VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematics, Statistics and Computer Science
Te Kura Tatau

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@mcs.vuw.ac.nz

Particle Swarm Optimisation for Image Classification

Hamish Evans

Supervisor: Mengjie Zhang

Submitted in partial fulfilment of the requirements for
Bachelor of Information Technology.

Abstract

With the processing power available to modern computers, reliable multi-class image classification is now a realistic task. However the need for efficient and effective classifiers and search techniques is still an area that needs to be developed to enable this. Particle Swarm Optimisation (PSO) has had some success in image classification, however all existing PSO image classification techniques have been used in conjunction with an existing object classification technique.

This project describes two new stand alone approaches to object and image classification. Instead of using PSO to evolve a set of good parameter values only for another, already existing, object classification technique, the new approach can be used as a stand alone method for classification. The first new PSO method treats all different features equally important and finds an optimal partition matrix to separate a data set into distinct class groups. The second new PSO method considers the relative importance of each feature and the noise factor, and evolves a weight matrix to mitigate the effects of noisy partitions and feature dimensions. The two methods are examined and compared with a popular method using PSO combined with the nearest centroid and another evolutionary computing method, genetic programming, on three image data sets of increasing difficulty. The results suggest that the new weighted PSO method outperform these existing methods on these object classification problems.

PSO has also been shown to convergence quickly. This means solutions are found efficiently but the solution found is often a local optimum solution. This project describes a new variant of PSO which integrates local search technique, Simulated Annealing, into the particle dynamics. The new technique is compared against the regular PSO versions of WPPSO and PSONCC and delivers some unexpected results.

Acknowledgements

I would like to acknowledge the following:

- My superviser Dr. Mengjie Zhang, for pushing me to work consistently throughout the year. The lessons you have taught me in work ethic, research skills, writing skills and evolutionary computing have been extremely valuable.
- My friend Mathew Campbell-Adams, for always being available to bounce ideas off and for proof reading my writing on multiple occasions.

Contents

Figures

Tables

Chapter 1

Introduction

Today's society relies on computer systems heavily, and everyday computer systems are placed with the trust to make important decisions. As a result of this there is a large demand for efficient and effective decision making algorithms. It was not so long ago that computer vision was an impossible task, however with the exponential increase in computer processing power, achieving computer vision is now a realistic possibility. The challenge is to develop and improve computer vision techniques so that they can equal or even supercede the ability of human sight. Some computer vision tasks may include: image processing, image segmentation, and image classification.

Evolutionary Computing (EC) is a subsection of Artificial Intelligence (AI), or more specifically a child branch of Computational Intelligence. EC obtained its name from its ties to the biological evolution process. All EC techniques maintain a population, where each individual in the population is a potential solution to the problem domain. The population is refined by some user-defined fitness function under an iterative progression, until a point of convergence is reached.

Evolutionary Computer Vision is the science which involves using EC techniques to build artificial systems that perform some specific task involving sight. There are many EC techniques which can be applied to Computer Vision, however this project concerns the application and advancement of two biologically inspired EC techniques, namely Particle Swarm Optimisation (PSO) and Genetic Programming (GP), for Image Classification tasks. The role of PSO and GP in Image Classification is to find the relationship between features extracted from images and the associated class label of images. The result is a classifier which can take as input a set of features extracted from an unseen image and predict the image's class label.

Being a relatively new technique in Evolutionary Computing, PSO is an undeveloped method for classification. Most of the existing PSO classification techniques have used PSO in combination with an established machine learning and classification algorithm such as with the Nearest Centroid method or Artificial Neural Networks. While these methods have achieved good results in a number of classification tasks, they have a number of limitations. For example, the hybrid methods will always be bound to the same limitations as the existing machine learning techniques that they are fundamentally based on. In addition, the hybrid methods almost always require some extra overhead to process the interaction between PSO and the classification algorithms.

PSO has also shown to have an early convergence problem. Due to the laws which dictate how particles move, particles will move quickly toward the best potential solution in the population. This means that the final solution found by PSO is often a local optimum solution rather than a global optimum solution. With limited computational resources available to PSO it is unreasonable to search every possible solution in a large or infinite solution space, so the challenge is to devise a variant of PSO that can maintain a balance in the population between optimal particles and suboptimal particles, as it may be that the suboptimal particles lead to the global optimum solution.

1.1 Goals

This first and primary goal of this project is to gain a greater insight into the ability of PSO as a stand alone classifier. This goal will be achieved by developing two new independent PSO approachs to object classification called Feature Partitioning Particle Swarm Optimisation (FPPSO) and Weighted Feature Partitioning Particle Swarm Optimisation (WFPPSO), which does not rely on any existing machine learning technique. The approach will be examined and compared with a common hybrid PSO-Nearest Centroid method and a GP method on a series of object classification problems of increasing difficulty. We will investigate whether the new approach can do a good enough job and whether the new approach outperforms these existing methods.

The secondary goal of this project is to analyse and address the early convergence problem in the PSO method. This goal will be achieved by creating a variant of PSO call Particle Swarm Optimisation Simulated Annealing (PSOSA), which as the name suggests integrates the local search technique, Simulated Annealing, into the movement of particles through the solution space. Simulated Annealing will be used to introduce more diversity into the population to help prevent early convergence.

1.1.1 Research Questions

1. Can the Particle Swarm Optimisation Nearest Centroid (PSONCC) method outperform the standard GP method on a sequence of multiclass image classification problems?
2. Can the new stand alone PSO techniques for classification, FPPSO and WFPPSO, outperform the established classification techniques PSONCC and GP on a sequence of multiclass image classification problems?
3. Can the PSO variant, PSOSA, mitigate the early convergence problem by integrating Simulated Annealing?

1.2 Major Contributions

This project has made the following major contributions:

1. This project shows how to use particle swarm optimisation (PSO) to construct classifiers for multi-class image classification problems. Two new PSO classification techniques which do not rely on any existing classification techniques have been developed. The performance of the new techniques is evaluated against existing classification techniques such as the PSO nearest centroid hybrid method and genetic pro-

gramming. One of the new methods, WFPPSO, shows to outperform the existing techniques in all but one experiment.

2. A paper titled "Particle Swarm Optimisation for Object Classification" has been submitted for acceptance at the 23rd International Conference on Image and Vision Computing New Zealand.

1.3 Organisation

This chapter has introduced the concepts of vision and classification tasks in evolutionary computing. It has also outlined the current limitations of PSO as a technique and a classifier.

Chapter 2 gives a more in depth background of Machine Learning and Evolutionary Computing. In sections 2.4 and existing object classification methods for PSO and GP are described respectively. Chapter 3 describes the image sets which are used to test all of the algorithms in this project.

Chapter 4 describes the baseline image classification techniques, PSONCC and GP. The specific parameters used in the experiments for each technique are presented and then the results for a series of image classification tasks are presented and analysed.

Chapter 5 gives an in depth description of the new stand alone PSO classification techniques, WFPPSO and FPPSO. Specifically this includes a high level description backed with an example, there is also a description of how the classifier is encoded into a particle.

Chapter 6 describes an experimental technique called GDPPSO, which extends the FPPSO framework to take a full probabilistic approach to object classification.

Chapter 7 describes Particle Swarm Optimisation Simulated Annealing which is a PSO variant that attempts to mitigate the effects of early convergence in PSO. The results are presented and compared against the non simulated annealing versions of the technique.

Chapter 8 outlines the overall conclusions and all the specific conclusions. Possible future directions and ideas are also dicussed.

Chapter 2

Literature Review

2.1 Overview Object Classification and Fundamental Elements

2.1.1 Features

A feature can be described as a discrete or continuous value that pertains to a type of entity. Features are the fundamental building block of many representations in AI search problems. The following sections describe the various categories of features and also the feature vector which is the standard format for representing an object in an object classification problem.

Categories of Features

Features can for the most part be grouped into a hierarchy of three groups: low level, mid-range and high level features. Low level features are values which do not abstract away from the object/image at all. The raw pixel values of an image are an example of low level features because they are a direct representation of the object, in this case an image. The major drawback of using low level features is the large amount of computational power to process them in an efficient way. For this reason low level features are only seldom used by object classification techniques [1].

The mid-range category of features is a somewhat debatable category. The category contains features which are domain independent but abstract away from the object/image in some way. Pixel statistics such as the mean or standard deviation of an image would be mid-range features. They are mid-range features because they are still domain independent (they can be applied to any image set) but do not directly represent the image (they abstract away from the raw pixel statistics). As we will later see, mid-range features are chosen to be used in the experiments contained within this project.

High level features can be described as domain dependent features, they describe something very unique about each object in the data set being classified. If an image set of different faces was being classified, then examples of high level features would be the distance between their eyes, length of their nose and the curvature of their mouth. They are high level features because you cannot obtain these types of features from all kinds of image sets. High level features are in most cases able to obtain the best classification accuracy, however they are used in a goal oriented situation rather than a technical oriented situation. For this reason they are not used within this paper.

Feature Vector

The feature vector is a vector of discrete or continuous values that describes a particular entity in a given search problem. It defines a standard format which can be relied upon by the search technique. In an object classification problem the general format of a feature vector is a series of features pertaining to the object followed by one or more output labels, this is defined in equation 2.1 where n is the number of features.

$$FeatureVector = (f_0, f_1, .., f_n, classlabel) \tag{2.1}$$

2.1.2 Data Sets

Training Set

The training set consists of all feature vectors that are used within the training/learning process of the classifier.

Testing Set

The testing set consists of all feature vectors that are not used within the training/learning process of the classifier. They are used to test the ability of the classifier to correctly classify unseen instances. The motivation for this is that practical classifiers are often required to classify instances that are not currently known.

Data Set Normalisation

The data set is normalised such that every value in each feature vector is in the range [0, 1]. This normalisation process ensures that each feature has an equal influence in the classification. For example, let the classifier be a mathematical function with one dimension in the data set in the range [1000, 10000], and another in the range [0, 100]. It would be likely that the dimension in the range [1000, 10000] would have a greater influence in the classification than the dimension in the range [0, 100]. This characteristic is, in most cases, undesirable. The data set is normalised according to equation 2.2.

$$x_{ij} = \frac{x_{ij} - x_j^{min}}{x_j^{range}} \tag{2.2}$$

Where x_j^{min} is the minimum value in the dimension j and $x_j^{range} = x_j^{max} - x_j^{min}$.

2.2 Overview of Machine Learning

Machine Learning is a subsection of AI and a large one at that. It encompasses all areas of AI which involve adapting to new circumstances and extrapolating patterns. There are a number of learning methods which machine learning techniques use, such as Supervised Learning, Unsupervised Learning and Reinforcement Learning. The following sections will briefly describe each of these learning methods.

2.2.1 Supervised Learning

Supervised Learning is a method which, in conjunction with the training set, enables the adaptation of a candidate solution. The candidate solution is applied to the feature vector of each instance in the training set to produce a predicted output value. The predicted output value of each instance in the training set is then compared to the desired output value of that instance. By this, the goodness of the candidate solution can be obtained, for example the total error rate.

