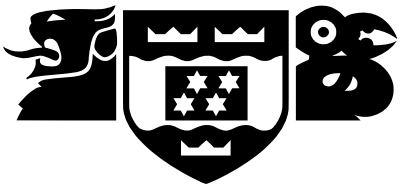
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

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

Ph.D. Proposal: Particle Swarm Optimisation for Feature Selection

Bing Xue

Supervisors: Mengjie Zhang and Will Browne

Submitted in partial fulfilment of the requirements for
PhD.

Abstract

Classification problems often have a large number of features in the datasets, but not all of them are useful for classification. Irrelevant and redundant features may even reduce the performance. *Feature selection* and *feature construction* can improve classification accuracy by selecting relevant features and constructing high-level features. However, feature selection and construction are difficult because of feature interactions and the large search space. Existing methods, such as greedy search and stochastic search, suffer from a variety of problems, such as stagnation in local optima and high computational cost. Therefore, an efficient global search technique is needed to address these problems. *Particle swarm optimisation* (PSO) is such a global search technique, which is computationally cheap and can converge fast. PSO has been successfully applied in many areas, but its potential for feature selection and construction has not been exhaustively explored. This work proposes to develop a PSO based approach to feature selection and construction in classification problems. The goal is to increase the classification accuracy, improve the computational efficiency and simplify the learnt algorithms by using only the selected or constructed features for classification. This will be achieved by developing new PSO based approaches to feature selection, single feature ranking, multi-objective feature selection and feature construction. The proposed approaches are expected to select one or more complementary feature subsets and construct high-level features to better describe the problem. This work also intends to develop a good evaluation function that can obtain an optimal feature subset for a given classification algorithm.

Contents

Chapter 1

Introduction

This chapter introduces this research proposal. It starts with the problem statement, then outlines the motivations, research goals and the organisation of this proposal.

1.1 Problem Statement

In many fields, such as machine learning, the data space representation of a dataset is defined by a set of features. The quality of the data space representation is one of the most important factors which influence the performance of a machine learning algorithm [93]. Inductive learning, where an agent learns from a set of observations, is a widely-practiced paradigm of machine learning. In this paradigm, the quality of the data space representation can be increased by using feature selection or feature construction to change the search space of the machine learning algorithm [61].

Feature selection is an important strategy in improving the representation of a problem. In many situations, which features are relevant is often unknown especially when the domain knowledge is unavailable or incomplete. Therefore, a large number of features are introduced to the input space to better present the problem. However, how well the features represent the problem depends mainly on the useful information that they contain, not on the total number of the introduced features. The presence of irrelevant and redundant features may mask or obscure the distribution of truly relevant features, and hence reduce the representational power of the whole feature set [115]. Meanwhile, a large number of features leads to the curse of dimensionality, which is a major obstacle in many machine learning tasks like classification [28]. Feature selection is an effective treatment for this situation. Feature selection (See Figure 1.1) aims to select a subset of relevant features that are necessary and sufficient to describe the target concept [19]. By reducing the irrelevant and redundant features, feature selection could decrease the dimensionality of the input space, reduce the amount of data needed for the learning process, shorten the running time, simplify the structure and/or improve the performance of the learnt models [19]. Naturally, an optimal feature subset is the smallest feature subset that can obtain the optimal performance, which makes feature selection a multi-objective problem [83, 107]. Note that feature selection algorithms choose a set of features from original features, and do not create new features.

The existing feature selection methods can be broadly classified into two categories: filter approaches and wrapper approaches. The search process in filter methods is independent of a learning algorithm and they are argued to be computationally less expensive and more general than wrapper approaches [19]. On the other hand, wrapper approaches conduct a search for the optimal feature subset using the learning algorithm itself as part of the

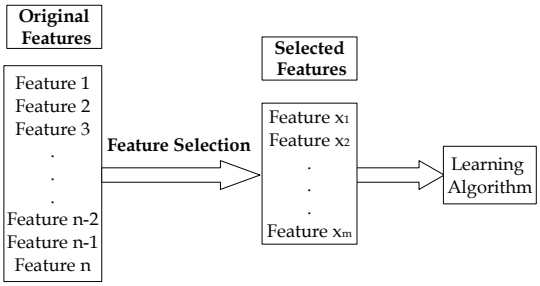


Figure 1.1: Feature selection, where $m < n$ and {Feature x_1 ... Feature x_m } $\in$ {Feature 1, ... Feature n } without replacement.

evaluation function. In a wrapper model, a feature selection algorithm exists as a wrapper around a learning algorithm and the learning algorithm is used as a "black box" by the feature selection algorithm. By considering the performance of the selected feature subset on a particular learning algorithm, wrappers can usually achieve better results than filter approaches [50].

Feature construction is a means of enhancing the quality of representation by transforming the original representation to a new representation in which the learning capability of an agent (e.g. a learning algorithm) can be improved [77]. Feature construction is an effective treatment for problems with epistatic features, which are usually more difficult than the problems with linearly separable features. Epistasis can be defined as interactions between features, where groups of several features working together are relevant but not the individual features alone [68]. Epistasis happens frequently in classification tasks. Feature construction can usually address such a difficult problem by constructing new high-level features based on the original features to build a more appropriate representation to describe the target concept [117]. In feature construction, when the original input space has a quantitative representative (for example when the features are attribute-value and values are numerical), the constructed features are a set of mathematical formulas. In case of nominal values in the original input space, the constructed features are a combination of logical and mathematical expressions.

Both feature selection and construction can individually change the input space representation of a dataset. It is also common to see the combined use of these two methods [62]. Feature selection can be used before feature construction to select a group of the original features that are then used in the feature construction process. Feature selection can also be applied after feature construction to remove redundancy.

However, both feature selection and construction are NP-hard combinatorial problems [110, 7] and in most cases the search space is too large to do an exhaustive search. Greedy feature selection and construction approaches usually suffer from the problem of becoming stuck in local optima [102]. In order to solve these problems, a heuristic search technique is needed to address these problems.

Particle swarm optimisation (PSO) is a heuristic search technique proposed by Kennedy and Eberhart in 1995 [21, 43], which is inspired by social behaviour, such as birds flocking and fish schooling. As a relatively new evolutionary technique, PSO is based on the idea of swarm intelligence. The underlying phenomenon of PSO is that knowledge is optimised by social interactions where the thinking is not only personal but also social. Compared with other techniques, such as genetic algorithms (GAs) and genetic programming (GP), PSO is easier to implement, has fewer parameters and can converge more quickly [22]. PSO is inherently continuous, i.e. it encodes variables into real values. Several discrete PSO

algorithms have also been proposed in the literature [17, 85, 65]. Both continuous PSO and discrete PSO have been shown to be a promising method for feature selection problems [60, 36, 71, 16, 17, 100]. However, PSO has never been used for feature construction.

1.2 Motivations

1.2.1 Challenges of Feature Selection and Feature Construction

Both feature selection and construction are difficult combinatorial problems [110, 7], especially when the number of the features is large. The task is challenging because:

1. an exhaustive search of the entire feature space is practically impossible in most situations. The number of possible feature subsets increases exponentially with respect to the number of original features (2^n possible subsets for n features) [32].

2. there can be two-way, three-way or complex multi-way interactions among features. As a result, a feature, which is weakly relevant or even completely irrelevant to the target concept by itself, can significantly improve classification accuracy if it is complementary to other features.

3. relevant features may be redundant because of feature interactions. A relevant feature may become redundant or even noisy when working together with other features. Therefore, the omission of some of them can remove unnecessary complexity (and noise) and improve classification accuracy.

4. high feature correlation does not imply absence of feature complementarity. The combination of highly correlated features produces redundancy, but these features could be complementary to each other because of feature interactions. There can be different degrees of redundancy and complementarity between correlated features in the feature space.

1.2.2 Why PSO

Development of efficient algorithms for feature selection and constructions is an open issue. In order to avoid exhaustive search, greedy algorithms are introduced to solve feature selection [108, 64] and feature construction [61] problems. However, such greedy feature selection and construction approaches usually suffer from the problem of becoming stuck in local optima. Therefore, a heuristic search technique is needed to solve these tasks. PSO is a heuristic search method. As a powerful search technique, PSO has been successfully implemented in many combinatorial problems, such as process planning and scheduling [30], vehicle routing problem [2], and electric power systems [6].

In feature selection and construction problems, the size of search space increases exponentially with respect to the number of features, which causes the problem of high computational cost, especially for wrapper approaches. Due to the advantages of a few parameters and fast convergence [22], PSO has high potential to solve feature selection and construction problems efficiently.

