simple QNN operates pretty much

Dataset	Orig. feat./MDC	KPCA/MDC	KGDA/MDC	GP/MDC			Difference *
	Best classification accuracy (%)			Best (%)	Average (%)	Std. (%)	
Balance	84.3	72.1	67.6	**93.3**	88.2	5.4	9%
Iris	**97.3**	89.3	94.7	**97.3**	96.6	0.7	0
Lenses	72.7	46.2	69.2	**84.6**	72.6	6.9	11.9%
BFD	66.6	71.2	**100**	**100**	98.2	0.4	0
MCM	97.1	70.9	93.2	**99.7**	95.7	4.3	2.6%
Lung cancer	47.1	58.8	47.1	**66.7**	58.7	5.3	7.9%
Zoo	84.6	80.8	94.2	**98.1**	93.4	3.1	3.9%

Table 6.5: The best classification accuracy (%) using original features, $(c-1)$ KPCA-extracted features, $(c-1)$ KGDA-extracted features and one GP-generated features respectively, with a MDC classifier on all the experimental data sets.

like a classical ANN composed of several layers of perceptrons, except that the output of each percepton is binary. If the sum of the input nodes is above a threshold, the node goes high, otherwise it stays low. The network as a whole computes a function by checking which output bit is high. There are no checks to make sure exactly one output is high. This allows the network to learn data sets which have one output high or binary-encoded outputs. Using this simple QNN classifier, the classification result for "Lenses" data set is 98.5%. Both of them are lower than that using GP/MLP, and higher than that using GP/MDC and GP/kNN. The classification accuracy obtained by using GA/MLP [1] for bearing fault detection and condition monitoring is the same as that using GP methods.

For the classification of lung cancer data, a MDC on the optimal discriminant plane [127] is constructed with the ability to handle small number of data samples. The idea of optimal discriminant plane is to use the generalised eigenvectors solved by means of SVD perturbation as the optimal discriminant directions. By doing this, high performance classifier can be constructed on the discriminant plane in the case of a small sample size. The classification success for the lung cancer data is achieved at 100%. For the Zoo data set, a SVM classifier is constructed and achieves 97% classification success.

6.4 Conclusion

It is now clear from Figures 6.3 and 6.5 that the single feature obtained from our proposed method scatter the class distribution into non-overlapping groups in the chosen data sets. The clusters belonging to different conditions are well separated within a one dimensional feature space. This achievement is interesting as it provides the direct knowledge about the problem at hand and effectively reveals the distributions due to different classes. By setting thresholds at those local minima,

Dataset	Best classification accuracy (%)				
	Pattern recognition systems for comparison		GP + MDC	GP + k-NN	GP + MLP
Balance	REFNE [125]	93.6 [125]	93.3	97.4	98.4
Iris	Quantum Neural Networks [126]	98.3 [126]	97.3	97.3	100
Lenses	Quantum Neural Networks [126]	98.35 [126]	84.6	90.9	100
BFD	GA/MLP [79]	100 [79]	100	100	100
MCM	GA/MLP [128]	100 [128]	99.7	99.9	99.9
Lung cancer	Optimal Discriminant Plane/MDC [127]	100 [127]	66.7	60.0	100
Zoo	SVM	97 [129]	98.1	98.1	98.1

Table 6.6: The classification success achieved by other pattern recognition systems using the seven data sets.

it does not require a sophisticated classifier to tell the class labels. Thanks to the fitness scheme designed in Section 6.2.1, the patterns of classes are evenly distributed. We do not see extremely condensed distributions or sparse distributions, which may require the construction of complicated classifier for class separation. Thus under such feature extraction scheme no computationally complex classifier is required for successful classification. Instead, simple thresholds will be sufficient for the task.

Summarising all the results obtained from different approaches for the pattern recognition problem based on seven different data sets, GP-based approach exhibits accurate and reliable performance throughout all experiments with the chosen data sets. It is observed that GP is not only capable of reducing the dimensionality, but also achieving an improvement in the classification accuracy. Using the single feature generated by GP leads to the improvement of classification accuracy and robustness, compared with other sets of features extracted by KPCA and KGDA.

There have been some earlier attempts [64,110] to use GP to generate features, using classification success rate as the fitness values for multi-category classification problems. As these belong to a wrapper type approach, the computation for training a classifier for each individual is expensive.