If the candidate function does not meet a satisfactory requirement it can be modified and re-evaluated. On the contrary, if it does, the evaluation process can be stopped. It is this iterative modification and re-evaluation that is the foundation of the supervised learning process. To test the candidate solution, that has reached a point of termination in the learning process, it is a applied to the testing set.

2.2.2 Unsupervised Learning

Unsupervised Learning is similar to Supervised Learning in that it uses a feature vector but it does not have the supervision of the output class label. Unsupervised learning is used to locate patterns in the data set feature vectors. The patterns can then be used in some way to separate the data set instances into groups and ultimately classes. The groups found via unsupervised learning do not always directly correlate to the regular groups one might expect.

2.2.3 Reinforcement Learning

Reinforcement Learning is a method somewhere in between that of Supervised Learning and Unsupervised Learning. It is a method of learning that gives the system some insight into how correct a candidate solution is, much like a hint. The degree to which a candidate solution is correct is represented by some discrete scale, rather than a binary correct or incorrect feedback.

2.3 Overview of Evolutionary Computing

Evolution is a theory that was first described Charles Darwin in the late 1880s [2]. The theory describes the progression of life through the means of natural selection. In other words all species, either in or out of existence, have come from common ancestors and have succeeded or failed by natural selection.

In the 1960s the concepts outlined by Darwin were adapted into computer systems to enable the programming of computers by means of natural selection [3]. The first search technique created was Genetic Algorithms [4], however the success of this technique later spawned many others such as: Genetic Programming [3] [5], Evolutionary Strategy and Learning Classifier Systems [6] [7].

More recently a new branch of techniques, that are based on social-psychological insight, have been created under the category Swarm Intelligence, namely: Particle Swarm Optimisation and Ant Colony Optimization.

This section gives an in depth overview of the techniques pertaining to this project, which are Particle Swarm Optimisation, Genetic Algorithms and Genetic Programming.

2.3.1 Particle Swarm Optimisation

Introduced by Jim Kennedy and Russell C. Eberhart in in 1995 [8] [9], PSO is a stochastic, population-based evolutionary algorithm for problem solving. It is a kind of swarm intelligence that is based on social-psychological principals and provides insight into social behavior.

The population in PSO is called the swarm, where each individual in the population is a particle. A particle is a position (a vector) in a d-dimensional solution space. A particle modifies its position according to its velocity (a vector), which is stored within the particle. A particle also maintains a reference to the best location it has come across in the solution space. This reference is called the *pbest*, short for previous best.

The particles in the swarm are interconnected by a topology, called a neighbourhood. Each particle can share information with its neighbours, which are determined by the neighbourhood. The best *pbest* out of all of a particle's neighbours is known as the *gbest*, short for global best. There are two commonly used neighbourhoods, the global neighbourhood and the ring neighbourhood, they are depicted in figures 2.1 and 2.2 respectively.

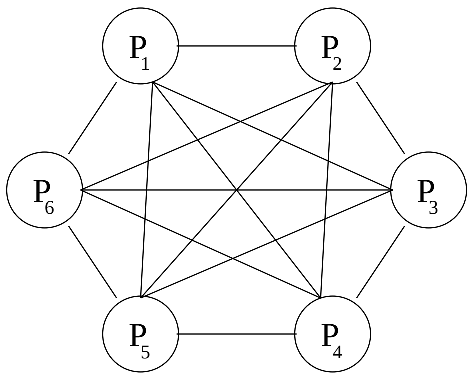

Figure 2.1: Global Neighbourhood

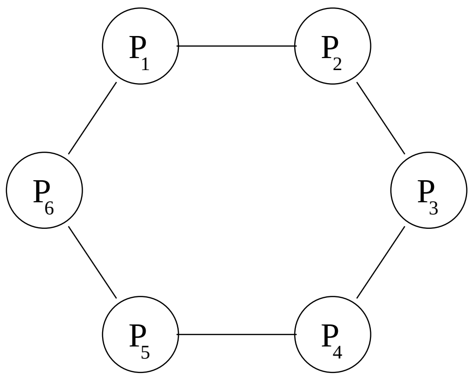

Figure 2.2: Ring Neighbourhood

The global neighbourhood supports fast information sharing between particles and is used in cases where efficiency is valued over effectiveness. A ring neighbourhood on the other hand slows the propagation of information through the swarm. This will increase the time taken for the swarm to converge, however it helps to prevent the swarm from converging on a local optimum solution by allowing the swarm to search more diverse areas of the solution space.

The optimisation process of PSO occurs in an iterative fashion. Within each iteration the velocity of each particle in the swarm is updated. Equation 2.3 dictates how a particle's velocity is updated, where r_1 and r_2 are two independently generated random numbers in the range [0,1] and χ is a constriction factor used to prevent velocity values from growing out of control. The particle position can then be updated according to formula 2.4 where χ is a constriction factor, X is the particle's current position in the solution space and r_1 and r_2 are two randomly generated floating point values in the range $[0, 1]$.

$$PV_{id} = \chi[PV_{id} + \phi_1 r_1(pbest_{id} - X_{id}) + \phi_2 r_2(lbest^n_{id} - X_{id})] \tag{2.3}$$

$$X_{id} = X_{id} + PV_{id} \tag{2.4}$$

2.3.2 Genetic Algorithm

Introduced by John Holland in 1954, the Genetic Algorithm is a highly parallel process that evolves fixed length character strings using evolutionary theory first described by Charles Darwin [2] in the 1880s. The Genetic Algorithm maintains a population of solutions which are selectively modified over an iterative process by genetic operators. The solution representation and genetic operators will be described in the next two sections.

Solution Representation

Solutions in Genetic Algorithms [4] are usually represented as a sequence of fixed length strings. Each value in the program string will represent some property associated with a solution to that problem domain. A simple example would be if you were the owner of a restaurant and wanted to implement a strategy to maximise your profits. Assume there were three binary decisions such as:

- Price - Should the price of steak be 10 or 15 dollars.
- Drink - Should wine or coke be served with the steak.
- Speed of Service - Should the customers have to wait for their steak or should it be served as soon as possible.

The problem could be encoded in a GA as a bit string of three binary values where the first bit represents the price property, the second represents the drink property and the last represents the speed of service property.

Genetic Operators

The evolution process in the Genetic Algorithm occurs over many iterations called generations. Each generation the current population of individuals (fixed length strings) are modified by two main genetic operators: crossover and mutation.

Crossover works by taking two individuals in the population. It then takes a substring at a random point from each of the individuals. The two substrings are swapped which produces two new offspring. This is analogous to biological sexual reproduction.

Mutation works by taking a singular individual in the population and randomly mutating one of the values in the bit string.

Evolution Process

A user defined fitness function is used by the Genetic Algorithm to guide the population to a point of convergence that is desirable. The fitness function is applied to each individual in the population at the end of every iteration. When the fitness function is applied to an individual it gives a *fitness* value which dictates the goodness of the individual. If the fitness of every individual in the population is known, the GA can select individuals for mating that will produce better offspring in the next generations.

2.3.3 Genetic Programming

GP is a population based Evolutionary Computing technique used to solve all kinds of problems such as symbolic regression and classification. The idea of Genetic Programming stemmed from the original Genetic Algorithm with the fundamental difference that individuals would be represented as dynamic tree structure rather than a fixed length string.

Solution Representation

Each individual in the GP population is represented as a tree structure consisting of both terminals (leaves) and functions. Functions generally fall into one of a few categories: standard arithmetic operations, standard programming operations, standard mathematical functions, logical function, or domain-specific functions. Each of these functions may take as arguments terminals or functions, which recursively build up a tree structure. Terminals which act as the leaf nodes of the tree are usually one of boolean, integer, real, vector or symbolic types [3]. Figure 2.3 depicts a simple tree structure of the addition function with a feature terminal and a floating point terminal.

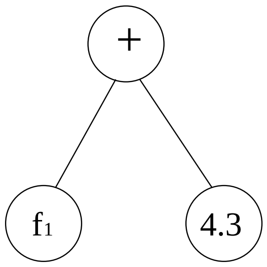

Figure 2.3: GP Tree Structured Program

Genetic Operators

The initial population of GP programs is generated randomly, however in the subsequent cases GP programs are generated using genetic operators. The two genetic operators used in GP are called crossover and mutation. Crossover takes two existing GP programs, then takes two different random subtrees from each of them. It then proceeds to swap the two subtrees, producing two different child GP programs. Mutation on the other hand only requires one parent and constructs offspring by randomly changing a certain subtree of the parents tree structure. The portion may be a terminal, a function or a combination of the two. Figures 2.4 and 2.5 depict the crossover and mutation genetic operators.

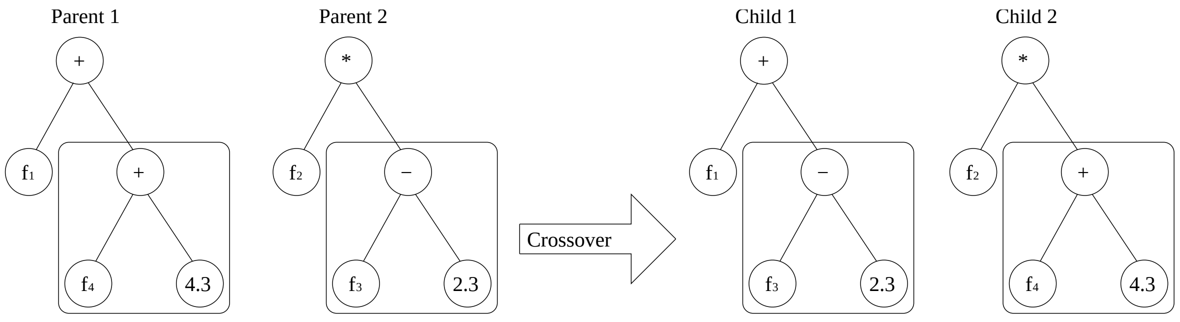

Figure 2.4: Crossover Genetic Operator

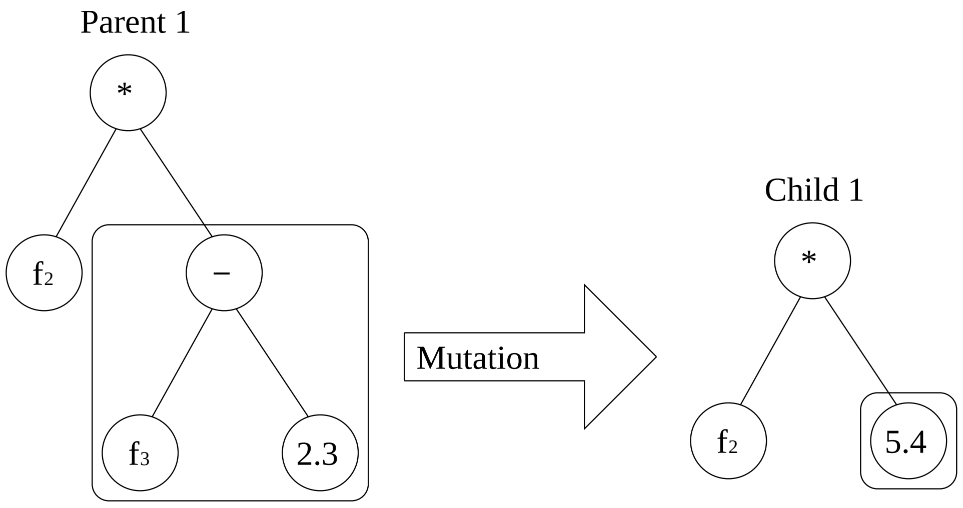

Figure 2.5: Mutation Genetic Operator

Evolution Process

The learning process in GP, known as the evolution process, is iterative. Each individual in the GP population is evaluated every evolution by some user defined fitness function which returns the goodness of an individual called the fitness.