For most feature selection and construction tasks, a good feature set is a group of complementary features. Features from the complementary group working with other features may not achieve good performance. Therefore, standard evolutionary operators, such as crossover and mutation, may not help the algorithm to produce better solutions when it reaches a near-optimal solution. The standard PSO does not use any standard evolutionary operators. In PSO, particles update their positions dimension by dimension to search for the

best solution, which reduces the probability of destructing a good feature combination. The good feature combination can be saved and the swarm searches for better feature subsets by adding or deleting one or more features every evaluation.

Feature interactions (epistasis) lead to the second, the third and the fourth difficulties (discussed above) in feature selection and construction tasks. Interactions between individuals are an important characteristic of the PSO algorithm. Knowledge in PSO is optimised by social interactions where the thinking is not only personal but also social. Individuals in PSO interact and share information with each other to search for the optimal solution, which is one of the main differences between PSO and other evolutionary computation techniques, such as GAs and GP. Due to feature interaction problems, the interactive nature of PSO is an advantage of using PSO for feature selection and construction.

1.2.3 Limitations of Current PSO for Feature Selection and Feature Construction

Representation, Topology and Search Mechanism

Most of the PSO applications use continuous PSO, but there are a few applications for discrete optimisation problems. Both feature selection and construction tasks are discrete problems, thus a discrete PSO is needed to solve these problems.

The most widely used discrete PSO is binary PSO (BPSO) proposed by Kennedy and Eberhart [44]. BPSO preserves the fundamental concept of the PSO algorithm, that is, the knowledge is optimised by social interactions within the population. However, not all important characteristics of the PSO algorithm are completely present in BPSO. This can be seen from the following aspects.

1. According to the main principles of PSO, the position of the next generation is based on its current position and velocity. However, in BPSO, each dimension of velocity is transformed by a sigmoid function, and then the position is updated solely by comparing the transformed velocity with a random generated number, which is in interval [0, 1]. If it is larger than the random number, the position becomes 1. Otherwise, the position becomes 0. Therefore, the value of the position of a particle in next generation is only dependent on its current velocity instead of dependent on both its current value and velocity.

2. Although BPSO also employs the same velocity update equation as continuous PSO, the sigmoid function and position update equations in BPSO show that a relative large velocity the probability of adjusting the position will decrease. However, it is usually expected that higher changing probability as the velocity increases.

3. A large maximum velocity in BPSO decreases the range explored by a particle, but it is expected a large maximum velocity could encourage a particle to search a larger space [44].

4. In continuous PSO, the velocity of a particle is a linear combination of its previous velocities. In BPSO, the sigmoid function that is used to limit the velocity between 0 and 1, makes the problem non-linear.

When using BPSO for feature selection, according to the traditional encoding scheme, the dimensionality of the search space is the number of features. A bit encoded mask shows whether a feature is selected or not. This encoding scheme makes the task a high-dimensional complex problem if the number of available features is large. A good encoding scheme to avoid such a situation needs to be investigated.

Topology structure is one of the key elements that influence the performance of PSO [22]. Research has shown the effects of topology on the performance of CPSO [22], but the influence of topology on BPSO has not been investigated.

When using BPSO for feature selection and construction, computational efficiency of the algorithm is important in real-world applications. In PSO based approaches, the majority of the computational time is used in evaluating the fitness of each individual. PSO shares many similarities with GA and research shows that GAs spend approximately a third of the time on evaluating already evaluated candidate solutions [18]. So it is highly likely that PSO also wastes much time in evaluating the already evaluated solutions. Feature selection process is computationally expensive mainly because of fitness evaluation, especially in a wrapper model. Therefore, the computational efficiency of the algorithm can be improved if it evaluates one candidate solution only once during the evolving process. However, not much research has been conducted on this topic.

Single Objective vs Multi-objective Feature Selection Approaches

Feature selection has two main objectives in classification problems, which are maximising the classification accuracy and minimising the number of features needed. Therefore, according to the objectives, feature selection approaches can be classified into three different categories as follows.

1. Feature selection approaches aim to maximise the classification accuracy irrespective of the number of selected features (See the first and second definitions for feature selection in Section 2.2).

   Most of the existing approaches fall into this category [100, 78]. However, when the number of features is large, these algorithms suffer from a variety of problems, such as becoming stuck in local optima, being computationally expensive, overfitting and poor generalisation [28].

2. Feature selection approaches aim to select a certain number of features and maximise the classification accuracy (See the third definition for feature selection in Section 2.2).

   Many existing approaches can select a certain number of features with the goal of maximising the classification accuracy, such as sequential forward selection (SFS) [108] and sequential backward selection (SBS) [64]. However, these methods usually suffer from the problem of becoming stuck in local optima [102].

3. Feature selection approaches aim solve the feature selection problem as a multi-objective task to maximise the classification accuracy and also minimise the number of selected features (See the fourth definition for feature selection in Section 2.2).

   Solving feature selection tasks as multi-objective problems is more challenging than as single objective problems. There are rare studies on feature subset selection as a multi-objective problem [107, 41]. The main reason is that it is very hard to obtain a set of uniformly distributed optimal solutions. The key challenge is to develop an efficient multi-objective approach to solving feature selection problems.

Single Feature Ranking for Feature Selection

Single feature ranking is a relaxed version of feature selection, which only requires the computation of the relative importance of the features and subsequently sorting them [32]. Feature selection can be accomplished by using only the few top-ranked features for classification. Many feature selection algorithms include single feature ranking as a principal or

auxiliary selection mechanism because of its simplicity, scalability, capability to avoid overfitting and good empirical success [32]. However, in most of the existing single feature ranking approaches, the ranking criteria, which measures the relative importance of each feature, is defined for individual features independently of the context of others (the presence or absence of some other features). In many real-word classification problems, such as a problem includes interaction features, a feature alone may not show any relevance to the target class, but it may become relevant in the presence of some other features. Therefore, a single feature ranking criterion, which takes into account the context of other features to improve the feature ranking performance needs to be investigated.

Feature Construction

In classification problems, the feature construction process is using original features to create new high-level features to improve the classification accuracy[115]. The constructed features are in fact mathematical expressions of the original features. If the feature construction function (mathematical expression) is known, feature construction is to select the best group of features that are used in the function to optimise the quality of the constructed features. However, a proper feature construction function is usually very difficult to design when domain knowledge is unknown. In such a situation, a feature construction approach has to simultaneously evolve a feature construction function and select the best group of features that are used to create new features. Therefore, a good feature construction approach should have a feature selection ability to search for the best combination of features for the feature construction function. For a certain dataset, feature selection methods search for the optimal feature subset in the feature space and feature construction approaches also search for the best combination to construct new features in this feature space. PSO is a powerful search technique for feature selection, but it has never been used for feature construction.

Fitness Evaluation

For both feature selection and construction, algorithms using different fitness evaluation functions will most likely obtain different results. A feature subset or constructed features could usually be optimal for one learning algorithm but not for all. For example, if information gain is used to evaluate a feature subset in a filter approach in the training process, the selected feature subset will be optimal for a decision tree algorithm. Therefore, information gain is the best fitness evaluation criterion for a decision tree classifier, but it is unknown whether information gain is also optimal for other classifiers. In wrapper approaches, it is obvious that if a classifier is applied in the training process, the selected feature subset or constructed features will be optimal for this classifier in unseen data, but it is unknown whether they are also optimal for other classifiers. If good evaluation functions can be established for commonly used classification algorithms, such as K-nearest neighbour (KNN), naive bayes (NB), decision tree (DT), it would be much more convenient for users.

If the best fitness evaluation function for a classifier is known, users can select the best fitness evaluation function for the desired classifier to obtain the best classification accuracy in unseen data and/or reduce the training time. However, not much work has been conducted on this topic.

1.3 Research Goals

The overall goal of this research is to investigate a new approach to using PSO for feature selection in classification problems. The specific research objectives of this work can be

itemised as follows.

1. Developing a new version of discrete PSO for feature selection. The goal is to explore the potential of discrete PSO for feature selection problems. To achieve this goal, it is necessary to investigate how a discrete PSO could search the feature subset space to find the best solution. Due to the limitations of the standard discrete (binary) PSO and the challenges of feature selection, this research goal leads to three objectives as follows.

   (a) Propose a new scheme to encode the solutions (particles) in discrete PSO for feature selection.

   According to the current encoding scheme in PSO for feature selection, the dimensionality of the search space is very high when the number of features is large and the encoded binary mask only shows whether a feature is selected and/or not. The new encoding scheme is expected to reduce the dimensionality of the search space or to show the relationship between features. The performance of the newly developed encoding scheme will be compared with that of the original encoding scheme.