In the earlier chapter, GP has been used to generate new features in a stand-alone manner, where multiple features were generated as opposed to a single feature and their fitness function is different from the fitness function in this approach. One of the major advantages of the approach in this chapter is that only one feature is required to solve the multi-class problems. What has been observed is that, during the evolutionary process, different classes appear to distribute very well in almost a non-overlapping manner. Consequently there is no need for a complicated classifier. Instead one can use a MDC or simply set thresholds.

For multidimensional pattern recognition problems, generally there is a reliance on the classifier to find the discrimination information from a large feature set in case of stand-alone MLP, or a well-chosen feature set in case of GA/MLP. In this chapter, a GP structure is proposed for multi-class nonlinear feature extraction problem based on Fisher criterion. The framework presented here enables the transformation of information contained in the data during an iterative process. Hence, the learning machine is able to learn directly from the data in the same way as conventional methods (such as FLDA and PCA), but under an evolutionary process. Under this framework, effective feature extraction can be achieved without the explicit knowledge of probabilistic distribution of data.

Although KPCA and GDA achieve similar classification success as GP/MDC in vibration data, the combination of GP and MDC is more cost-effective as it only requires one feature and a simple classifier. Furthermore, GP is less computationally demanding compared with GA for feature selection. While GA/MLP [1] takes a couple of days to work out a solution, GP achieves a similar level of success in only a few hours. The proposed method requires comparatively less computation, since it does not contain wrapper type classifier-based fitness measure.

From the different experiments presented in this chapter, it is demonstrated that the proposed GP framework performs reasonably well (either the best or equally the best). The classification success in these data sets are among the highest. It is an efficient learning tool by combining different nonlinear functions to transform useful information into a one dimensional feature space, in which the characteristic of each class is given a prominence. Compared with other GP-based methods which need c GP-trees [108,109,123,130] to solve a c-class ($c > 2$) pattern recognition problem, the approach proposed here needs only a single GP run to produce a single tree representative. This proves to be an extremely

promising start.

6.5 Summary

This chapter first provides an overview of feature selection/generation systems for multi-class pattern recognition task. A novel method based on GP for obtaining a single feature by single run for multi-class problem is introduced. The GP-generated features based on the proposed algorithm exhibit discrimination capability. Seven experimental data set are utilised for examining the capability and efficiency of the proposed method. Furthermore, the parameters used in experiments are provided for seven data sets. The classification results of using GP-generated features, KPCA-extracted features and GDA-extracted features with MLP, kNN and MDC are compared over seven data sets. The advantages of the proposed method for multi-class pattern recognition problems are concluded at the end.

Chapter 7

Conclusions

In this thesis, as an important task in pattern recognition problems, feature extraction is addressed under an integrated framework of evolutionary computation. This design benefits from the learning ability of the evolutionary algorithms, as well as the low computational demand of statistical analysis to the problem. It is believed that this realisation has the advantage of going through a relatively large number of generations within a reasonably short period of time. Hence good quality features can be obtained quickly under such configuration.

In Chapter 2, a literature review is conducted to cover the available feature extraction methods and classifiers. Their strength and limitations have been addressed and analysed, in order for a fair comparison with the mainly proposed method in this research. Inspired by the biological evolution process, the feature extraction task in this research work has been addressed under an integrated framework of genetic programming and statistical analysis. In order to cover different scenarios in practical applications, three genetic-programming based feature extraction systems are designed and developed for the dual-class and multi-class pattern recognition problems. First, a multi-feature system for multi-class problem is designed and tested using different data sets. Second, a

multi-feature system for dual-class problem is developed and tested. Finally, a single-feature system is constructed for the multi-class problem and tested using various data sets.

7.1 Discussion & Remarks

In the following text, issues concerning the construction and testing of the genetic-programming based approach will be addressed and conclusions will be drawn upon important system parameter settings and choice of design for different practical applications.

Principal component analysis, as one of the most straightforward feature extraction/dimensionlity reduction methods, is still a popular approach for unsupervised problems. For supervised problems, Fisher linear discriminant analysis provides an essential analysis to the statistical data distribution. Variations of linear discriminant analysis, such as the alternative Fisher linear discriminant analysis and the eigenvector-based heteroscedastic linear discriminant analysis, are equipped with improved separability.