Once the fitness of each individual in the population has been evaluated, a mating pool must be selected from the population, which can be used to produce the next generation of GP programs. The mating pool selection process can be done via a number of methods, such as tournament selection[10] or roulette wheel selection [3]. Roulette wheel selection is the more common selection process which selects individuals of the mating pool proportionate to their fitness. Once the mating pool has been selected, genetic operators can be used to generate the next population. This cycle is repeated until some termination criteria is met.

2.4 Overview of PSO Classification Methods

Particle Swarm Optimisation has been used for classification with a decent amount of success. However all existing PSO classification methods have been used in combination with an existing classification method, such as the Nearest Centroid method or with Neural Networks. When PSO is used in combination with an existing classification method it forms what is termed a *hybrid technique*. The following sections will describe the Nearest Centroid method and Artificial Neural Networks, and how PSO is used to assist in classification.

2.4.1 PSO-Nearest Centroid Hybrid Technique

The Nearest Centroid method has been around for some time and relates closely to other proximity search techniques such as the nearest neighbor method. PSO can be combined with the nearest centroid method to achieve better results. The following sections describe the nearest centroid classifier and how PSO can be combined with the Nearest Centroid Classifier to form a hybrid technique.

Nearest Centroid Classifier

The standard Nearest Centroid Classifier (NCC) uses a set of centroids to classify a feature vector. There is a centroid for every class, where the centroid for a particular class is the mid position of every feature vector of that class, in the training set. This means that each centroid is a vector of size d, where d is the number of dimensions in a feature vector.

To classify a feature vector using the class centroids, the distance between each centroid and the feature vector is calculated. There is a variety of distance measures that can be used, however the most common distance measure is the Euclidean distance, which is calculated using equation 2.5.

$$Euclidean(x,y) = \sqrt{\sum_{i=0}^{d}(x_i - y_i)^2} \tag{2.5}$$

The class of the centroid, which has the minimum distance from the feature vector, is the classification of that feature vector.

PSO and NCC Hybridisation

PSO can be combined with NCC (PSO-NCC) to optimise the class centroids, however using the term centroid is no longer accurate in this technique since, in most cases, they do not represent the middle point of the class instances. A more accurate term would be *prototypes*.

To use PSO to evolve the class prototypes, each particle must be encoded as a $c \times d$ matrix, where c is the number of distinct classes in the training set and d is the number dimensions in each feature vector, this is shown by equation 2.6.

$$particle_i = \begin{pmatrix} cv_{(1,1)} & .. & cv_{(1,d)} \\ .. & .. & .. \\ cv_{(c,1)} & .. & cv_{(c,d)} \end{pmatrix} \tag{2.6}$$

As each particle changes its position in the solution space by equations 2.3 and 2.4, the particle is effectively changing the positions of the class prototypes. As a result of this the class prototype positions are often irregular. However irregular positions often result in better classification accuracy than using the regular centroids. An example of two class prototypes that have been evolved in a two dimensional solution space using PSO-NCC are depicted in figure 2.6.

The two class prototypes, at the bottom of the graph, are the points which each instance is measured from. The closest class prototype becomes the classification for a given instance, the boxes of each class prototype encapsulate the instances which will be classified as the class of that prototype.

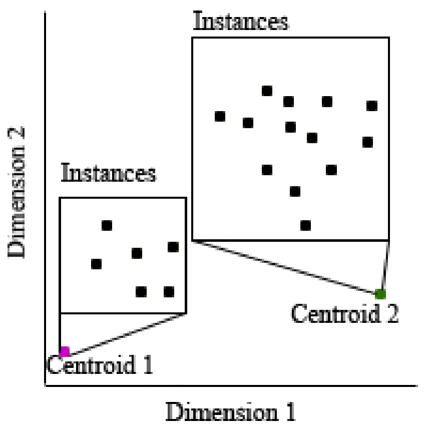

Figure 2.6: Class Prototypes Evolved by PSO-NCC

2.4.2 PSO-Neural Networks Hybrid Technique

Artificial Neural Networks (ANN) have been the spearhead of many AI tasks since the 1980s, for this reason they are the most researched field in AI. Multi-Layer Feed-Forward Neural Networks are a type of ANN in which the information moves in one direction, there are no cycles or loops in the network. In recent years PSO has been used in combination with ANNs to optimise the weights of the ANN. The following sections describe Multi-Layer Feed-Forward Neural Networks and how PSO can be used to optimise them.

Feedfoward Neural Network

A Feed-Forward Neural Network (FNN) is an adaptive artificial system which is based on the ideology of the human brain's neural structure. The standard structure of a Feed-Forward Neural Network consists of three layers of nodes (input layer, hidden layer and output layer), where each node in a layer is connected to every node in the following layer. Every connection also has a weight value. All information moves in one direction, from the input layer through the hidden layer(s) to the output layer.

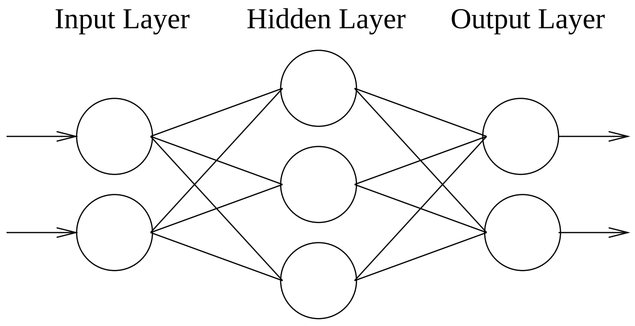

Figure 2.7: Feed Foward Neural Network

The feed-foward pass is used to obtain a classification output using a FNN. Each value in the feature vector is passed to an input node, which means that there are an equal number of input nodes as there are feature dimensions. The output from each input node is then passed to every node in the hidden layer and multiplied by $weight_{input \rightarrow hidden}$, where

$input \rightarrow hidden$ is the connection from the input node to the hidden node. The output from each hidden node is then passed to every node in the output layer and multiplied by $weight_{hidden \rightarrow output}$.

The output vector from the FNN can finally be mapped to a classification. For example if the problem was a three class classification problem, class 1 would be represented as the $(0,0,1)$ vector, class 2 would be represented as the $(0,1,0)$ vector and class 3 would be represented as the $(1,0,0)$ vector. To map an output vector from the ANN to a classification, a winner takes all method is used, where the largest output value is rounded to 1 and then lesser values are rounded down to 0. This means the output vector will equal one of the class vector representations and classified as the corresponding class.

The training of the connection weights is usually done via a method called the back propagation (BP) algorithm. The BP algorithm initially sets all the connection weights to small random values. Then for each particular feed-forward pass, the weight change between each layer is calculated by the following steps.

1. Take the output vector from the feed-forward pass
2. Calculate β, how beneficial the change is (in terms of lower error), for output nodes:
$$\beta_z = d_z - o_z \tag{2.7}$$
3. Compute β for each hidden node:
$$\beta_h = \sum_z w_{h \rightarrow o} o_z (1 - o_z) \beta_z \tag{2.8}$$
4. Compute and store the weight changes for all weights:
$$\Delta w_{i \rightarrow h} = \eta o_i o_h (1 - o_h) \beta_h \tag{2.9}$$
$$\Delta w_{h \rightarrow o} = \eta o_h o_z (1 - o_z) \beta_z \tag{2.10}$$

Genetic Algorithms [4] have also been used, either in combination with the BP algorithm[11] or as a stand alone method[12], to evolve the FNN architecture and weights. Genetic Algorithms have shown to contribute a more effective global search to the BP algorithm which uses a gradient decent search known for it's ability to search local areas very effectively [12].

PSO and FNN Hybridisation

PSO has also shown to be an effective technique for evolving the weights of a FNN. With a pre-defined FNN structure each particle can be encoded as the set of weight values, such as in equation 2.11, where m is the number of input nodes, n is the number of hidden nodes and o is the number of output nodes.

$$particle_i = \begin{pmatrix} w_1 & .. & w_m \\ w_1 & .. & w_n \\ w_1 & .. & w_o \end{pmatrix} \tag{2.11}$$

Every iteration each particle is evaluated by copying the particle input, hidden and output weights into the FNN input, hidden and output weights respectively. The particle fitness is then the error rate of the ANN.

2.5 Overview of Genetic Programming Classification Methods

There are many ways to implement multi-class classification in genetic programming, such as static class boundary determination, slotted dynamic class boundary determination and centered dynamic class boundary determination [13], [5]. This project has a low focus on GP therefore the most basic of methods, static class boundary determination, was used in implement GP classification.

In static class boundary determination the real numbers are split into n slots, where n is the number of classes in the classification task. Therefore each slot is associated with one class. Using the static class boundaries any floating point value outputed from a genetic program can be mapped to a class label. Equation 2.12 dictates the output class label, where T is some constant value, v is the program output.

$$Class = \left\{ \begin{array}{cc} class_1, & v <= 0 \\ class_2, & 0 < v <= T \\ class_3, & T < v <= 2T \\ ..., & ... \\ class_i, & (i-2)*T < v <= (i-1)*T \\ ..., & ... \\ class_n, & v > (n-2)*T \end{array} \right\} \tag{2.12}$$

2.6 Overview of Particle Swarm Optimisation Variants

Particle Swarm Optimisation has shown to be a very successful optimisation technique with very little overhead. However, PSO is often accused of having an early convergence problem. This is due to particles flying toward the best known solution. Therefore if the global optimum solution is not on the path between any given particle and the current *gbest* then it is likely the swarm will converge on a local optimum solution. In many problems a global optimum solution is required therefore this characteristic is undesirable. Some PSO variant methods have been created to try and overcome this limitation and are described in the following sections.