   (b) Design a new topology for discrete PSO for feature selection and construction problems.

   There are many different available topology structures, which work well in continuous PSO. This research will start with comparing the performance of commonly used topologies in discrete PSO for feature selection and construction. Based on analysing the results and considering the characteristics of the feature search space, a new topology for discrete PSO for feature selection problems will be developed.

   (c) Investigate new position and velocity update equations and add search operators to optimise the performance of discrete PSO.

   In order to overcome the limitations of the standard discrete PSO, this objective is to investigate new velocity and position update equations in discrete PSO or/and add new search operators. The newly developed equations and search operators are expected to better present the core concept of the PSO algorithm than the original binary update equations.

2. Investigate a single objective feature selection algorithm. The goal is to develop a single objective feature selection algorithm based on the proposed discrete PSO. The subset of features resulting from this algorithm are expected to be sufficient to reflect different aspects of the dataset. The performance of discrete PSO will be compared with that of canonical standard PSO and other relevant benchmark algorithms, such as SFS and SBS. This research goal leads to the four objectives as follows.

   (a) Investigate a strategy to improve the computation efficiency (caching strategy) of discrete PSO for feature selection.

   One of the main challenges in the feature selection problem is that the computational cost is high, especially for wrapper approaches. This research is to develop a strategy to improve the computation efficiency of discrete PSO for feature selection without compromising its performance.

   (b) Use discrete PSO to select a subset of features to maximise the classification accuracy irrespective of the number of selected features.

   The fitness function is to maximise the classification accuracy irrespective of the number of selected features. It will be investigated whether discrete PSO can

select a good feature subset with which a learning algorithm could achieve better performance than with all features and feature subsets obtained by other existing approaches.

(c) Use discrete PSO to select a feature subset with a certain number of features and maximise the classification accuracy.

Further, discrete PSO will be to select a certain number of features with a fitness function of maximising the classification accuracy. As there are many combinations for a feature subset with a particular number of features, discrete PSO is expected to search for the best combination (subset) of features from the possible combinations.

(d) Use discrete PSO for single feature ranking for feature selection.

In the objective 2(c), when the number of features is one, the best solution should be the first top-ranked feature in single feature ranking. Therefore, this objective is to develop a feature selection algorithm based on single feature ranking and discrete PSO. Due to feature interaction problems (epistasis), the challenge is to develop a criterion to measure the relative importance of each feature in the presence or absence of some other features.

3. Investigate multi-objective discrete PSO for feature selection with the objectives of maximising the classification accuracy and minimising the number of features.

As the objectives of feature selection involve both the classification accuracy and the number of selected features, the goal is to use discrete PSO to solve the feature selection task as a multi-objective problem. The challenge is to develop a multi-objective discrete PSO, which can find a set of uniformly distributed solutions. The performance of the proposed multi-objective discrete PSO will be compared with that of other multi-objective algorithms.

4. Utilise discrete PSO for feature construction.

This objective aims to discover if discrete PSO can be an appropriate approach to constructing new high-level features using original features. The new constructed features are expected to transform the original input space in a way that can enhance the learning ability of a learning algorithm. The key challenge is to formulate a feature construction function to guide the discrete PSO search. By finding such a solution, PSO is expected to find the best combinations of features for this function to create high-level features and the constructed features could improve the classification performance of the learning algorithm. This research goal can be achieved in two following ways.

(a) Design fixed functions to construct new high-level features and use discrete PSO to select the original features from feature space to construct new high-level features.

Depending on the number of features, instances and classes, it may be needed to investigate different feature construction functions that will be used in PSO to construct high-level features. In order to optimise the representational power of the constructed features, it is needed to discover the relationship between the feature construction function, the number of original features needed, and the classification performance.

(b) Use discrete PSO to automatically evolve a function to construct new high-level features and also select original features.

A good feature construction function should be designed based on domain knowledge, but this is usually incomplete or unavailable. In order to overcome this

situation, the objective here is to use discrete PSO to evolve the feature construction function and select the original features simultaneously. The key challenge is how discrete PSO could evolve the feature construction function. New search operators will be designed for discrete PSO, which could evolve mathematical and/or logical functions.

5. Predict the fitness evaluation function with which a feature selection algorithm could obtain the best feature subset for a given learning algorithm.

   This research goal is optional depending on the time available and the related research will start after the above four goals are achieved. The goal is to investigate a good fitness evaluation function used in the training process of a feature selection approach, which will obtain a good feature subset for a given learning algorithm. This research will focus on commonly used learning algorithms, such as K-nearest neighbour (KNN), naive bayes (NB), decision tree (DT), support vector machine (SVM) and learning classifier system (LCS). In filter approaches, entropy, information gain, correlation coefficient and other commonly used fitness evaluation functions will be investigated to determine which of them is optimal for a given learning algorithm. In wrapper approaches, simple learning algorithms will be investigated to determine whether they can be used in the training process to obtain a feature subset, which is optimal for the desired learning algorithm. It is necessary to discover the relationship between the fitness evaluation function and the classification performance of the learning algorithms.

1.4 Organisation of Proposal

The remainder of this proposal is organised as follows. Chapter 2 presents essential background and related work in PSO, feature selection and feature construction. The review covers evolutionary computation and machine learning, particularly PSO, feature selection and construction. It also discusses open questions and current challenges that form the motivations of this work. Chapter 3 presents preliminary work conducted in single feature ranking and binary PSO based feature subset ranking for feature selection. Chapter 4 provides a detailed research plan and a timeline of the tasks in this project.

Chapter 2

Literature Review

This chapter reviews the literature on evolutionary computation techniques as main approaches to solving feature selection problems. This chapter covers essential background and basic concepts of evolutionary computation and machine learning, particularly PSO, feature selection and construction. It reviews typical related work in feature selection and construction problems using conventional methods and evolutionary computation techniques.

2.1 Machine Learning for Classification

Machine learning is a major research area of artificial intelligence. It is concerned with the design, analysis, implementation, and application of programs that are capable of learning in their environment [66, 69, 5]. A machine learning system is expected to be able to automatically improve its performance at certain task as it gains more experience [69].

Mitchell [69] provided a widely quoted definition of machine learning:

> "computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**" .

Machine learning algorithms use a feedback mechanism to change their behaviour (learn). Depending on the type of feedback, machine learning algorithms can be classified into three main categories: supervised, unsupervised and reinforcement learning [93].

- In supervised learning, the learner is learning with labeled class examples or instances. The desired outputs for a problem are known in advance, and the goal is to learn a function that maps inputs to desired outputs. Classification is a typical form of supervised learning.
- In unsupervised learning, the learner is learning without labeled class examples, which means there is no correct answers for the learner to explicitly learn from. It attempts to find inherent patterns that can then be used to determine clusters for given instances.
- In reinforcement learning, desired outputs are not directly provided. Every action of the learner has some impact to the environment, and the environment provides feedback on the goodness of its action in the form of rewards and punishments. The learner learns based on the rewards and punishments it receives from the environment.

Classification is one of the major tasks in machine learning, which refers to an algorithmic procedure for assigning a given piece of input data into one of the given categories [10].

During the classification process, a learnt classifier is needed, which takes the values of the features or attributes of an object as input and the predefined class labels for the object as output. The classifier is learnt by a learning algorithm, also called a classifier inducer, which uses a set of examples to learn a classifier that is expected to correctly predict the class label of unseen (future) instances [10].

A typical classification example is the email spam-catching system, which is important and necessary in real-world applications. Given a set of emails marked as "spam" and "non-spam" , the learner will learn the characteristics (features) of the spam emails and then the learnt classifier is able to process future email messages to mark them as "spam" and "non-spam".

2.1.1 Training and Testing

Common to classification problems are the processes of training and testing. The process by which a learning algorithm (classifier inducer) uses observations to learn a new classifier is called the *training* process and the process by which the learning algorithm is tested is called the *testing* process [69]. During the training process, the classifier is learnt from a collection of observations from the problem domain called instances, which is called the training set. The algorithm learns important knowledge or rules in the training set by building models and adjusting the corresponding parameters. The performance of the algorithm is then evaluated on the test set, which is a collection of instances in the same problem domain, except that these are not used and remain unseen in the training process.

The learning ability of algorithms is usually examined by applying them to a set of benchmark problems. Benchmark problems are usually chosen from datasets that are publicly accessible to researchers (e.g. UCI Machine Learning Repository [26]) so that results can be verified and the performance can be checked. A dataset usually has a training set and a test set. In such problems, the learning algorithm then becomes two-fold: to discover or learn different kinds of knowledge or rules from the training set, and apply these rules to the test set to measure the learnt model. However, many benchmark problems do not have a specific test set or some of them only have a small number of available instances in the dataset. To evaluate the performance of a classifier on these problems, it is necessary to use some resampling methods, such as n-fold cross-validation [72].