The advantage of the linear algorithms is the fast analytic solution to the problem. However, the limitation of these methods becomes obvious in the data with nonlinearity. Kernel functions are introduced to handle the nonlinearity. After the transformation of the kernel matrix, the data are mapped into a new space, in which the linear features are capable of separating data patterns. The kernel versions of above-mentioned feature extraction methods are called KPCA and KGDA respectively.

The involvement of evolutionary computation in the feature extraction tasks starts from the use of genetic algorithm as a feature selection tool under the process of evolution. The algorithm benefits from the statistical behaviour of the

feature selection process and the searching power in a relatively large searching space. However, this is the first time the features are *selected* rather than analytically combined. Unwanted information could be ignored and only the features with discrimination information can contribute to the classification. However, this GA-based design requires the classification results as the fitness measure, hence slows down the computation speed.

The benefit of using genetic programming as a tool for feature extraction is the learning ability of the evolutionary computation. As a machine learning technique, genetic programming learns directly from the data and builds up the intelligence during the evolution process. The mathematical paradigm of genetic programming determines that it does not require the knowledge of statistical distribution of data. Hence it can be applied to many practical problems, where it is often difficult to obtain such information. The design of a fitness measure guarantees that once a certain level is reached, the quality of the feature will never drop below that level. Another motivation for using genetic programming is the significant advance of the computing power of modern computers. It is very possible to undertake a complicated computing job in a home personal computer, which can only completed by a main frame computer 10 years ago.

Theoretically, classification result is the best measure for the fitness as the ultimate goal is to conduct classification. Within a fixed period of time, a slow fitness measure provides excellent assessment, however only a limited number of generations have been reached hence the feature space may not have been fully explored and the feature may be far from optimised. It has been concluded from this research that the fitness function is best designed as a tradeoff between accuracy and efficiency.

In this regard, a Fisher-criterion based fitness measure is designed for the task of ranking the candidates during the evolution process. By testing the rate of

between-class scatter over the within-class scatter, the separation ability of each feature is quantified through primitive mathematical calculations. Compared to the standard classifier approaches, which usually require the order of N^2 or N^3 mathematical operations (N is the number of samples), the Fisher-criterion based fitness measure merely requires $2N$ sums and N multiplications.

In this research, there is a notable achievement in terms of classification of multi-pattern problems. The single feature obtained in Chapter 6 reduces the dimensionality into the minimal value without losing the discrimination information in the data. Within a one dimensional feature space, the clusters belonging to different conditions are well separated. By setting thresholds at certain levels, it does not require a sophisticated classifier to tell the class labels. Thanks to the fitness measure designed, the distribution of all classes are well presented to the classifier. Neither extremely condensed distributions nor sparse distributions are present in the data. Thus under such feature extraction scheme no computationally complex classifier is required for successful classifications. Instead simple thresholds will be sufficient for the task. It is believed that this model will be attractive for industrial applications, where the most cost-effective solutions are desired.

A series of experiments have been conducted using various data sets to evaluate the performance of the three GP-based feature extraction systems. Experimental results demonstrate the superior capability of the GP-based feature extractor for independently identifying features from the raw data. Compared to classical feature extraction methods, this approach provides better accuracy in the classification results as well as the robustness over different classifiers. Overall results covering three systems prove an extremely promising design for solving practical feature extraction problems.

7.2 Future Work

In this thesis, a series of linear feature extraction algorithms have been investigated, such as FLDA, PCA, A-FLDA and nonlinear feature extraction algorithms, such as KPCA, GDA. A linear feature extraction algorithm: Modified FLDA and three GP generation algorithms are proposed to overcome the limitation of existing algorithms. This research first utilises genetic programming together with Fisher criterion analysis for the feature extraction tasks. The feature generation algorithms are developed based on the GP structure for dual-class and multi-class classification problems. All the algorithms demonstrate the capability of feature generation and dimensionality reduction in various pattern recognition experiments. The experimental results show that the classification performance is improved using features generated by GP.

There are quite a few interesting topics raised during the research work, that might be worth further examination. One example is to utilise other statistical analysis for the design of fitness function. In addition, the tree structure of the genetic programming paradigm can have more variations. In this research, the chromosome has only one output. Designing a multi-input multi-output tree structure to represent the chromosome will be an interesting task in order to improve the feature efficiency.

In this research, we only focus on the feature generation rather than GP structure or operations themselves for improving the efficiency. It is a valuable research issue to improve the performance and reduce the computation cost by developing new genetic operations.