2.6.1 Fitness Distance Ratio Particle Swarm Optimisation

Particle velocity in standard Particle Swarm Optimisation is influenced by the best known particle *pbest* and the best known particle in the neighbourhood, *gbest*. Fitness Distance Ratio Particle Swarm Optimization (FDR-PSO) adds a third influential particle called *nbest*, which is the particle that maximises the fitness distance ratio, calculated by equation 2.13.

$$\frac{Fitness(P_j) - Fitness(X_i)}{|P_{jd} - X_{id}|} \tag{2.13}$$

Where P is the candidate *nbest* particle and |...| denotes the absolute value. The particle velocity is then updated by equation 2.14:

$$PV_{id} = \chi[PV_{id} + \phi_1 r_1(pbest_{id} - X_{id}) + \phi_2 r_2(gbest_{id} - X_{id}) + \phi_3 r_3(nbest_{id} - X_{id}] \tag{2.14}$$

Results have shown that with a higher weighting on ϕ_3, FDR-PSO out performs standard PSO in many optimisation problems [14].

2.6.2 Particle Swarm Optimisation With Mutation

Particle Swarm Optimisation with Mutation is a variation of PSO which uses the genetic operator, mutation, to prevent premature convergence. There is a P_{mute} probability that the mutation operator randomly changes the particle position. The mutation operator introduces unseen parts of solution space into the swarm, in many cases this can contribute to the finding of a better solution than the current *gbest* [15] [16].

PSO with mutation was used to optimise four functions: sphere, sckley, rastrigin and rosenbrock of dimensions 10, 20 and 30. The results found showed that the mutation operator provided significant improvement in performance for the rastrigin and rosenbrock functions for larger dimension values. However the mutation operator actually decreased the performance in the sphere function.

2.6.3 Comprehensive Learning Particle Swarm Optimisation

Comprehensive Learning Particle Swarm Optimisation (CLPSO) is a variation of PSO where the particle velocities are only influenced by one best position rather than two. Equation 2.15 dictates how the particle velocity is calculated.

$$PV_i^d = \chi[PV_i^d + \phi_1 r_1 (pbest_{fi(d)}^d - X_i^d)] \tag{2.15}$$

where $f_i = [f_i(1), f_i(2), ..., f_i(D)]$ defines which particles *pbest* the particle *i* should follow. $pbest_{fi(d)}^d$ can be the corresponding dimension of any particle's *pbest* including its own *pbest*. The decision depends on the learning probability P_c. A random number is generated for each dimension, if the random number is greater than P_c then $pbest_{fi(d)}^d$ refers to its own *pbest* otherwise it will refer to a different particle's *pbest*. Results obtained from using CLPSO to solve multi-model problems show to outperform standard PSO [17].

Chapter 3

Image Sets

Three image sets of varying difficulty were used to test the algorithms described in this project: *shape*, *face* and *texture*. Each image set consisted of multiple classes of images and each individual image contained a singular class and some kind of background or noise. Section 3.1 describes the features that were extracted from each of the images to form the image feature vectors. The remaining sections describe each of the image sets in approximately ascending order of difficulty. However image set difficulty often varies from technique to technique as we will see in later results.

3.1 Image Features Used

The features used for the experiments undertaken in this project are image pixel statistics. The mean and standard deviation of multiple sub-regions from the full image comprise the feature vector. Figure 3.1 depicts 2 and 4 sub-regions respectively, each of which the mean and standard deviation will be extracted from.

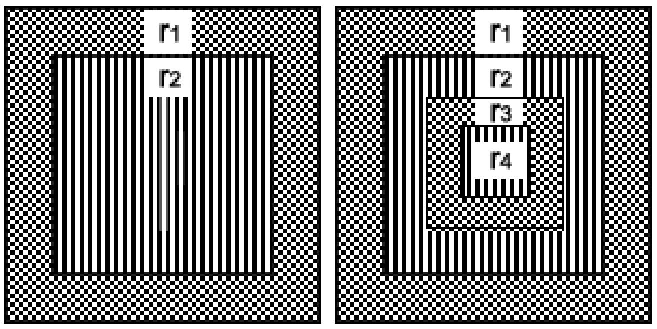

Figure 3.1: Image with Two Sub-Regions (left) and Image with Four Sub-Regions (right)

The motivation for using such primitive features is questionable. Such features could mean lower performance is to be expected than if more features or higher level features were used. However if good classification results can be achieved with a simple set of domain independent features (such as pixel statistics), equal or better results can be achieved by obtaining more features from other sub-regions or by using higher level features that are dependent on the image set. Development of good specific features for a particular task is beyond the scope of this project.

3.2 Shape

The *shape* image set consisted of four classes of shapes: light squares, dark-grey squares, dark-grey circles and black circles, each with varying pixel means and standard deviations. Each class was generated using a Gaussian filter against a uniform background. The *shape* image set consisted of 200 image cut outs (50 from each class). The *shape* training set and testing set were both generated using TFCV for every experiment in this project. Figure 3.2 depicts the four classes of shapes.

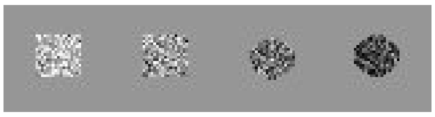

Figure 3.2: Shapes with varying pixel means

The pixel mean and standard deviation of each different class of image are presented in table 3.1. An important observation is that all of the class mean values are well separated. This suggests that the *shape* classes will be fairly easy to distinguish between and high classification accuracy is to be expected..

Class	Mean	Standard Deviation
Light Squares	150	30
Dark-Grey Squares	135	35
Dark-Grey Circles	120	40
Black Circles	105	65

Table 3.1: Shape Image Pixel Mean and Standard Deviations

3.3 Face

The *face* image set consists of images from six different people. The images were taken at different times of the day with varying lighting, facial expressions (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). All the images are taken against a dark homogeneous background and the subjects are in up-right, frontal position. The images are taken from *set1, set2, set3, set8, set15 and set16* in the ORL face database which contains 40 different image sets of people [18] [19]. The *face* image set consisted of 60 image cut outs (10 from each class). The *face* training set and testing set were both generated using TFCV for every experiment in this project. Figure 3.3 depicts the six classes of faces.

Due to the number of possible variations between each image, features generated from the face image set are likely be less separable than those from the Shape image set. Therefore we would expect the classification difficulty of this image set to be a step up from the basic shape image set.

3.4 Textures

The *texture* image set consisted of six similar texture images, *hearingbone weave, woolen cloth, wood grain, bark, straw* and *raffia*. The images were taken by a camera under natural light and can be found in the web-based image database held by SIPI of USC [20]. The *texture* image

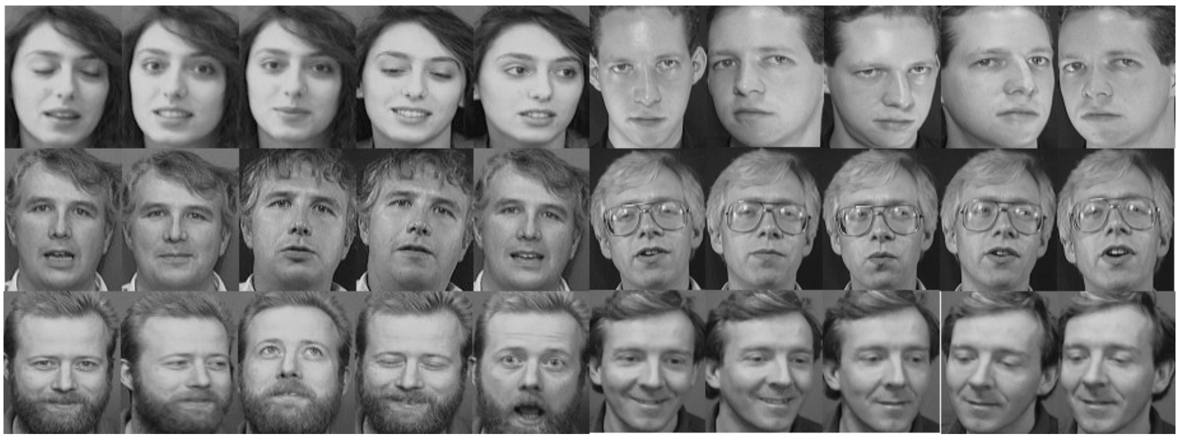
Figure 3.3: Six Classes of Face Images

set consisted of 1350 image cut outs (225 from each class). The training set was created from 675 texture cut out images with an even distribution of classes. The remaining 675 texture cut out images comprised the testing set. Figure 3.4 depicts each respective class of texture that was used for image classification. To the human eye these textures are easily distinguishable, however their pixel statistics are very similar. We would expect the texture image set to be hard to classify.

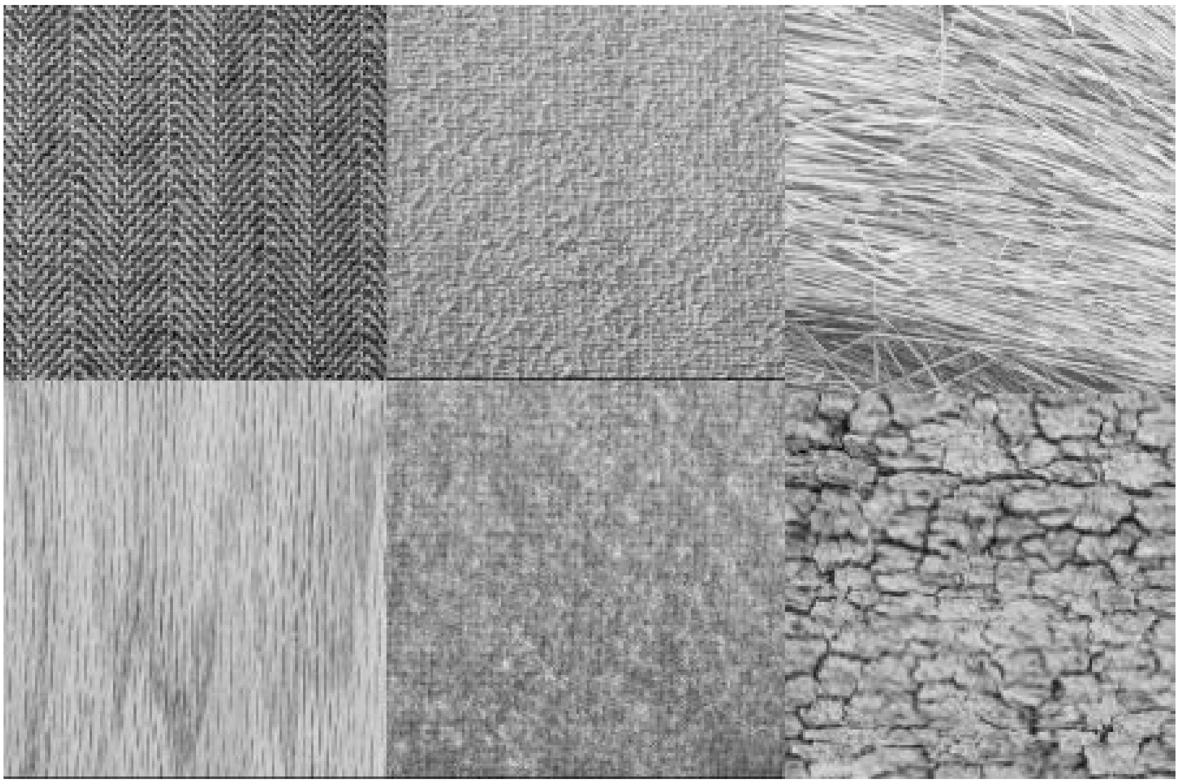
Figure 3.4: Six Classes of Texture Images

3.5 Summary of Image Sets

Table 3.2 contains a summary of each of the image sets.