In n-fold cross-validation, the dataset is randomly partitioned into n folds (partitions) and the folds are near-equal size. In stratified n-fold cross-validation, the folds are selected so that the proportion of instances from different classes, remains the same in all folds. Subsequently, a single fold of the n folds is retained as the test set for testing the model, and the remaining $n-1$ folds are used as training set. The cross-validation process is then repeated n times, with each of the n folds used exactly once as the test set. The n results from the n experiments can then be averaged to produce a single estimate of classification performance. The advantage of such a method is that all observations are used for both training and testing, and each observation is used for testing exactly once. Generally, a larger n will produce an estimate with smaller bias because of the higher proportion of observations in the training set, but potentially higher variance (on top of being computationally expensive) [72]. Leave-one-out cross-validation (LOOCV) is an extreme case of n-fold cross-validation, which uses a single observation from the dataset as the test set, and the remaining observations as the training set. This is the same with a n-fold cross-validation with n being equal to the number of observations in the dataset. Note that n-fold cross-validation is usually used when the number of examples in the entire dataset is small.

2.1.2 Classifiers

Many different learning algorithms (classifiers) have been proposed in machine learning. Most commonly used classifiers, which will be used in this research, are reviewed in this section, such as K-nearest neighbour (KNN), bayesian classifiers, support vector machines (SVMs), and decision tree classifiers (DT).

2.1.2.1 K-nearest Neighbour Classifier (KNN)

KNN is a type of instance-based learning approach to classification. When using KNN for classification, it calculates the distances between the test instance and every instances in the training set, then assigns the test instance to the class that is the most common in the k nearest neighbours, where k is a positive integer, typically small. Euclidean distance, manhattean distance and other distance measures can be used to measure the distance between objects in KNN [5].

In KNN, there is no explicit training phase or it is very minimal. In other words, KNN does not use the training data points to do any generalisation. The training data in KNN is needed during the testing process, which is in contrast to other techniques like SVM where the training set and all non support vectors (hyperplanes) can be safely discarded. KNN is a simple learning algorithm, but works well in practice. KNN does not make any assumptions on the underlying data distribution. This is because in real-world applications, most of the datasets do not obey the typical theoretical assumptions (e.g. gaussian mixtures, linearly separable) which are needed in certain classifiers[5]. However, for a large training set, KNN requires large memory and is very time-consuming to make a decision [25].

2.1.2.2 Decision Tree Classifiers (DT)

DT learning is an approach to approximating discrete-valued functions [69]. DT classifiers partition the input training data into smaller subsets by producing optimal rules or decisions, also called nodes, which maximise the information gained [69]. The learnt decision tree is a hierarchy of nodes, where leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Instances are classified by sorting them down the tree from the root to certain label nodes. For a given instance, the classification process starts at the root node by testing the value of the feature at the root node and then moves to one of the child nodes. Then the process is repeated for the subtree rooted at the new node.

The main problem in learning a decision tree is to determine which feature should be tested at each node of the tree. Most decision tree learning algorithms employ a top-down greedy search through the space of possible decision trees, such as the iterative dichotomiser 3 algorithm (ID3) [90], the C4.5 algorithm [89] and its Java version, the J48 algorithm [109]. These algorithms use an entropy function to measure the homogeneity of examples and choose the best node at each stage. The developed model can be expressed as a set of 'if-then' decision rules to improve human readability. Meanwhile, decision trees are easy to modify. A disadvantage of decision trees is that they are weak in separating non-rectangular areas in the input space [89], which creates two-way or multi-way feature interactions (See the second challenge discussed in feature selection and construction in the first paragraph in Section 1.2).

2.1.2.3 Support Vector Machines (SVMs)

SVMs are supervised learning methods based on the statistical learning theory. SVMs require that each data instance is represented as a vector of real numbers. The main idea of SVMs is to map the input data to high dimensional space, where SVMs could construct a hyperplane or a set of hyperplanes which are used to create a decision boundaries for classification [35]. SVMs aim to maximise the distances between the hyperplanes and the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalisation error of the classifier [35]. Instances are classified based on what side of these hyperplanes they fall on.

A particular advantage of SVMs over other learning algorithms is that they can be analysed theoretically using concepts from the computational learning theory, and at the same time can achieve good performance when applied to real-world problems [34]. From a practical point of view, the most serious disadvantage of SVMs is the high algorithmic complexity and extensive memory requirements in large-scale tasks [103].

2.1.2.4 Bayesian Classifiers

Bayesian classifiers are probabilistic methods for classification. Their assumptions are that the behaviour of data (input-output relationships) can be captured in probability distributions and features or attributes of the problem are statistically independent [69]. A Bayesian algorithm stores a simple probabilistic summary for each class and this summary contains the conditional probability of each feature or attribute value given the class, as well as the probability (or base rate) of the class [58].

Naive bayes (NB) classifiers are the most common and straightforward bayesian classifiers. It has been shown that NB classifiers are quite competitive with other classifiers, such as DT and neural networks (NN) [67]. NB classifiers make significant use of the assumption that all input features are conditionally independent. This assumption can not be applied to many real-world problems where there are interdependency between the input features, which causes two-way or multi-way feature interactions (See the second challenge in feature selection and construction discussed in the first paragraph in Section 1.2).

2.2 Feature Selection

Feature selection, also known as variable selection or attribute selection, is an important data preprocessing technique. Feature selection is defined by many researchers according to different criteria, but most of them are similar in intuition and/or content [19]. The following lists those that are conceptually different and cover a range of definitions.

- Improving predictive accuracy: feature selection is to choose a subset of features for improving the predictive performance or reducing the complexity of the model without significantly decreasing prediction accuracy of the classifier built using only the selected features [51].

- Approximating original class distribution: feature selection is to select a subset of features such that the resulting class distribution, given only the values of the selected features, is as close as possible to the original class distribution given all features [51].

- Classical: feature selection is to select m features from n original features, $m < n$, such that the value of a criterion function is optimised over all subsets of size m [76].

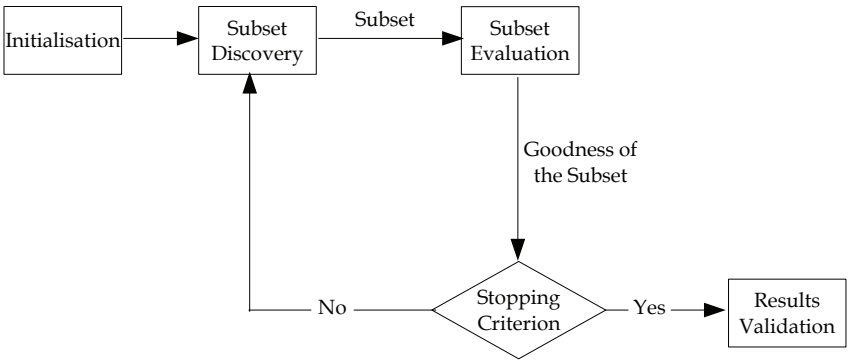


Figure 2.1: General feature selection process [19].

- Idealised: feature selection is to find the minimally sized feature subset that is necessary and sufficient to describe the target concept [49].

Note that the second definition emphasises the class distribution of the training set, whereas the third definition emphasises on selecting the best combination of m features based on a certain criterion.

Overall, feature selection is the process of finding a minimal subset of features that is necessary and sufficient to solve a classification problem. Naturally, the optimal feature subset is the smallest subset that can obtain the highest classification performance, which makes feature selection a multi-objective problem [83, 107].

Feature selection leads to dimensionality reduction by eliminating irrelevant and redundant features from the dataset, which in turn improves the classification performance and makes the learning and execution processes faster. Models constructed using a smaller number of features are also easier to interpret.

The search space of a feature selection problem has 2^n points where n is the number of available features. A feature selection algorithm explores the search space of different feature combinations to find the best feature subset. As the search space grows exponentially along with the number of features, it is impractical to search the whole space exhaustively [50] in most situations.

2.2.1 General Feature Selection Process

Generally, there are five basic steps in a typical feature selection approach [19] (see Figure 2.1).

1. A feature selection algorithm starts with a initialisation procedure.

   The initalisation procedure is the first step of a feature selection algorithm and it is based on all the original features in the problem. For example, in a PSO based feature selection algorithm, the dimensionality of the search space is usually set as the number of all features in this procedure.

2. A discovery procedure to generate candidate subsets.

   It is a search procedure [57], which can start with no features, all features, or a random subset of features. Many search techniques, including conventional methods and evolutionary techniques, are applied in this generation step to search for the best subset of features.