Another interesting area is the application to image processing. As it is well known that machine learning methods demand significant computations, in particular for image processing problems, where usually a huge number of features are

calculated for the optimisation. It will be very useful to simplify the genetic programming computation structure and improve the capability for handling large quantify of data for computation.

Bibliography

[1] L. B. Jack and A. K. Nandi, "Genetic algorithms for feature selection in machine condition monitoring with vibration signals," *IEE Proc. Vision, Image and signal Processing*, vol. 147, no. 3, pp. 205–212, 2000.

[2] J. Bellegarda, "Statistical techniques for robust ASR: Review and perspectives," 1997, pp. KN–33–KN–36.

[3] S. K. Das, "Some dimensionality reduction studies in continuous speech recognition," 1983, pp. 292–295.

[4] Y. Chow, M. Dunham, O. Kimball, M. Krasner, G. F. Kubula, J. Markhoul, P. Price, S. Roucos, and R. Schwartz, "The BBN continuous speech recognition system," 1987, pp. 89–92.

[5] W. Huang, R. P. Lippmann, and B. Gold, "A neural net approach to speech recognition," 1988, pp. 99–102.

[6] A. H. Toselli, A. Juan, and E. Vidal, "Spontaneous handwriting recognition and classification," *Pattern Recognition, International Conference on*, vol. 1, pp. 433–436, 2004.

[7] C. Bahlmann, "Directional features in online handwriting recognition," *Pattern Recognition*, vol. 39, no. 1, Jan. 2006.

[8] S. Impedovo, *Fundamentals in Handwriting Recognition.* Springer-Verlag, 1994.

[9] J. C. Simon, "The recognition of handwriting," in *ICPR*, 1998, p. K.S. Fu Lecture.

[10] L. D. Harmon, "The recognition of faces," *Scientific American*, vol. 229, pp. 71–82, 1973.

[11] D. Beymer, "Face recognition under varying pose," Massachusetts Institute of Technology - Artificial Intelligence Laboratory, A.I. Memo No. 1461, Dec. 1993.

[12] R. Brunelli and T. Poggio, "Face recognition through geometric features," in *Proceedings 2nd European Conference on Computer Vision*, Italy, May 1992, pp. 792–800.

[13] ——, "Face recognition: Features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, 1993.

[14] T. M.E., "Principal component analysis (2nd ed.). i. t. jolliffe," *Journal of the American Statistical Association*, vol. 98, pp. 1082–1083, January 2003.

[15] B. Zhang, J. Yan, N. Liu, Q. Cheng, Z. Chen, and W.-Y. Ma, "Supervised semi-definite embedding for image manifolds," in *ICME*. IEEE, 2005, pp. 592–595.

[16] J. Yang and V. Honvar, "Feature subset selection using a genetic algorithm," *IEEE Intelligent Systems*, vol. 13, pp. 44–49, 1998.

[17] F. Z. Brill, "Genetic algorithms for feature selection," Master's thesis, University of Virginia, 1990.

[18] F. Hussein, N. Kharma, and R. Ward, "Genetic algorithms for feature selection and weighting, a review and study," in *ICDAR*, 2001, pp. 1240–1244.

[19] M. L. Raymer, W. F. Punch, and A. K. Jain, "Dimensionality reduction using genetic algorithms," Aug. 16 2000.

[20] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, November 1995.

[21] B. Christopher, "Neural networks: A pattern recognition perspective," 1996.

[22] B. D. Ripley, *Pattern Recognition and Neural Networks.* Cambridge, UK: Cambridge University Press, 1996.

[23] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," in *Knowledge Discovery and Data Mining*, 1998, vol. 2, no. 4, pp. 121–167.

[24] S.-W. Lee and A. Verri, Eds., *Pattern Recognition for Support Vector Machines*, ser. Lecture Notes in Computer Science (LNCS). New York: Springer-Verlag, 2002, vol. 2388.

[25] C. C. Aggarwal, "On randomization, public information and the curse of dimensionality," in *ICDE.* IEEE, 2007, pp. 136–145.

[26] Kuo and Sloan, "Lifting the curse of dimensionality," *NOTICES: Notices of the American Mathematical Society*, vol. 52, 2005.

[27] J. Shlens, "A tutorial on principal component analysis," Tech. Rep., 2002.

[28] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[29] G. J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, ser. Wiley Series in Probability and Statistics. WileyBlackwell.