Dataset	Similarity	Classes	Images	Size
Shapes	Low	4	10x4	30x30
Faces	Medium-High	6	10x6	90x110
Textures	High	6	225x6	40x40

Table 3.2: Summary of Image Sets

Chapter 4

Results for Two Existing Classification Methods

4.1 Overview

In this chapter we are looking to achieve the goal of a baseline for image classification which latter methods can be compared against. Without a baseline it is hard to gauge the ability of a new technique. The two existing techniques used for image classification in these sections are the PSO-Nearest Centroid Method and Genetic Programming, which were both described extensively in earlier chapters.

4.2 Experiment Configurations

The specific configuration settings of a search technique can dictate largely the effectiveness of the resulting classifier produced. This is especially prominent in GP which has such a vast range of configuration abilities. This can be both a blessing and a curse; while GP provides the potential functionality to solve many different kinds of problems, there is a fair amount of overhead in setting a GP configuration correctly to solve even a simple problem. The following sections will describe the various parameter values used by the search techniques used in the baseline experiments and all proceeding experiments.

4.2.1 PSO Configuration

The following is a brief description of each of important parameters used in PSO.

- Swam Size - The number of particles in the swarm.
- Max Iterations - The number of iterations before the optimisation process is terminated.
- Velocity Min Value - The minimum value any dimension value in a particle's velocity vector can obtain. This stops particles from flying too fast and over shooting a solution.
- Velocity Max Value - The maximum value any dimension value in a particle's velocity vector can obtain. This stops particles from flying too fast and over shooting a solution.
- Dimension Min Value - The minimum value any dimension value in the particle's centroid vector can obtain.

- Dimension Min Value - The maximum value any dimension value in the particle's centroid vector can obtain.
- Constriction Factor - A value used to stop the particles velocity growing too fast.

Table 4.1 details the specific values used in PSO for the baseline experiments. These same values were used for every image set.

Parameter Kind	Parameter Name	Value
Search Parameters	Swarm Size	40
	Max Iterations	200
Particle Parameters	Velocity Min Value	-1.0
	Velocity Max Value	1.0
	Dimension Min Value	0.0
	Dimension Max Value	1.0
	Constriction Factor	0.74

Table 4.1: PSO Parameter Values

4.2.2 GP Configuration

The GP configuration defines a set of building blocks and properties that facilitate the solving of a problem efficiently. The following sections describe the building blocks (function and terminal sets) and the properties (parameter values) used in the GP baseline experiments. It is important to note that while it is likely that the GP configuration could have been modified to achieve even greater classification accuracies, this is a baseline experiment which means that the GP configuration is the very much a basic/standard configuration.

Function Set

The set of possible functions used in a GP program was the standard arithmetic operators (with a small exception) and an *if* conditional statement. The exception to the arithmetic operators is the *protected division* operator which will return 0 if the denominator is 0, a characteristic not seen in regular division. The *if* conditional is quite unique in that it takes three arguments: the first is a boolean condition, if the boolean holds the second argument (a subtree) is executed, otherwise the third argument (a subtree) is executed. Table 4.2 details the function set used.

Function	Description
+	Standard arithmetical addition
-	Standard arithmetical subtraction
	Standard arithmetical multiplication
/	Protected division. Returns 0 if the denominator is 0.
if	Executes argument two if argument one is true other executes argument three.

Table 4.2: GP Function Set

Terminal Set

The set of possible terminals used in a GP program consisted of the set of features and the set of floating point numbers in the range [0, 100]. It is important to note that set of features is of varying size depending on the specific experiment. The floating point terminals are generated at the same time as the node and remain static throughout the duration of the evolution process.

Fitness Function

The fitness function used for all GP experiments was the classification accuracy on the training set.

Parameter Values

The following is a brief description of each of important parameters used in PSO:

- Population Size - The number of individuals/genomes in the population.
- Max-Depth - The maximum depth a tree (individual) can obtain.
- Max Generations - The maximum number of generations before the evolution process is terminated.
- Reproduction-rate - Also called the elitism rate, this is the percentage of individuals in the current population that are copied directly into the next generation of individuals, without being modified by any genetic operators.
- Crossover-rate - The percentage of individuals in the mating pool that will be used to reproduce offspring using the crossover operator.
- Mutation-rate - The percentage of individuals in the mating pool that will be used to reproduce offspring using the mutation operator.

Table 4.3 details the specific values used in GP for the baseline experiments. These same values were used for every image set.

Parameter Kind	Parameter Name	Value
Search Parameters	Population Size	1000
	Max-Depth	6
	Max-Generations	100
Genetic Operators	Reproduction-rate	2%
	Crossover-rate	70%
	Mutation-rate	28%

Table 4.3: GP Parameter Values

4.3 Baseline Results

The mean values of 30 runs using PSO-NCC and GP classification techniques, against each of the image sets described in section 3, are presented in table 4.4.

For the Face and Shape data sets, ten-fold cross validation (TFCV) was used, and the accuracy reported is the average accuracy across all ten sub-runs. TFCV is used because the Face and Shape image sets are relatively small and by using TFCV the training and testing sets are maximised.

Dataset	Technique	Features	Accuracy (%)
Shapes	PSO-NCC	4	100.00
		8	100.00
	GP	4	99.10
		8	99.30
Faces	PSO-NCC	4	86.07
		8	85.78
	GP	4	49.89
		8	52.61
Textures	PSO-NCC	4	83.01
		8	76.7838
	GP	4	63.52
		8	61.48

Table 4.4: Baseline Image Classification Results

For the Shapes image set, PSO-NCC obtained perfect results and GP obtained near perfect results. This is to be expected as each different class of shape has a separable set of pixel statistics obtained through a Gaussian distribution. It confirms the hypothesis that the shape image set is easy to classify.

For the Faces image set, PSO-NCC showed to outperform GP by a significant amount of 36.16 %. The reasons for GP struggling on the face image set is that, although TFCV was used on the face image set, the number of training examples was not enough for sufficient training of the GP program. It is also confirmation that GP struggles to obtain good classification accuracy using the static class boundary classification map when the number of classes is large. It is important to note that even though GP only obtained a classification accuracy of 49.89 % with 4 features, it is still better than a random choice classifier. This is because there are six classes of images, a random choice classifier would obtain a mean accuracy of approximately 20 %.

For the Texture image set, PSO-NCC obtained a worse classification accuracy with 8 features than with 4 features. This is quite unusual in most cases however in this case it is probably more a reflection on the suitability of the features selected to classify the textures. Feature selection is a very open ended question and beyond the scope of this project. Interestingly GP was more effective in classifying the Texture image set than the Faces image set, this relates back to the number of training instances. The Texture image set had 180 images per class in the training set compared to the Faces image set which had 9 images per class in the training set. However due to the inseparability of the texture features, GP still struggled to obtain a high classification, its best being 63.52 %.

4.4 Chapter Summary

In this chapter the goal of a baseline for image classification was achieved using two established object classification techniques. The two techniques used were the PSO-Nearest Centroid and Genetic Programming and each technique was tested on three image sets of varying difficulty. The results suggested that PSO-Nearest Centroid method outperformed Genetic Programming in general. The contribution of this chapter is that it will serve as a benchmark to compare new techniques against later in the project.

Chapter 5

PSO Stand Alone Object Classification Techniques

5.1 Overview

The PSONCC technique was described in section 2.4.1. This section describes two new PSO-based methods for object classification which do not rely on any existing object classification techniques. The two techniques are called *Feature Partitioning Particle Swarm Optimisation* (FPPSO) and *Weighted Feature Partitioning Particle Swarm Optimisation* (WFPPSO). In general over this chapter we hope to distill the effective difference between FPPSO and WFPPSO and analyse whether they can do a good enough job on a sequence of multiclass image classification problems. Specifically we are interested what the effective difference between WFPPSO and FPPSO and whether the weighting provides effective benefits.

This chapter is structured as follows. The motivation for the new stand alone techniques are described in section 5.2. This is followed by two sections which describe FPPSO and WFPPSO respectively. Finally the classification results of FPPSO and WFPPSO are presented and analysed in section 5.5.

5.2 Motivation for a Stand Alone PSO technique

PSO has been used for object classification with varying success [21] [22] [23], however these techniques have used PSO in combination with an existing object classification technique. The following sections describe two new PSO techniques that do not rely on any existing object classification techniques. The ideology of feature partitioning stemmed from Genetic Algorithm classification techniques [4] [24], however FPPSO and WFPPSO differ somewhat.

5.3 Feature Partitioning Particle Swarm Optimisation

Feature Partitioning Particle Swarm Optimisation (FPPSO) is a classification technique which tries to find an optimal Partition Matrix (PM) to separate a data set into distinct class groups. PSO is used to evolve a class partition for each class on every feature dimension. Together these form the PM which is the fundamental basis used for object classification. A class partition in this case is simply a lower and upper limit, which together constitute the bounds which a value can fall within.

To classify a feature vector using FPPSO, each feature in the feature vector is first classified separately using the PM. These are termed dimensional classifications. For example, if a feature vector such as $(0.15, 0.35, 0.27)$ was presented to the PM defined in 5.1, class partitions $([0.1, 0.2], [0.25, 0.5], [0.25, 0.4])$ would be matched on dimensions $1, 2$ and 3 respectively. This would return dimensional classifications $(1, 1, 2)$.

$$PM = \begin{pmatrix} (0.1, 0.2) & (0.3, 0.6) & (0.7, 1.0) \\ (0.25, 0.5) & (0.05, 0.1) & (0.6, 1.0) \\ (0.9, 1.0) & (0.25, 0.4) & (0.0, 0.2) \end{pmatrix} \tag{5.1}$$

Secondly, the number of votes each class obtained is calculated. In the example referred to previously, the vote counts would be $(2, 1, 0)$ for classes $1, 2$ and 3 respectively. The class with the most votes is the final classification, in this case the final classification would be class 1 with 2 out of 3 votes. Figure 5.1 depicts the entire classification process.