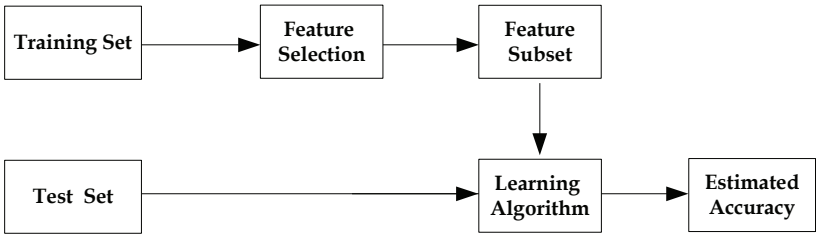


Figure 2.2: A filter feature selection approach in which the features are filtered independently the learning algorithm.

3. An evaluation function to measure the subset.

   Feature subsets produced by the generation procedure will be measured by a fitness evaluation function to determine their goodness. Fitness evaluation function plays an important role in a feature selection approach, because it helps guide the algorithm to search for the optimal feature subset.

4. Based on given criteria to decide when to stop.

   Stopping criteria can be based on the generation procedure and evaluation function. Criteria based on the generation procedure can be whether a predefined number of features are selected and whether a predetermined maximum number of iterations have been reached. Evaluation based criteria include whether addition or deletion of any feature does not produce a better subset and whether the optimal subset according to certain evaluation functions has been obtained. The loop continues until the stopping criterion is satisfied.

5. A validation procedure to check whether the subset is valid.

   The validation procedure is not part of the feature selection process itself, but a feature selection approach must be validated. The selected feature subset will be validated on the test set. The results will be compared with previously established results or the results of predefined benchmark techniques.

2.2.2 Filter vs Wrapper Approaches

For a feature selection problem, the optimal feature subset is always relative to a certain fitness evaluation function. The optimal feature subset chosen using one fitness evaluation function is usually not the same feature subset chosen using another fitness evaluation function. Based on whether a learning algorithm is used in the fitness evaluation function or not, the existing feature selection methods can be broadly classified into two categories: filter approaches and wrapper approaches [57]. A filter feature selection approach is independent of a learning algorithm whereas wrappers use a learning algorithm in the fitness evaluation function.

Figure 2.2 shows the diagram of a feature selection system taking a filter approach. In filter approaches, the search process is independent of learning algorithms. The goodness of feature subsets are evaluated based on a certain criterion like distance measure, information measure and consistency measure [19]. Filter approaches are argued to be computationally less expensive and more general than wrappers [111, 40], but filter approaches totally ignore the performance of the selected feature subset on the learning algorithm, which usually

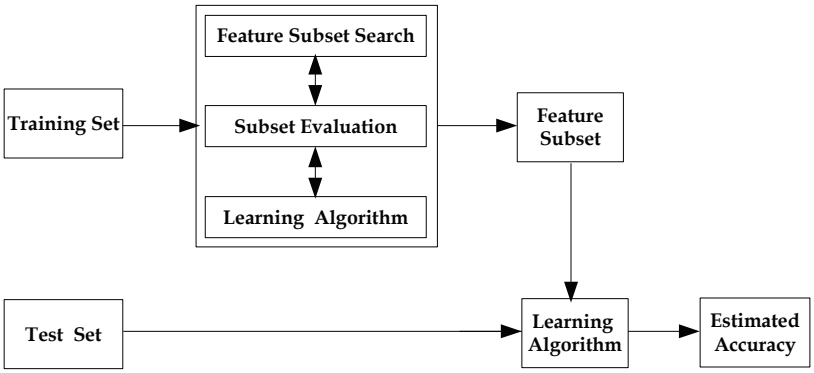


Figure 2.3: A wrapper feature selection approach which exists as a wrapper around the learning algorithm.

leads to lower performance than wrapper approaches on a particular learning algorithm [40].

Figure 2.3 shows the diagram of a wrapper feature selection approach. In a wrapper model, the feature selection algorithm exists as a wrapper around a learning algorithm and the learning algorithm is used as a "black box" by the feature selection algorithm [40]. The performance of the learning algorithm is used to guide the search by evaluating the goodness of feature subsets. Compared with filter approaches, wrappers usually produce better performance because of the interaction between the learning algorithm and the training set in the subset searching process [50]. However, wrapper feature selection approaches are usually computationally more expensive than filters because each evaluation of a candidate solution needs a learning algorithm to be trained and tested [38].

2.2.3 Classical Methods for Feature Selection

2.2.3.1 Filter Feature Selection Approaches

A filter feature selection approach searches for the optimal feature subset from the search space based on a certain evaluation criterion, which is independent of the learning algorithms. There are many filter feature selection algorithms that have been proposed and typical algorithms are reviewed in this section.

Many filters have been proposed based on different criteria, including distance measures [56], dependency measures [112], consistency measures [113], and information measures [24]. Besides the evaluation criterion, how to search for the best feature subset is another important factor in feature selection methods. The FOCUS algorithm is a classical filter feature selection algorithm, which was originally defined for noise-free Boolean domains [4]. It starts with an empty feature subset and exhaustively examines all subsets of features and then selects the minimal subset of features that is sufficient to determine the class label for all instances in the training set. This preference for a small subset of features is referred to as the MIN-FEATURES bias. This bias may lead to poor generalisation performance of the learning algorithm with the selected features [50]. Meanwhile, the FOCUS algorithm performs an exhaustive search to determine the best feature subset, which is computationally expensive.

The Relief algorithm is another popular filter feature selection method that assigns a relevance weight to each feature [48]. The weight is intended to denote the relevance of the feature to the target concept. Relief samples instances randomly from the training set

and updates the relevance values based on the difference between the selected instance and the two nearest instances of the same and opposite class (the "near-hit" and "near-miss"). However, the Relief algorithm does not deal with redundant features, because it attempts to find all relevant features regardless of the redundancy between them [52], which is the third challenge in feature selection and construction as discussed in Section 1.2.

Decision trees use only relevant features that are required to completely classify the training set and remove all other features. Cardie [11] proposed a filter based feature selection algorithm that used a decision tree algorithm to select a subset of features for a nearest neighbourhood algorithm. Experiments showed that the subset generated by a decision tree helped the nearest neighbour algorithm to reduce its prediction error. However, the features that are good (or not good) for the decision tree are not necessary useful (or not useful) for the nearest neighbour algorithm, which will lead to poor feature selection performance.

Yu and Liu [112] claimed that feature relevance alone was insufficient for efficient feature selection of high-dimensional data. They proposed a feature selection algorithm that took both relevance and redundancy into account. The algorithm, however, is limited to problems that only have discrete features.

Recently, different evolutionary computation techniques have been applied in filter feature selection approaches, such as PSO [106], GAs [13], GP [79], and ant colony optimisation (ACO) [39]. Typical approaches will be reviewed in Section 2.4.

2.2.3.2 Wrapper Feature Selection Approaches

Generally, wrapper feature selection approaches are usually computationally more expensive than filters because each evaluation involves training and testing the learning algorithm [88]. As the search space of a feature selection problem has 2^n possible points where n is the number of available features in the dataset, it is impossible to search the whole search space exhaustively in most cases. Therefore, most of existing wrappers employ greedy or stochastic search strategies [28].

Sequential forward selection (SFS) [108] and sequential backward selection (SBS) [64] are two commonly used wrapper feature selection approaches. Both of them use a greedy hill-climbing search strategy to search for the optimal feature subset. SFS starts with an empty set of features and iteratively adds one feature at one time until no improvement in classification accuracy can be achieved. By contrast, SBS sequentially removes features from a full candidate feature subset until the removal of further features does not increase the classification accuracy. Both SFS and SBS suffer from the so-called nesting effect, which means once a feature is selected (discarded) it cannot be discarded (selected) later. Therefore, both SFS and SBS are easily trapped in local optima [50]. In addition, both SFS and SBS require long computational time when the the number of features is large [50].

In order to avoid nesting effect, Stearns [97] proposed a "plus-l-take away-r" method in which SFS was applied l times forward and then SBS was applied for r back tracking steps. However, determining the best values of (l, r) was a challenge. In order to solve this problem, Pudil et al. [87] proposed floating selection methods, sequential backward floating selection (SBFS) and sequential forward floating selection (SFFS) to automatically determine the values of (l, r). In addition, the values of (l, r) in SBFS and SFFS that denotes the number of forward and backtracking steps are dynamically controlled instead of being fixed in the "plus-l-take away-r" method. Although the floating methods are regarded to be at least as good as the best sequential method, they are still likely to become trapped in a local optimal solution even the criterion function is monotonic and the scale of the problem is quite small [114].