[30] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[31] B. Scholkopf, E. Smola, K. robert Muller, L. Bottou, C. Burges, H. Bulthoff, K. Gegenfurtner, and P. Haffner, "Nonlinear component analysis as a kernel eigenvalue problem."

[32] R. Rivest, T. Cormen, C. Leiserson, and C. Stein, *Introduction to algorithms.* MIT Press, Cambridge, Massachusetts, 2001.

[33] A. G. V. Kumar, A. Grama and G. Karypis, *Introduction to parallel computing.* Benjamin, Cummings, 1994.

[34] L. E. Moses, *Think and Explain with Statistics.* Reading, MA:: Addison-Wesley, 1986.

[35] E. Lehmann and J. P. Romano, *Testing Statistical Hypotheses*, 3rd ed.

[36] M. O'Mahony, *Sensory Evaluation of Food: Statistical Methods and Procedures.* CRC Press, 1986, p. 487.

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

[38] E. Fix and J. Hodges, "Discriminatory analysis: Nonparametric discrimination: Consistency prperties," in *Tech. Report 4*, USAF School of Aciation Medicine, Feb. 1951.

[39] C.-L. Huang, T.-S. Hsu, and C.-M. Liu, "The mahalanobis-taguchi system - neural network algorithm for data-mining in dynamic environments," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5475–5480, 2009.

[40] M. V. Shirvaikar and M. M. Trivedi, "A neural network filter to detect small targets in high clutter backgrounds," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 252–257, Jan. 1995.

[41] M. Nørgaard, Ed., *Neural networks for modelling and control of dynamic systems: a practitioner's handbook*, ser. Advanced textbooks in control and signal processing. pub-SV:adr: Springer-Verlag, 2000.

[42] J. Levesley, "Book review: *Radial basis functions: theory and implementations*," *Mathematics of Computation*, vol. 73, no. 247, pp. 1578–1581, Jul. 2004.

[43] T. Kohonen and T. Honkela, "Kohonen network," *Scholarpedia*, vol. 2, no. 1, p. 1568, 2007.

[44] Wong, "Stochastic neural networks," *ALGRTHMICA: Algorithmica*, vol. 6, 1991.

[45] F. Azam, "Biologically inspired modular neural networks," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2000.

[46] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review,Cornell Aeronautical Laboratory*, vol. 65, no. 6, pp. 386–408, 1958.

[47] A. K. Jain and M. D. Ramaswami, *Classifier design with parzen window*, E. S. Gelsema and L. N. Kanal, Eds.

[48] I. Kushchu, "Genetic programming and evolutionary generalization," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 431–442, Oct. 2002.

[49] H. de Garis, "Introduction to evolutionary computing," *Evolutionary Computation*, vol. 12, no. 2, pp. 269–271, 2004.

[50] D. B. Fogel, "An introduction to evolutionary computation," in *Evolutionary Computation: the fossil record*, D. B. Fogel, Ed. Piscataway, NJ: IEEE Press, 1998, pp. 1–2.

[51] V. Nissen and J. Biethahn, "An introduction to evolutionary algorithms," in *Evolutionary algorithms in management applications*, J. Biethahn and V. Nissen, Eds. Berlin: Springer-Verlag, 1995, pp. 3–43.

[52] T. Bäck, "Introduction (evolutionary algorithms and their standard instances)," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Bristol, New York: Institute of Physics Publishing and Oxford University Press, 1997, pp. B1.1:1–4.

[53] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications.* dpunkt, Heidelberg and Morgan Kaufmann, San Francisco, 1998.

[54] J. R. Koza, "Introduction to genetic programming," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. Cambridge, MA, USA: MIT Press, 1994, ch. 2, pp. 21–42.

[55] ——, "Genetic programming: Automatic programming of computers," *EvoNews*, vol. 1, no. 3, pp. 4–7, Mar. 1997.

[56] ——, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge: MIT Press, 1992.

[57] L. M. Howard and D. J. D'Angelo, "The ga-p: a genetic algorithm and genetic programming hybrid," *IEEE Intelligent Systems and Their Applications*, vol. 10, pp. 11–15, June 1995.

[58] A. D. Parkins and A. K. Nandi, "Genetic programming techniques for hand written digit recognition," *Signal Processing*, vol. 84, no. 12, pp. 2345–2365, 2004.

[59] P. Day and A. K. Nandi, "Robust text-independent speaker verification using genetic programming," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 1, pp. 285–295, 2007.