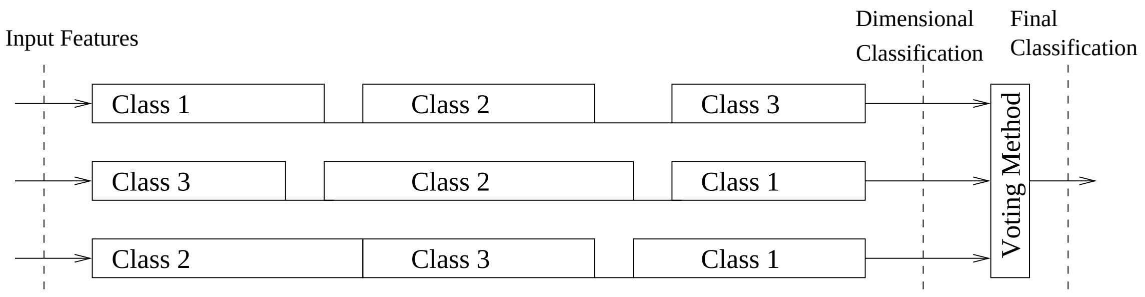

Figure 5.1: FPPSO Classification Process

5.3.1 Particle Encoding and Fitness

The PM is encoded in the particle as a *dimensions* $\times$ *classes* matrix, where each row r in the matrix describes the set of class partitions for $dimension_r$, as shown in Equation 5.2. Each value in the matrix is a value pair, which corresponds to the lower and upper bounds of a partition.

$$particle_i = \begin{pmatrix} (lower, upper)_{(1,1)} & .. & (lower, upper)_{(1,c)} \\ .. & .. & .. \\ (lower, upper)_{(d,1)} & .. & (lower, upper)_{(d,c)} \end{pmatrix} \tag{5.2}$$

The fitness of each partition in the matrix is evaluated individually and is defined by equation 5.3. Formally this equation is known as the sensitivity.

$$Fitness_{i,j} = truepositives/(truepositives + falsepositives) \tag{5.3}$$

5.4 Weighted Feature Partitioning Particle Swarm Optimisation

WFPPSO is an extension of the FPPSO classification algorithm which uses the same framework as FPPSO for classification but includes a number of weights to increase the classification accuracy. Once the best partition matrix has been found by the FPPSO method, weights for each partition and each dimension are learned. For the purpose of this paper the partition weights are called the Partition Weight Matrix (PWM) and the dimension weights are called the Dimensional Weight Vector (DWV).

The PWM is $dimensions \times classes$, for each $PM_{(i,j)}$ value there is a corresponding $PWM_{(i,j)}$ value, which dictates the strength of the $(i,j)th$ Partition. Partitions which are not separable are likely to have lower weights and partitions which are separable are likely to have higher weights. This is because non-separable partitions are likely to misclassify and in order to maximise the fitness function lower weights must be put on partitions that are likely to misclassify.

The DWV is of size *dimensions*. Each DWV_i value dictates the influence of the ith dimensional classification on the final classification. This DWV helps to enhance dimensions with less noise and lower the dimensions that are likely to misclassify.

To classify a feature vector using WFPPSO each of the features must be classified separately using the PM. For example, if an instance such as $(0.15, 0.35, 0.27)$ was presented to the PM defined in 5.4, class partitions $([0.1, 0.2], [0.25, 0.5], [0.25, 0.4])$ would be matched on dimensions $1, 2$ and 3 respectively. This would return dimensional classifications $(1, 1, 2)$.

$$PM = \begin{pmatrix} (0.1, 0.2) & (0.3, 0.6) & (0.7, 1.0) \\ (0.25, 0.5) & (0.05, 0.1) & (0.6, 1.0) \\ (0.9, 1.0) & (0.25, 0.4) & (0.0, 0.2) \end{pmatrix} \tag{5.4}$$

Secondly each dimensional classification is multiplied by their corresponding class weight defined in the CWM. In the example referred to previously, if we had a CWV such as equation 5.5, the $(classlabel : weight)$ pairs would be $(1 : 0.8, 1 : 0.6, 2 : 0.5)$.

$$CWM = \begin{pmatrix} 0.8 & 0.6 & 0.6 \\ 0.6 & 0.8 & 0.5 \\ 1.0 & 0.5 & 0.5 \end{pmatrix} \tag{5.5}$$

Thirdly each weighted dimensional classification is multiplied by their corresponding dimension weight defined in the DWV. In the example referred to previously, if we had a DWV such as equation 5.6, the resultant vector of $(classweight : weight)$ pairs would be $(1 : 0.8, 1 : 0.3, 2 : 0.25)$.

$$DWV = (1.0, 0.5, 0.5) \tag{5.6}$$

The final step is to sum the weighted votes for each class. In the example referred to previously, we would count $(1.1, 0.25, 0.0)$ for classes $1, 2$ and 3 respectively. The class with the most weighted votes is the final classification, in this case the final classification would be class 1. Figure 5.2 depicts the entire classification process.

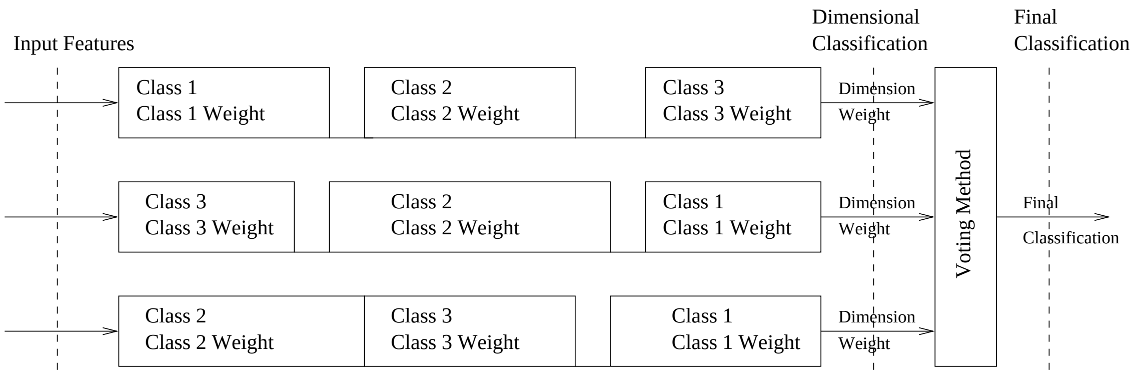

Figure 5.2: WFPPSO Classification Process

5.4.1 Particle Encoding and Fitness Function

The PWM is encoded in each particle as a *dimensions* × *classes* matrix of floating point values in the range [0, 1]. Each row r in the matrix describes the set of partition weights for $dimension_r$. The DWV is encoded in the same particle as a vector of size *dimensions*. Equation 5.7 shows how the PWM and DWV are encoded in one particle.

$$particle_i = \begin{pmatrix} pw_{(1,1)} & .. & pw_{(1,c)} \\ .. & .. & .. \\ pw_{(d,1)} & .. & pw_{(d,c)} \\ dw_1 & .. & dw_d \end{pmatrix} \tag{5.7}$$

The fitness of each particle is the classification accuracy on the training set, using the feature partitions, the particle PWM and the particle DWV in WPPSO.

5.5 Results and Analysis

The mean values of 30 runs using FPPSO and WFPPSO classification techniques, against each of the image sets described in section 3, are presented in table 5.1.

For the Face and Shape data sets, ten-fold cross validation (TFCV) was used, and the accuracy reported is the average accuracy across all ten sub-runs. TFCV is used because the Face and Shape image sets are relatively small and by using TFCV the size of the training and testing sets are maximised.

For the Shapes image set, WFPPSO obtained perfect results and FPPSO obtained near perfect results. This is to be expected as each different class of shape has a separable set of pixel statistics obtained through a Gaussian distribution. However this does suggest that the underlying FPPSO technique is less than satisfactory if it cannot find partitions for perfectly seperable classes.

Dataset	Technique	Features	Avg Accuracy (%)
Shapes	WFPPSO	4	100.00
		8	100.00
	FPPSO	4	99.75
		8	99.98
	PSONCC	4	100.00
		8	100.00
	GP	4	99.10
		8	99.30
Faces	WFPPSO	4	92.44
		8	97.39
	FPPSO	4	72.56
		8	76.78
	PSONCC	4	86.07
		8	85.78
	GP	4	49.89
		8	52.61
Textures	WFPPSO	4	80.02
		8	81.16
	FPPSO	4	61.11
		8	62.91
	PSONCC	4	83.01
		8	76.78
	GP	4	63.52
		8	61.48

Table 5.1: Comparison between classification techniques

For the Faces image set, WFPPSO showed to outperform FPPSO by a significant amount of 20.61 %. The main reason for FPPSO struggling to classify the face image set is that the separability of the feature dimensions is higher than the shape image set. WFPPSO is able to partially overcome this by promoting seperable partitions with high weight values and mitigate noisy partitions with low weight values. It is the first confirmation that the CWM and DWV provide significant benefits in terms of increasing the effectiveness (classification accuracy).

For the Texture image set, both of the techniques struggled to obtain a high classification accuracy. However WFPPSO showed to outperform the regular technique FPPSO by 18.2507 %. This confirms the significance of the CWM and the DWV to the WFPPSO technique.

The new method, WFPPSO, showed in most cases to be the most effective classification technique, being closely challenged by PSONCC in some cases. However when the number of features increased the accuracy achieved by PSONCC dropped while the accuracy achieved by WFPPSO increased. This suggests that WFPPSO is more well behaved than PSONCC and could potentially achieve higher accuracy rates with higher dimension data sets. As this is a relatively simple implementation of WFPPSO, there is potential to achieve even higher classification results in future improved implementations.

The new techniques WFPPSO and FPPSO have showed some promising results on these images sets. However they still struggled in some of the experiments to obtain a high classification accuracy. This is because FPPSO relies on the data set being seperable on the majority of the classes, if this is not true then it will always misclassify to some degree. WFPPSO tries to overcome this somewhat by the use of weights, however it still does not have the ability to transform a non-seperable data set into a seperable data set. For this reason WFPPSO struggles on some classification problems.

5.6 Chapter Summary

In this chapter the goal of a developing two new stand alone PSO classification techniques, WFPPSO and FPPSO, was achieved. The classification result of WFPPSO and FPPSO on a sequence of image sets was described. The new techniques, particularly WFPPSO showed to outperform the existing techniques in the majority of the experiments. The effectiveness of WFPPSO was significantly larger than FPPSO in all cases which suggests that the addition of the weights to the FPPSO technique was valuable. Some limitations of the new techniques were also extracted from the result set.

Chapter 6

A Gaussian Approach to Feature Partitioning PSO

6.1 Overview

This chapter describes a technique which pertains to the experimentation and further expansion of the FPPSO technique. The technique described is called Gaussian Distribution Partitioning Particle Swarm Optimisation (GDPPSO).

The goal of this chapter is to dig a bit deeper into the capabilities of Partitioning using PSO. More specifically it looks to find out whether GDPPSO provides any effective benefits over the regular FPPSO technique.

6.2 The Gaussian Approach

One major limitation or drawback that could be suggested about the current implementation of Feature Partitioning is that there are particular value ranges on some dimensions, which are not covered by any FPPSO. Currently an instance value which falls into one of these unknown regions, is classified as the class label associated with the closest Feature Partition.

This technique is often inaccurate as each class has a different spread of data which can make the Euclidean distance an insufficient measure. For example, the feature value in feature 6.1 will be classified as class 0 using the current technique when it is more likely to be class 1.