Based on the best-first algorithm and SFFS, Gutlein et al. [31] proposed a linear forward

selection (LFS) in which the number of features considered in each step was restricted. Because of the small number of evaluations in each step, LFS improves the computational efficiency of sequential forward methods while maintaining comparable accuracy of the selected feature subset. However, LSF starts with ranking all the individual features without considering the presence or absence of some other features, which in turn limits the performance of the LSF algorithm in problems where there are interactions between features.

Recently, evolutionary computation techniques have been applied in wrapper feature selection models, such as PSO [60], GAs [83], GP [74], and ACO [95]. Typical methods will be reviewed in Section 2.4.

2.2.4 Single Feature Ranking

Single feature ranking is a relaxed version of feature selection [32]. Single feature ranking is computationally cheap, because it only requires the computation of the relative importance of the features and subsequently sorting them.

In single feature ranking, a score denotes the relative importance of a single feature, which is measured by a predefined criterion. All the features are ranked according to the score and then feature selection can be accomplished by selecting a small number of top-ranked features. Normally, users specify the number of top-ranked features they need according to their requirements. There are also analytical methods to determine the best number of features [98].

Most single feature ranking methods fall into the filter approach category and not much work has been conducted on wrapper based single feature ranking [80]. Many measures have been proposed to evaluate the relative importance of each feature in single feature ranking approaches, including information gain, gain ratio, mutual information and so forth [59].

Most of existing single feature ranking approaches only measure the goodness of a single feature, not taking into account the interaction between groups of features [32, 92]. Neshatian et al. [82] proposed a feature ranking approach in which the importance of each feature was evaluated in a group of features. In the proposed method, GP was applied to evaluate the importance of a set of features by incorporating them into the construction of GP programs. Each feature was scored based on the frequency of appearance in a collection of GP programs and the fitness of those programs. Experiments showed that with a few of top-ranked features a variety of different classifiers could achieve better classification performance than with all features. However, the proposed method has not been compared with other feature ranking approaches, and whether the combination of the top-ranked features is redundant or not has not been investigated either.

In many classification problems, groups of several features working together are relevant but not the individual features alone. If any feature in the group is absent, the rest features become irrelevant to the problem. This phenomenon is known in evolution theory as epistasis [68]. If there are epistatic features included in the problem, most of existing approaches might not obtain good feature ranking performance (e.g. the combination of the top-ranked features is a redundant subset). An effective single feature evaluation criterion which considers epistasis phenomenon can potentially improve the performance of single feature ranking, but this needs to be further investigated.

2.2.5 Feature Construction

Many classification algorithms, particularly those based on symbolic learning (e.g. decision trees), cannot achieve adequate classification performance when facing difficult real-world

problems [55]. As one of the feature transformation methods, feature construction could significantly improve the performance of the classification algorithms by constructing new features, which could improve the quality of representation of the problem and reduce its complexity [101]. Based on whether a learning algorithm is included in the feature construction process or not, feature construction approaches (like feature selection methods) can be roughly divided into two categories, which are wrapper approaches and filter approaches [61].

In wrapper approaches, feature construction and induction are integrated into one single algorithm, where new features are constructed within the induction process of the learning algorithm. Murthy et al. [75] proposed a system which constructed new features by linearly combining the original features in the process of inducting an decision tree. Zheng [116] studied the effects on decision tree learning of constructing different types of attributes and proposed a single tree learning algorithm to reduce effects of factors, such as new attribute search strategies, evaluation functions, and stopping criteria. The proposed approach could outperform the standard decision tree approaches. However, wrapper feature construction approaches have a disadvantage of losing the generality and increasing the processing time in general [84].

In filter approaches, the process of feature construction is a separate, independent preprocessing stage and the new features are constructed before the classification algorithm is applied to build the model. Recently, more feature construction methods fall into this category than wrapper category because of computational efficiency and generality of filter approaches [84, 73, 96]. Due to the capability of GP in building programs and expression, it has been used as an efficient technique for feature construction. Otero et al. [84] used GP to construct new features with information gain ratio as the fitness function. C4.5 classifier was applied for classification and its performance was improved by using the newly constructed features together with the original features. This improvement may be because both C4.5 classifier and GP use information gain ratio as fitness function and it is unknown whether a different classifier still could benefit as much as C4.5. Muharram and Smith [73] used the same method in [84] for feature construction and evaluated the performance of the newly constructed features using four different classifiers. Adding the newly constructed features into the feature space could not improve the performance of four classifiers in two of the five datasets. Most of the feature construction approaches increase the number of features because of adding newly constructed feature(s) into the feature space, which in turn increases the dimensionality of the problem [62]. Smith and Bull [96] proposed an approach using GP to construct features and then using a GA for feature selection to reduce the number of features. Krawiec [54] developed a GP based feature construction approach to constructing a fixed number of new features to replace the original features. Neshatian et al. [81] used GP to construct multiple features with which the classifier could achieve higher classification accuracy than using constructed features together with original features.

As a powerful evolutionary computation technique, PSO is easy to implement, has a few parameters and can converge fast [22]. However, the potential of PSO in feature construction has not been investigated.

2.3 Particle Swarm Optimisation (PSO)

2.3.1 Evolutionary Computation

Evolutionary computation (EC) is an area of artificial intelligence that covers the majority of the techniques inspired by principles of biological evolution. EC techniques have been applied successfully to solve a variety of real-world problems [22]. In general, EC consists

of evolutionary algorithms (EAs), swarm intelligence (SI) and other techniques.

2.3.1.1 Evolutionary Algorithms

Evolutionary Algorithms (EAs) is a subset of evolutionary computation, which are population based metaheuristic optimisation algorithms. An EA uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection. In EAs, each candidate solution of the optimisation problem is represented as an individual in the population. The fitness function determines the goodness of each individual. Evolution of the population then takes place after the repeated application of the above operators. Two important EAs, genetic algorithms and genetic programming, are reviewed as follow.

Genetic Algorithms. Genetic algorithms (GAs) are evolutionary algorithms that use the principle of natural selection [29]. In GAs, candidate solutions of the problem are encoded as a population of chromosomes and the population is evolved to search for the optimal solution by applying genetic operators. Compared to analytical optimisation methods like gradient based optimisation, GAs are less likely to be trapped in local optima. They, however, tend to be computationally expensive.

Genetic Programming. Genetic programming (GP) is a sophisticated EA which is used to evolve a computer program that performs a user-defined task [53]. In GP, each individual is a computer program and a population of computer programs is optimised according to a fitness landscape determined by a program's ability to perform a given task. GP has been successful as a technique for getting computers to automatically solve problems without having to tell them explicitly how.

2.3.1.2 Swarm Intelligence

Swarm Intelligence (SI) algorithms are inspired by the collective intelligence of social insects. A swarm is a population of interacting individuals that is able to optimise global objectives through collaborative search of the space and the intelligence lies in the networks of interactions among individuals, and between individuals and the environment. There is a general stochastic (or chaotic) tendency in a swarm for individuals to move toward a centre of mass in the population on critical dimensions, resulting in convergence on an optimum [46]. Two main techniques in SI are particle swarm optimisation and ant colony optimisation.

Particle Swarm Optimisation. Particle swarm optimisation (PSO) is a SI algorithm inspired by social behaviour of birds flocking or fish schooling [43]. In PSO, each candidate solution of the problem is encoded as a particle moving in the search space and each particle can remember its best experience. The whole swarm searches for the optimal solution by updating the position of each particle based on the best experience of its own and its neighbouring particles [46]. PSO is a simple but powerful search technique, which has been applied successfully in a variety range of areas [22].

Ant Colony Optimisation. Ant colony optimisation (ACO) takes inspiration from the behaviour of real ants seeking the shortest path between their colony and a source of food [20]. Candidate solutions of the problem are represented as ants in the population. These ants deposit pheromone on the ground in order to mark their favourable path that should be followed by other members of the colony. The best solution is the "path" that has the most pheromone. This mechanism allows the algorithms explicitly use the elements of previous solutions, which is the characteristic of ACO algorithms.