[60] L. Zhang and A. K. Nandi, "Neutral offspring controlling operators in genetic programming," *Pattern Recognition*, vol. 40, no. 6, pp. 2696–2705, 2007.

[61] T.Bach, *Evolutionary Algorithms in Theory and Practice.* New York: Oxford University Press, 1996.

[62] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* Piscataway: IEEE Press, 1995.

[63] G. A. Rovithakis, M. Maniadakis, and M. Zervakis, "A hybrid neural network/genetic algorithm approach to optimizing feature extraction for signal classification," *IEEE Trans. Syst., Man, Cybern. Part.B*, vol. 34, pp. 695–703, Feb. 2004.

[64] J. Sherrah, "Automatic feature extraction using genetic programming," in *Late Breaking Papers at the 1997 Genetic Programming Conference*, J. R.

Koza, Ed. Stanford University, CA, USA: Stanford Bookstore, 13–16 Jul. 1997, p. 298.

[65] W. A. Tackett, "Mining the genetic program," *IEEE Intelligent Systems and Their Applications*, vol. 10, pp. 28–38, June 1995.

[66] W. Banzhaf, J. R. Koza, C. Ryan, L. Spector, and C. Jacob, "Genetic programming," *IEEE Intelligent Systems and Their Applications*, vol. 15, pp. 74–84, May-June 2000.

[67] I. Benyahia and J. Potvin, "Decision support for vehicle dispatching using genetic programming," *IEEE Trans. Syst., Man, Cybern. Part.A*, vol. 28, no. 3, pp. 306–314, May 1998.

[68] L. Zhang, L. B. Jack, and A. K. Nandi, "Fault detection using genetic progamming," *Mechanical Systems Signal Processing*, vol. 19, pp. 271–289, 2005.

[69] M. C. J. Bot, "Feature extraction for the k-nearest neighbour classifier with genetic programming," in *Graduate Student Workshop*, C. Ryan, Ed., San Francisco, California, USA, 7 Jul. 2001, pp. 397–400.

[70] Y. Zhang and P. I. Rockett, "A generic optimal feature extraction method using multiobjective genetic programming," Department of Electronic and Electrical Engineering, University of Sheffield, UK, Tech. Rep., 2006.

[71] Y. Zhang and P. Rockett, "Feature extraction using multi-objective genetic programming," in *Multi-Objective Machine Learning*, ser. Studies in Computational Intelligence, Y. Jin, Ed. Springer, 2006, vol. 16, pp. 75–99.

[72] J. Holland, *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[73] K. DeJong, *An Analysis of the Behaviour of of a Class of Genetic Adaptive Systems.* Ann Arbor: Ph.D Thesis, 1975.

[74] M. L. D. Wong and A. K. Nandi, "Automatic digital modulation recognition using artificial neural network and genetic algorithm," *Signal Processing*, vol. 84, pp. 351–365, 2004.

[75] K. S. Tang, K. F. Man, S. Kwong, and Q. he, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22–37, 1996.

[76] D. S. Whitely, "A genetic algorithms tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, 1994.

[77] G. Syswerda, "A study of reproduction in generational and steady state genetic algorithms," *Foundations of Genetic Algorithms*, 1991.

[78] S. Legg, M. Hutter, and A. Kumar, "Tournament versus fitness uniform selection," in *Congress on Evolutionary Computation*, vol. 2, June, 2004, pp. 2144 – 2151.

[79] L. B. Jack, *Applications of Artificial Intelligence in Machine Condition Monitoring.* Ph.D Thesis, 2000.

[80] C. Grosan and A. Abraham, "Stock market modeling using genetic programming ensembles," in *Genetic Systems Programming*, ser. Studies in Computational Intelligence, N. Nedjah, L. de Macedo Mourelle, and A. Abraham, Eds. Springer, 2006, vol. 13.

[81] T. Sripramong and C. Toumazou, "The invention of CMOS amplifiers using genetic programming and current-flow analysis," *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 11, pp. 1237–1252, Nov. 2002.

[82] T. Ehlis, "Application of genetic programming to the snake game," *Gamedev.Net*, no. 175, 2000.

[83] P. Day and A. K. Nandi, "Binary string fitness characterisation and comparative partner selection in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 724–735, 2008.