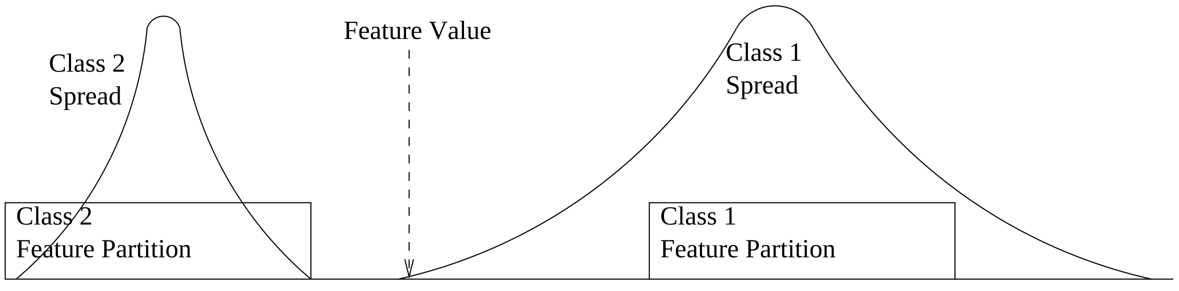

Figure 6.1: Feature Partitions versus Gaussians

A gaussian approach to FPPSO was created to gain a better insight into whether this is a problem with the current FFPSO implementation. The Gaussian approach, Gaussian Distribution Partitioning Particle Swarm Optimisation (GDPPSO) follows much the same process as the regular FPPSO however it has a few fundamental differences. Rather than evolve the lower and upper limits of the each feature partition, GDPPSO evolves a mean and standard deviation for each class on every dimension which can each be used to created a gaussian distribution. A particle is now encoded by equation 6.1 where *d* is the number of dimensions and *c* is the number of classes.

$$particle_i = \begin{pmatrix} (\mu_1, \sigma_1) & .. & (\mu_1, \sigma_c) \\ .. & .. & .. \\ (\mu_d, \sigma_1) & .. & (\mu_d, \sigma_c) \end{pmatrix} \tag{6.1}$$

The classification of a feature vector in GDPPSO occurs by first obtaining the probability of each class on each dimension. These probabilities are calculated by the Gaussian distributions created by the mean and standard deviation values found within each particle. The probabilities can be visualised in a $dimensions \times classes$ matrix where the $(i, j)th$ value in the matrix is the probability of a feature vector being the *ith* class according to the *jth* dimension. An example can be seen in equation 6.2.

$$\begin{pmatrix} 0.8 & 0.2 & 0.3 \\ 0.6 & 0.1 & 0.2 \\ 0.5 & 0.3 & 0.4 \end{pmatrix} \tag{6.2}$$

The final classification using GDPPSO is obtained by summing the probability for each class. By equation 6.2 we would obtain $(1.9, 0.6, 0.9)$ for classes 1, 2 and 3 respectively. The final classification would be class 1 with the summed probability 1.9.

6.2.1 Results and Analysis

The mean values of 30 runs using FPPSO and GDPPSO classification techniques are presented in table 6.1.

For the Face and Shape data sets, ten-fold cross validation (TFCV) was used, and the accuracy reported is the average accuracy across all ten sub-runs. TFCV is used because the Face and Shape image sets are relatively small and by using TFCV the training and testing sets are maximised.

The results presented in table 6.1 suggest that using a fully probabilistic approach for partitioning is not as effective as using hard boundaries for partitioning for these particular classification problems. However the technique still being in its infant stages, there are many changes that could be made to the implementation to potentially increase the effectiveness of the technique. This is a concern for future work, as Gaussian distributions have been used, via other methods, as an effective approach for classification [5].

Dataset	Technique	Features	Avg Accuracy (%)
Shapes	GDPPSO	4	99.56
		8	99.45
	FPPSO	4	99.75
		8	99.98
Faces	GDPPSO	4	58.72
		8	54.39
	FPPSO	4	72.56
		8	76.78
Textures	GDPPSO	4	65.27
		8	63.57
	FPPSO	4	61.11
		8	62.91

Table 6.1: Comparison Gaussian and Non-Gaussian Partitioning

6.3 Chapter Summary

In this chapter the goal of describing an experimental technique which expands on the regular FPPSO technique was achieved. The new technique was called GDPPSO. The initial goal of GDPPSO was to overcome the uncertainty of classes having different data spreads and also to verify whether using a probabilistic approach would provide any effective benefits. However the results found that the original ideaology was not reflected in the effectiveness of GDPPSO did. Nonetheless it is still valuable insight and something to investigate further in future work.

Chapter 7

Particle Swarm Optimisation With Simulated Annealing

7.1 Overview

This section extends the standard PSO to incorporate the local search technique Simulated Annealing. The new technique is called Particle Swarm Optimisation Simulated Annealing (PSOSA). By integrating Simulated Annealing into how the particles fly through the solution space, we would expect that the new technique mitigates early convergence. Ultimately this should mean PSOSA produces better classifiers.

In general over this chapter we hope to distill whether PSOSA can effectively mitigate the early convergence problem in PSO. Specifically we are interested in how the PSOSA version of WFPPSO and PSONCC compare to the regular PSO version of WFPPSO and PSONCC and also what are some of the reasons for the success or failure of PSOSA.

This chapter is structured as follows. The early convergence problem in PSO is described in section 7.2 which forms the motivation for creating PSOSA. A general overview of Simulated Annealing is described in section 7.3 followed by a description of how it is combined with PSO to form PSOSA. Finally the classification results of PSOSA are presented and analysed in section 7.5.

7.2 Early Convergence Problem in PSO

Hill climbing is a local search technique which from any given state will choose the best neighboring state as its successor. A search technique that never makes a move towards a worse state, such as hill climbing, is prone to converge at a local optimum solution. Simulated Annealing is a local search technique which aims to avoid the limitations of greedy search algorithms such as hill climbing. PSO is not as greedy as hill climbing in that it can move to a worse state, however it does still suffers from a similar problem. If the global optimum solution is not on the path from any given particle to the *gbest* it is unlikely that the global optimum solution will not be found due to the swarm being stuck in a local minimum. If Simulated Annealing is integrated into the movement of particles there is a higher chance the swarm can escape a local minima.

7.3 Simulated Annealing

Simulated Annealing is a generic probabilistic meta-algorithm for global optimisation problems. The SA ideology comes from metallurgy, where annealing is the process used to temper or harden metals and glass by heating them to a high temperature and then gradually cooling them [25]. Rather then take the next best move, simulated annealing also calculates a random move. If the random move improves the situation it is always taken, otherwise, the algorithm may accept the move with some probability less than 1. The probability decreases exponentially with the badness of the move. The probability also decreases as the *temperature* T goes down with time. This means bad moves are more likely to be allowed at the start when the temperature is high, and they become more unlikely as T decreases. Equations 7.2 and 7.2 define the probability P of accepting the random move, where f is the fitness and k and λ are some constant values.

$$P \propto T^{-1} \tag{7.1}$$
$$P = ke^{-\lambda f} \tag{7.2}$$

7.4 PSO and Simulated Annealing Integration

Simulated Annealing can be integrated into the movement of particles in PSO. This is implemented by each iteration, offering two possible movements to a particle. One of the movements will be the regular movement calculated by equation 2.3. The other movement will be a random move. The random move will be accepted by some probability P defined by equation 7.1. Specifically the following steps are taken each time a particle moves:

1. Calculate a random position in the solution space.
2. Calculate the regular new position P of the particle by equations 2.3 and 2.4.
3. Evaluate the fitness of each of the preceeding positions.
4. If the random position is better than P move to the random position.
5. Otherwise calculate the temperature T by equation 7.3.

$$T = (iterations - iteration)/iterations \tag{7.3}$$

6. Take the random move with a probability proportionate to T.
7. Otherwise move to P.

The introduction of the random move will introduce more areas of the search space to the Swarm and ultimately increase the chances of the global optimum solution being found. Figure 7.1 depicts the regular particle movement versus particle movement with simulated annealing.

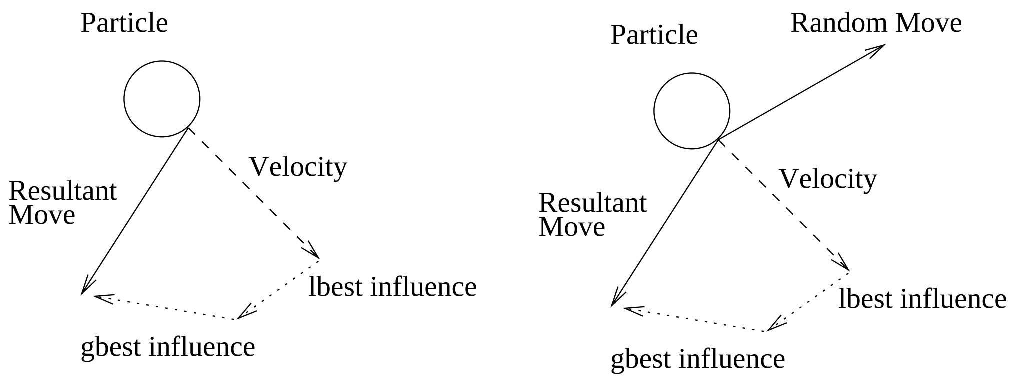

Figure 7.1: Regular Particle Movement (left) and Simulated Annealing Particle Movement (right)

7.5 Results and Analysis

The mean values of 30 runs using WFPPSO and PSONCC classification techniques with and without simulated annealing are presented in table 7.1.

For the Face and Shape data sets, ten-fold cross validation (TFCV) was used, and the accuracy reported is the average accuracy across all ten sub-runs. TFCV is used because the Face and Shape image sets are relatively small and by using TFCV the training and testing sets are maximised.

A consistent pattern seen across all of the of the experiments is that the addition of Simulated Annealing provided no increase in effectiveness (classification accuracy). This could be due to a number of factors.

The first reason could be that PSOSA is simply not suited for these particular data sets. More specifically their is not a diverse range of solutions to these classfication problems, therefore by randomly changing the positions of the particles the local optimisation of the classifier will be disrupted.

A second reason could be related to this particular implementation, a close observer will notice that definition of Simulated Annealing does not match this implementation directly. The text book definition of Simulated Annealing takes the random move proportionate to the temperature and the exponential badness of the move. This implementation only takes the temperature into account. This means that too many bad moves are being introduced to the swarm and consequently affecting quality of the final classifier produced.