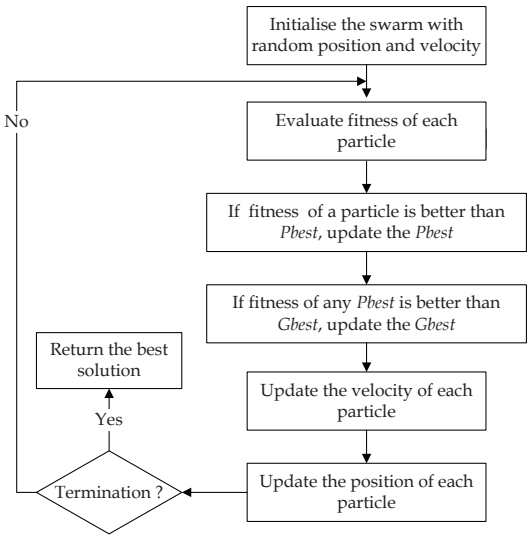


Figure 2.4: The flowchart of PSO

2.3.2 Standard Particle Swarm Optimisation (PSO)

PSO is an evolutionary computation technique proposed by Kennedy and Eberhart in 1995 [43]. In PSO, each solution can be represented as a particle in the search space. A vector $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ presents the position of particle i, where D is the dimensionality of the search space. The velocity of particle i is represented as $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The best previous position of each particle is recorded as the personal best called *Pbest* and the best position obtained thus far is called *Gbest*. The swarm is initialised with a population of random solutions and searches for the best solution by updating the velocity and the position of each particle according to the following equations:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{2.1}$$

$$\begin{aligned} v_{id}^{t+1} &= w * v_{id}^{t} + c_1 * r_1 * (p_{id} - x_{id}^{t}) \\ &+ c_2 * r_2 * (p_{gd} - x_{id}^{t}) \end{aligned} \tag{2.2}$$

where t denotes iteration t in the search process. c_1 and c_2 are acceleration constants. r_1 and r_2 are random values uniformly distributed in [0, 1]. p_{id} presents the *Pbest* and p_{gd} stands for the *Gbest*. w is inertia weight, which can make a balance between the global search and the local search to improve the performance of PSO. The velocity v_{id}^{t} is limited by a predefined maximum velocity, v_{max} and $v_{id}^{t} \in [-v_{max}, v_{max}]$.

PSO was originall proposed in a continuous version. In order to distinguish with the discrete version PSO, the standard PSO in this work is also called continuous PSO (CPSO). Algorithm 1 and Figure 2.4 show the pseudo-code and the flowchart of continuous CPSO, in which each particle is assumed to take the entire population as its topological neighbours. First, each particle is initialised with a random velocity and a random position in a D-dimensional search space. Second, the fitness of each particle is evaluated by the predefined fitness function, and then based on *Pbest* and *Gbest*, the velocity and the position of each particle are updated according to Equation 2.2 and 2.1 to search the possible best solution. During the search process, if the fitness of the particle is better than that of *Pbest*,

Algorithm 1: CPSO

Input : w: inertia weight; c_1, c_2: acceleration constants
v_{max}: maximum velocity; D: dimension of search space
P: the population size
T: maximum iterations

Output: *Gbest*
best fitness value

```
1  begin
2      randomly initialise the position and velocity of each particle;
3      while T or the stopping criterion is not met do
4          evaluate fitness of each particle;
5          for p=1 to P do
6              if fitness of x_p is better than that of Pbest_p then
7                  Pbest_p = x_p ;                  // Update the Pbest of particle p
8              end
9              if fitness of Pbest_p is better than that of Gbest then
10                 Gbest = Pbest_p ;                // Update the Gbest of particle p
11             end
12         end
13         for p=1 to P do
14             for d=1 to D do
15                 update the velocity of particle p according to Equation 2.2;
16                 update the position of particle p according to Equation 2.1;
17             end
18         end
19     end
20     return Gbest and its fitness value;
21 end
```

then its position will be saved to replace the *Pbest*. If the fitness of any *Pbest* is better than *Gbest*, the *Gbest* will be replaced by *Pbest*. The algorithm is terminated until the predefined maximum number of iterations or another stopping criterion is met.

2.3.3 Binary Particle Swarm Optimisation (BPSO)

PSO was originally introduced as an optimisation technique for real-number search spaces. However, many optimisation problems occur in a space featuring discrete, qualitative distinctions between variables and between levels of variables. To extend the implementation of the original PSO, Kennedy and Eberhart [44] developed a binary particle swarm optimisation (BPSO) for discrete problems. The velocity in BPSO represents the probability of element in the particle taking value 1 or 0. Equation (2.2) is still applied to update the velocity while x_{id}, p_{id} and p_{gd} are integers of 1 or 0. A sigmoid function $s(v_{id})$ is introduced to transform v_{id} to the range of (0, 1). BPSO updates the position of each particle according to the following formulae:

$$x_{id} = \begin{cases} 1, \text{ if } rand() < s(v_{id}) \\ 0, \text{ otherwise} \end{cases} \tag{2.3}$$

where

$$s(v_{id}) = \frac{1}{1 + e^{-v_{id}}} \tag{2.4}$$

where $s(v_{id})$ is a sigmoid limiting transformation. $rand()$ is a random number selected from a uniform distribution in [0,1].

2.3.4 Recent Work on BPSO

Since PSO was proposed by Kennedy and Eberhart in 1995, it has been analysed and improved for many problems [46, 22]. Most of the improvements are proposed for CPSO, including changing parameters, update equations, topology and introducing ideas from other algorithm (See [46, 22]). Recently, researchers propose different modified versions of BPSO to improve its performance in solving discrete problems. Modified BPSO algorithms are reviewed in this section.

Huang and Dun [36] showed that both binary and continuous values for elements in the position vector of particles in PSO could be used simultaneously. Therefore, problems with both binary and continuous variables could be optimised by PSO in a single algorithm.

Khanesar et al. [47] discussed two main concerns about BPSO. The first one was the parameters in BPSO, which produced opposite effects in BPSO compared with those in CPSO [22]. For example, a large maximum velocity encourages exploration in CPSO while a small maximum velocity promotes exploration in BPSO. The second concern about BPSO was that the position of a particle was updated solely using the velocity while the position in CPSO was updated based on both velocity and current position. Based on the analysis on these two concerns, Khanesar et al. defined a new interpretation for the velocity in BPSO and proposed two variables to represent the velocity which denoted the probability of the position changing from 0 to 1 and from 1 to 0. The proposed BPSO outperformed the basic BPSO on certain benchmark functions.

Modiri and Kiasaleh [70] investigated the influence of *Pbest* and velocity values in BPSO. By initialising the velocity with the predefined maximum or minimum velocity, the performance of BPSO could be improved on certain benchmark functions. It was also shown that if the range of parameter values (e.g. number of iteration, maximum or minimum velocity) in BPSO was large, the effect of *Pbest* was intrinsically encompassed by that of *Gbest*. However, *Pbest* contains some useful information and totally ignoring it also may lead to poor performance in real-world problems.

Wang et al. [105] introduced two operators called mutation and dissipation, concepts of natural evolutionary theory into BPSO, and developed a mutation-dissipation based BPSO. Menhas et al. [65] proposed a hybrid binary PSO (HBPSO) by introducing crossover and mutation operators into BPSO to accelerate the convergence speed and ensure the diversity of the swarm. HBPSO outperformed the modified BPSO algorithm proposed by Wang et al. [105].

Sadri and Suen [94] introduced two concepts, birth and mortality rates into standard BPSO. Population size in the proposed BPSO was changed by the birth operator and the death operator during the search process. The proposed BPSO increased the exploration power of BPSO.

Pampara et al. [85] proposed another version of BPSO in which a trigonometric function was utilised to generate a bit string from continuous numbers. In the proposed algorithm, a large binary search space could be presented by a small continuous number. Therefore, the optimisation process could be performed in a short time.

Topology that defines how particles are connected to each other as an information sharing or exchanging mechanism, is one of the key elements that influence the performance of

PSO [45, 42, 22]. Many different types of topologies have been proposed and widely used in CPSO, such as ring topology, star topology, and wheel topology [22]. Research has shown the effects of topology on the performance of CPSO [22], but the influence of topology on BPSO has not been investigated.

2.4 Evolutionary Computation Techniques for Feature Selection

2.4.1 PSO for Feature Selection

2.4.1.1 Continuous PSO (CPSO) for Feature Selection

Generally, when a CPSO algorithm is applied for feature selection problems, the dimensionality of the search space is n, where n is the number of the available features in the dataset. Each particle in the swarm is formed by a vector of n real numbers. The position of particle i in dth dimension, x_{id} is usually in interval [0, 1]. In order to determine whether a feature will be selected or not, a threshold $0 < \theta < 1$ is needed to compare with the real numbers in the position vector. If $x_{id} > \theta$, then the corresponding feature d will be selected. Otherwise, feature d will be abandoned. Recently, CPSO has been applied in both filter and wrapper feature selection approaches.