[84] H. Guo, Q. Zhang, and A. K. Nandi, "Feature extraction and dimensionality reduction by genetic programming based on the fisher criterion," *Expert Systems*, vol. 25, no. 5, pp. 444–459, 2008.

[85] T. Mu, A. K. Nandi, and R. M. Rangayyan, "Classification of breast masses via nonlinear transformation of features based on a kernel matrix," *Medical and Biological Engineering and Computing*, vol. 45, no. 8, pp. 769–780, 2007.

[86] L. Zhang and A. K. Nandi, "Neutral offspring controlling operators in genetic programming," *Pattern Recognition*, vol. 40, no. 6, pp. 2696–2705, 2007.

[87] ——, "Fault classification using genetic programming," *Mechanical Systems and Signal Processing*, vol. 21, no. 3, pp. 1273–1284, 2007.

[88] R. J. Nandi, A. K. Nandi, R. M. Rangayyan, and D. Scutt, "Classification of breast masses in mammograms using genetic programming and feature selection," *Medical and Biological Engineering and Computing*, vol. 44, no. 8, pp. 683–694, 2006.

[89] H. Guo and A. K. Nandi, "Breast cancer diagnosis using genetic programming generated feature," *Pattern Recognition*, vol. 39, no. 5, pp. 980–987, 2006.

[90] J. R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.

[91] ——, "Improved use of continuous attributes in c4.5," *Journal of Artificial Intelligence Research*, vol. 4, pp. 77–90, 1996.

[92] W. B. Langdon, "The evolution of size in variable length representations," in *In 1998 IEEE International Conference on Evolutionary Computation*, 1998, pp. 633–638.

[93] W. A. Tackett, *Recombination, Selection, and the Genetic Construction of Computer Programs.* California: Ph.D Thesis, 1994.

[94] T. Soule and J. A. Foster, "Code size and depth flows in genetic programming," in *In Genetic Programming 1997: Proceedings of the Second Annual Conference*, 1997, pp. 313–320.

[95] W. Banzhaf and W. B. Langdon, "Some consideration on the reason for bolat," *Genetic Programming and Evolvable Machines*, vol. 3, pp. 81–91, 2002.

[96] C.-L. Lin, "Improving the generalization capability of the RBF neural networks via the use of linear regression techniques," Jul. 27 2001.

[97] T. Evgeniou and M. Pontil, "A note on the generalization performance of kernel classifiers with margin," MIT Artificial Intelligence Laboratory, Tech. Rep. CBCL-184, May 1 2000.

[98] T. Joachims, "Estimating the generalization performance of a SVM efficiently," Jan. 12 1999.

[99] I. Kuscu, "Promoting generalization of learned behaviours in genetic programming," in *Fifth International Conference on Parallel Problem Solving from Nature*, ser. LNCS, A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, Eds., vol. 1498. Amsterdam: Springer-Verlag, 27-30 Sep. 1998, pp. 491–500.

[100] W. A. Tackett and A. Carmi, "The donut problem: Scalability, generalization and breeding policies in genetic programming," in *Advances in Genetic Programming*, ser. Complex Adaptive Systems, K. E. Kinnear, Ed. Cambridge: MIT Press, 1994, pp. 143–176.

[101] M. J. Cavaretta and K. Chellapilla, "Data mining using genetic programming: The implications of parsimony on generalization error," in *1999 Congress on Evolutionary Computation.* Piscataway, NJ: IEEE Service Center, 1999, pp. 1330–1337.

[102] T. Ito, H. Iba, and M. Kimura, "Robustness of robust programs generated by genetic programming," in *in Proc. 1st Annu. Conference Genetic Programming*, Stanford, CA, July 1996, pp. 321–326.

[103] I. Kuscu, "Evolution of learning rules for hard learning problems," in *In Proc. 5th Annu. Conference Evolutionary Progamming.* Cambridge, MA: MIT Press, Feb.-Mar. 1996, pp. 91–99.

[104] ——, "Evolving a generalized behavior: Artificial ant problem revisited," in *in Proc. 7th Annu. Conf. Evolutionary Programming*, Berlin, Germany, Berlin, pp. 799–808.

[105] T. F. Bersano-Begey and J. M. Daida, "A discussion on generality and robustness and a framework for fitness set construction in genetic programming to promote robstness," in *Late Breaking Paper 1997 Genetic Programming Conference*, Stanford CA, July 1997, pp. 11–18.