Dataset	Technique	Simulated Annealing	Features	Avg Accuracy ($\mu \pm \sigma$)
Shapes	WFPPSO	True	4	99.99 ± 0.0014
		False	4	100.00 ± 0.0000
		True	8	100.00 ± 0.0000
		False	8	99.98 ± 0.0000
	PSONCC	True	4	99.27 ± 0.0086
		False	4	100.00 ± 0.0000
		True	8	99.34 ± 0.0090
		False	8	100.00 ± 0.0000
Faces	WFPPSO	True	4	89.15 ± 0.0122
		False	4	92.44 ± 0.0000
		True	8	93.17 ± 0.0113
		False	8	97.39 ± 0.0000
	PSONCC	True	4	69.86 ± 0.0170
		False	4	86.07 ± 0.0000
		True	8	62.98 ± 0.0150
		False	8	85.78 ± 0.0000
Textures	WFPPSO	True	4	73.74 ± 0.0194
		False	4	80.02 ± 0.0000
		True	8	73.01 ± 0.0266
		False	8	81.16 ± 0.0000
	PSONCC	True	4	65.98 ± 0.0397
		False	4	83.01 ± 0.0000
		True	8	64.86 ± 0.0242
		False	8	76.78 ± 0.0000

Table 7.1: Comparison between classification techniques with and without Simulated Annealing

7.6 Chapter Summary

In this chapter the goal of a integrating Simulated Annealing into PSO to create a new technique was achieved. The new technique was called PSOSA. The initial goal of PSOSA was to try and mitigate the early convergence problem in PSO however some unexpected results were found. The addition of Simulated Annealing provided no benefits in terms of effectiveness. However these results are not useless as any research is valuable knowledge for future reference.

Chapter 8

Conclusions

At the start of this project some initial goals were hoped to be achieved, they are outlined in 1.1. The following sections look at the overall research questions that were outlined and how the actual results compare to what was initially expected. Section 8.1 describes the overall conclusions in relation to the broad goals that were outlined. Section 8.2 describes the conclusions in relation to some of the more specific research questions that were outlined throughout the project.

8.1 Overall Conclusions

The main goal of this project was to construct and describe a stand alone Particle Swarm Optimisation technique for classification, which did not rely on an existing classification algorithm. Inspection of the results found in section 5.5 suggest that this goal was successfully achieved through the FPPSO and WFPPSO classification techniques. While WFPPSO is still not perfect, it does not rely on any established classification algorithms such as that of PSONCC and also provides a solid foundation to expand further research from. It is a new contribution to PSO and will help its progression in general classification tasks.

A secondary goal was to analyse and address the early convergence problem in PSO by creating a PSO variant called Particle Swarm Optimisation Simulated Annealing. The results found in section 7.5 suggest that this goal was addressed but not achieved in the grand scheme of things. PSOSA provided not benefits to the effectiveness of either the PSONCC and WFPPSO classification techniques. However the investigation is still a valuable insight.

8.2 Specific Conclusions

In section 1.1.1 some specific research questions were outlined. The first question concerned the comparison of the effectiveness of established classfication techniques (PSO-NCC and GP). The two techniques had no difficulty classifying the *shape* data set, however as the number of classes and the difficulty increased in the image sets PSO-NCC outperformed the basic GP technique.

The second question was whether the new stand alone classification techniques, WFPPSO and FPPSO, could outperform the baseline classification techniques on a sequence of image classification tasks. The results presented in section 5.5 showed that the WFPPSO classification technique outperformed both the baseline techniques in close to all of the experiments.

FPPSO on the other hand struggled to classify the harder image sets (face and texture). FPPSO underperformed the PSO-NCC method and outperformed GP in most cases. From these findings we can also distil that the CWM and DWV are a valuable contribution to the FPPSO technique.

The last question concerned whether PSOSA could mitigate the the early convergence problem in PSO. The results in section 7.5 suggest that this is false, however it could be that PSO does not suffer from an early convergence problem on these particular data sets. Further research is necessary to make a full conclusion on this question however the findings in this project suggest PSOSA does not help with early convergence.

8.3 Future Work

The following section describes some of the ideas related to a technique, called Multi-Dimensional Feature Partitioning Particle Swarm Optimisation, that I was interested in implementing. However due to time constraints it must be left to future work.

8.3.1 Multi-Dimensional Feature Partitioning Particle Swarm Optimisation

One characteristic of the partitioning algorithms discussed in this project is the assumption that each feature dimension is independent. This is in many cases an incorrect assumption. A basic example that exploits this assumption is the XOR problem as depicts in figure 8.1.

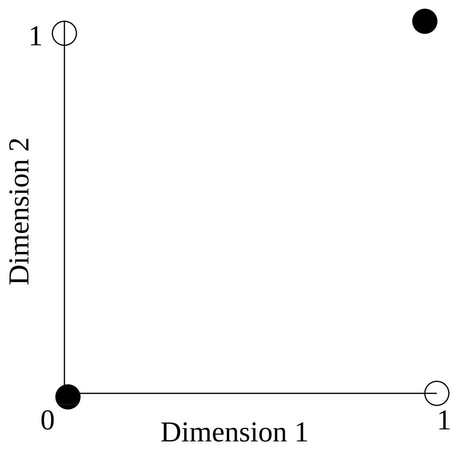

Figure 8.1: XOR Problem

Using independent dimensions it is not possible to classify an instance correctly. However if both dimensional values dictate the classification it is then possible to classify all instances correctly. This is a behaviour similar to that seen in decision trees [25].

If we were to use MDWFPPSO to solve the XOR problem we would first evolve the feature partitions in much the same manner as FPPSO, the resultant partitions would be approximately those found in equation 8.1.

$$FeaturePartitions = \begin{pmatrix} (0, 0.1) & (0.9, 1.0) \\ (0, 0.1) & (0.9, 1.0) \end{pmatrix} \tag{8.1}$$

The weighting pass is where the difference comes in. Extra weight matrices are evolved to describe the relationship between any n dimensions and a given class. For the XOR problem we need two extra matrices, one to describe the relationship between dimensions 1 and 2 and the class label 1. The other matrix describes the relationship between dimensions 1 and 2 and the class label 2. These matrices are displayed in equations 8.2 and 8.3.

$$Class1Weights = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} \tag{8.2}$$

$$Class2Weights = \begin{pmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \end{pmatrix} \tag{8.3}$$

For the matrix 8.2 pertaining to class 1 there is a high weight put on the upper left weight, which means that if the value is between partition 1 on dimension 1, $(0, 0.1)$ and partition 1 on dimension 2, $(0, 0.1)$, return a high weight for class 1. Likewise if it is between partition 2 on dimension 1, $(0.9, 1.0)$ and partition 2 on dimension 2, $(0.9, 1.0)$, return a high weight for class 1.

The matrix 8.3 pertaining to class 2 is the inverse of matrix 8.2 which is what we would expect for the XOR problem.

PSO can be used to evolve the values found in the matrices described above. However there is one major limitation about with this technique. Assume we have a much larger classification problem, such as six classes where each feature vector was of eight dimensions. If we were to evolve a multi-dimensional matrix to describe the relationship between any n features and a particular class a very large amount of computational power would be needed. Therefore some kind of heuristic is needed to prune redundant matrices or a user-defined value is needed which dictates the largest number of dimensions to be combined. These are all questions to be answered in future work and are beyond to scope of this project.

Bibliography

[1] M. Zhang, "Pixel based neural networks for multiclass object detection," tech. rep., Victoria University School of Mathematical Computing Sciences, 2001.

[2] C. Darwin, *The Origin of Species: By Means of Natural Selection*. Albemarle Street, London: John Murray, 1866.

[3] J. R. Koza, *Genetic Programming On the Programming of Computers by Means of Natural Selection*. Cambridge, Massachusetts: The MIT Press, 1992.

[4] H. B. Kamepalli, "The optimal basics for gas," tech. rep., Regional Engineering College Warangal, India, April 2001.

[5] W. Smart, "Genetic programming for multiclass object classification," tech. rep., Victoria University School of Mathematical Computing Sciences, 2005.

[6] S. W. Wilson, "Zcs: A zeroth level classifier system," *Evolutionary Computation*, vol. 2, no. 1, pp. 1–18, 1994.

[7] A. Geyer-Schulz, "Holland classifier systems," *SIGAPL APL Quote Quad*, vol. 25, no. 4, pp. 43–55, 1995.

[8] K. Kennedy and R. Eberhart, "Particle swarm optimisation," tech. rep., Purdue School of Engineering and Technology, 1995.

[9] K. Kennedy and R. Eberhart, "A new optimizer using particle swarm theory," tech. rep., Purdue School of Engineering and Technology, 1995.

[10] T. Blickle and L. Thiele, "A mathematical analysis of tournament selection," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 9–16, Morgan Kaufmann, 1995.

[11] D. White and P. Ligomenides, "Gannet: A genetic algorithm for optimizing topology and weights in neural network design," in *New Trends in Neural Computation*, pp. 322–327, Springer Berlin / Heidelberg, January 1993.

[12] J. R. Koza and J. P. Rice, "Genetic generation of both the weights and architecture for a neural network," in *In International Joint Conference on Neural Networks*, pp. 397–404, IEEE, 1991.

[13] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in *Proceedings of the 2001 Congress on Evolutionary Computation, 2001.*, pp. 1070–1077, 2001.

[14] T. Peram, K. Veeramachaneni, and C. Mohan, "Fitness-distance-ratio based particle swarm optimization," *Swarm Intelligence Symposium*, 2003.

[15] A. Stacey, M. Jancic, and I. Grundy, "Particle swarm optimization with mutation," tech. rep., Dept. of Math. and Stat., R. Melbourne Inst. of Technol., Australia, 2003.

[16] H. N. and I. H., "Particle swarm optimisation with gaussian mutation," in *Proceedings of the IEEE Swarm Intelligence*, 2003.

[17] J. Liang, A. Qin, P. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *Evolutionary Computation, IEEE Transactions*, vol. 10, 2006.

[18] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification.," in *Proceedings of the 2nd IEEE workshop on Applications of Computer Vision*, 1994.

[19] "Orl face image set," April 2008. http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/data/.

[20] "Usc texture database," April 2008. http://sipi.usc.edu/database/database.cgi?volume=textures.

[21] C. guang Chang, D. wei Wang, Y. chen Liu, and B. ku Qi, "Application of particle swarm optimization based bp neural network on engineering project risk evaluating," *International Conference on Natural Computation*, vol. 1, pp. 750–754, 2007.

[22] H.-B. G. Liang Gao, Chi Zhou and Y.-R. Shi, "Combining particle swarm optimization and neural network for diagnosis of unexplained syncope," in *Computational Intelligence and Bioinformatics*, pp. 174–181, Springer Berlin / Heidelberg, September 2006.

[23] P. B. Paula Brito, Guy Cucumel and F. de Carvalho, *Partitioning by Particle Swarm Optimisation*, ch. 2. Springer Berlin Heidelberg, 2007.

[24] A. G. H. and S. I., "A genetic algorithm for classification by feature partitioning," in *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993.

[25] S. J. Russel and P. Norvig, *Artificial Intelligence A Modern Approach*. Pearson Education Inc, second ed., 1995.

[26] M. Zhang and U. Bhowan, "Pixel statistics and program size in genetic programming for object detection," tech. rep., School of Mathematical and Computer Sciences, Victoria University of Wellington, 2004.

[27] A. W. Mohemmed and Z. Zhang, "Particle swarm optimization for data classification," tech. rep., School of Mathematical and Computer Sciences, Victoria University of Wellington, 2005.