[106] D. Hooper and N. S. Flann, "Improving the accuracy and robustness of genetic programming through expression simplification," in *In Proc. of 1st Annu. Conference Genetic Programming*, Stanford, CA, July 1996, p. 429.

[107] F. D. Francone, P. Nordin, and W. Banzhaf, "Benchmarking the generalization capabilities of a compiling genetic programming system using sparse data sets," in *In Proc. of 1st Annu. Conference Genetic Programming*, Stanford, CA, July 1996, pp. 72–80.

[108] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Arawal, "Application of genetic programming for multicategory pattern classification," *IEEE Trans. on Evolutionary Computation*, vol. 4, no. 3, pp. 242–258, 2000.

[109] M. Brameier and W. Banzhaf, "A comparison of linear genetic programming and neural networks in medical data mining," *IEEE Trans. on Evolutionary Computation*, vol. 5, no. 1, pp. 17–26, 2001.

[110] M. Kotani, S. Ozawa, M. Nasak, and K.Akazawa, "Emergence of feature extraction function using genetic programming," in *Knowledge-Based Intelligent Information Engineering Systems, Third International Conference*, 1997, pp. 149–152.

[111] K. Ragulskis and A. Uurkauskas, *Vibration of Bearings*. Bristol PA: Hemisphere, 1989.

[112] G. Lipovszky, K. Solyomvari, and G. Varga, *Vibration Testing of Machines and Their Maintenance.* Amsterdam, The Netherlands: Elsevier, 1990.

[113] T. A. Harris, *Rolling Bearing Analysis.* New York: Wiley, 1991.

[114] S. Korablev, V. Shapin, and Y. Filatov, *Vibration Diagnostics in Precision Instruments.* Bristol PA: Hemisphere, 1989.

[115] P. Chen, T. Toyota, and Z. He, "Automated function generation of symptom parameters and application to fault diagnosis of machinery under variable operating conditions," *IEEE Trans. Syst., Man, Cybern. Part.A*, vol. 31, no. 6, pp. 775–781, Nov. 2001.

[116] D. E. Butler, "The shock pulse method for the detection of damaged rolling bearing," *NDT Int.*, vol. 6, pp. 92–95.

[117] S. Petridis and S. J. Perantonis, "On the relation between discriminant analysis and mutual information for supervised linear feature extraction," *Pattern Recognition*, vol. 37, pp. 857–874, 2004.

[118] S. Chen and X. Yang, "Alternative linear discriminant classifier," *Pattern Recognition*, vol. 37, pp. 1545–1547, 2004.

[119] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998.

[120] W. Street, W. Wolberg, and O. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," in *International Symposium on Electronic Imaging: Science and Technology*, San Jose, CA, 1993, no. 1905, pp. 861–870.

[121] B. C. Chien, J. Y. Lin, and T. P. Hong, "Learning discriminant functions with fuzzy attributes for classification using genetic programming," *Expert Syst. with Applications*, vol. 23, pp. 31–37, 2002.

[122] B. C. Chien, J. Y. Lin, and W. P. Yang, "Learning effective classifiers with Z-value measure based on genetic programming," *Pattern Recognition*, vol. 37, pp. 1957–1972, 2004.

[123] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Trans. on Evolutionary Computation*, vol. 8, pp. 183–196, 2004.

[124] ——, "Genetic programming for simultaneous feature selection and classifier design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, pp. 106–117, Feb. 2006.

[125] Z.-H. Zhou, Y. Jiang, and S. Chen, "Extracting symbolic rules from trained neural network ensembles," *AI Commun*, vol. 16, p. 3, 2003.

[126] B. Ricks and D. Ventura, "Training a quantum neural network," in *NIPS*, 2003.

[127] Z. Hong and Y. J. Yang, "Optimal discriminant plane for a small number of samples and design method of classifier on the plane," *Pattern Recognition*, vol. 24, pp. 317–324, 1991.

[128] L. B. Jack, A. K. Nandi, and A. C. McCormick, "Diagnosis of rolling element bearing faults using radial basis function networks," *Applied Signal Processing*, vol. 6, no. 1, pp. 25–32, 1999.

[129] M. Deshpande and G. Karypis, "Using conjunction of attribute values for

classification," in *In Proceedings of the eleventh CIKM.* ACM Press, 2002, pp. 356–364.

[130] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Trans. Syst., Man and Cybern. Part B*, vol. 35, no. 1, pp. 89–99, 2005.