Azevedo et al. [8] proposed a wrapper feature selection approach using CPSO and SVM for personal identification in keystroke dynamic system. This research also investigated the parameter settings in CPSO, including population size p, acceleration factor c_1, c_2 and the threshold θ. It was shown that CPSO with $p = 100$ particles, $c_1 = c_2 = 1.5$, and $\theta = 0.7$ performed best. Experiments showed that the proposed approach produces better performance than a GA with SVM model. However, the proposed algorithm obtained a relatively high false acceptance rate, which should be low in most identification systems.

Later, Lin et al. [60] proposed a wrapper feature selection approach (PSO+SVM) using CPSO and SVM. The difference from the method in [8] was that PSO+SVM could optimise the parameters in SVM and search for the best feature subset simultaneously. Mohemmed et al. [71] proposed a hybrid method (PSOAdaBoost), which incorporated PSO with an AdaBoost framework for face detection. The proposed PSOAdaBoost algorithm aimed to search for the best feature subset and determine the decision thresholds of AdaBoost simultaneously, which would also speed up the process of the training and increase the accuracy of weak classifiers in AdaBoost. Both PSO+SVM and PSOAdaBoost could optimise the feature subset and parameters in one process. More work like this could be conducted to improve the performance of feature selection using PSO with other learning algorithms in a wrapper model.

Feature selection was also considered a multi-objective problem [83, 107], which aimed to select the smallest subset that could obtain the highest classification performance. Although using PSO to feature selection as a single objective problem could obtain good feature selection results, it can only produce one subset of features. Feature selection as a multi-objective problem producing several good feature subsets can meet different requirements in real-world applications [107]. Therefore, Paoli et al. [86] developed a filter feature selection method for clustering hyperspectral images based on multi-objective PSO. It could solve clustering, feature selection and class number estimation simultaneously. In order to handle these three different issues, three different optimisation criteria were applied to guide the search process, which were log-likelihood function, Bhattacharyya statistical distance [86] between classes, and the minimum description length. Experiments showed that the proposed approach could obtain satisfactory classification accuracy while reduce the number of features needed. A good guess of class number was also achieved. However, the

computational time may reach several hours, depending on the image size.

Esseghir et al. [23] proposed a filter-wrapper feature selection model based on CPSO, which aimed to integrate the strengths of both filters and wrappers. The proposed filter-wrapper scheme encoded the position of each particle in CPSO with filter scores of features, which reflected feature-class dependency levels, and then CPSO was applied to adjust the scores to search for the best feature subset. The positive score meant the corresponding feature was selected, otherwise it was abandoned. The fitness of each particle was the classification accuracy achieved by a KNN classifier with the selected features. The results showed that the proposed method could achieve slightly better performance than BPSO based filter approach. However, the performance of the proposed approach has not been compared with that of a wrapper approach.

2.4.1.2 BPSO for Feature Selection

Generally, when using BPSO to solve feature selection problems [100, 110], the representation of a particle is a n-bit binary string, where n is the number of features and the dimensionality of the search space. The feature mask is in Boolean that "1" represents the feature will be selected and "0" otherwise. Many BPSO based filter and wrapper feature selection approaches have been proposed in recent years.

Chakraborty [14] compared the performance of BPSO with that of GA in a filter feature selection approach with fuzzy sets based fitness function. The results showed that BPSO performs better than GA in terms of classification accuracy.

Inertia weight can improve the performance of BPSO by properly balancing its local search and global search. Yang et al. [111] proposed two strategies to determine the inertia weight of BPSO. Experiments on a wrapper feature selection model suggested that the two proposed BPSOs outperformed other methods, including SFS, "plus-l-take away-r" method, SFFS, sequential GA and different hybrid GAs.

In order to avoid the particles converging at local optima, Yang et al. [110] proposed a strategy to renew the *Gbest* during the search process to keep the diversity of the population in BPSO. In the proposed algorithm, when *Gbest* was identical after three generations, a Boolean operator 'and(.)' would 'and' each bit of the *Pbest* of all particles in an attempt to create a new *Gbest*. Experimental results illustrated that the proposed method usually achieved higher classification accuracy with fewer features than GA and standard BPSO.

Chuang et al. [16] also developed a strategy for *Gbest* in BPSO for feature selection in which *Gbest* would be reset to zero if it maintained the same value after several iterations. Experiments with cancer-related human gene expression datasets showed that the proposed BPSO outperformed the algorithm proposed by Yang et al. [110] in most cases.

Wang et al. [106] proposed an improved BPSO by defining the velocity as the number of elements that should be changed. The performance of the improved BPSO was compared with that of GA in a filter feature selection model based on rough sets theories. Experimental results showed that the improved BPSO was computationally less expensive than GA in terms of both memory and running time. They also concluded that most of the running time was consumed by the computation of the rough sets, which was a drawback of using rough sets to solve the feature selection problems.

Unler and Murat [100] modified the standard BPSO by extending social learning to update the velocity of the particles. Meanwhile, an adaptive feature subset selection strategy was developed, where the features were selected not only according to the likelihood calculated by BPSO, but also according to their contribution to the subset of features already selected. The improved BPSO was applied to a wrapper feature selection model for binary classification problems. Experimental results indicated that the proposed BPSO method out-

performed the tabu search and scatter search algorithms.

Alba et al. [3] combined a geometric BPSO with a SVM algorithm for feature selection, where the current position, *Pbest* and *Gbest* of a particle were used as three parents in a three-parent mask-based crossover operator to create a new position for the particle instead of using the position update equation. Experiments on high dimensional microarray problems showed that the proposed algorithm could achieve slightly higher accuracy than GA with SVM in most cases. Meanwhile, experiments also showed that the initialisation of the BPSO had a great influence in the performance since it introduced an early subset of acceptable solutions in the evolution process.

Talbi et al. [99] proposed also a geometric BPSO and compare it with GA using SVM for the feature selection in high dimensional microarray data. They concluded that the performance of the proposed BPSO was superior to GA in terms of accuracy.

Liu et al. [63] proposed a multiple swarm BPSO (MSPSO) to search for the best feature subset and optimise the parameters of SVM. Experimental results showed that the proposed feature selection methods could achieve higher classification accuracy with a smaller subset of features than grid search, standard BPSO and GA. However, the proposed MSPSO was computationally more expensive than other three methods because of the large population size and complicated communication rules between different subswarms.

Huang and Dun [36] developed a wrapper feature selection method based on BPSO and SVM, which used BPSO to search for the best feature subset and CPSO to simultaneously optimise the parameters in the kernel function of SVM, respectively. Experiments showed that the proposed algorithm could determine the parameters, search for the optimal feature subset simultaneously and also achieve high classification accuracy.

2.4.2 Other Evolutionary Computation Techniques for Feature Selection

Besides PSO, many other evolutionary algorithms also have been applied in feature selection problems such as GAs, GP, and ACO.

2.4.2.1 Genetic Algorithms (GAs) for Feature Selection

Generally, in a GA based feature selection approach, each individual (chromosome) in the population represents a subset of features. For a n-dimensional feature search space, each chromosome is encoded by a n-bit binary string. The bit with value '1' indicates the feature is selected in the subset, and '0' otherwise. Crossover, mutation and reproduction operators are applied in the algorithm to search for the optimal subset of features [37]. GAs have been applied to both filter and wrapper models for feature selection as a single objective and also a multi-objective problem.

Before developing the feature selection method based on fuzzy sets and BPSO [14], Chakraborty [13] proposed a GA with fuzzy sets based fitness function to build a filter approach for feature selection. This method had the same fitness function as BPSO based method [14]. The GA based feature selection method was robust but the computational time was usually long. The proposed method adopted the computationally light fuzzy fitness function to shorten the running time. However, the performance of proposed method was worse than that of CPSO based feature selection method in [14] in terms of classification accuracy, number of selected features and computation time.

Banerjee et al. [9] proposed a filter feature selection method for microarray gene expression data based on NSGA-II and rough set. Since the data typically consisted of a large number of redundant features, an initial redundancy reduction was performed to enable faster convergence and also reduce the computational complexity.

Yuan et al. [113] proposed a two-phase feature selection approach using both filter and wrapper, which aimed to take advantages of both models. The proposed method started with a filter model to remove irrelevant features, and then a wrapper approach was applied to remove the redundant features. In the filter phase, GA was employed for feature selection with inconsistency criterion to evaluate the fitness of solutions. The wrapper phase started with a feedforward neural network whose input nodes were features in the optimal feature subset obtained in the first phase. The proposed approach intended to reduce the computation cost in the wrapper approach in the second phase by deleting irrelevant features in the first phase. However, because of feature interactions (epistasis), the proposed algorithm may remove the features in the first phase, which are elements in the best feature subset.

GAs are also applied to the wrapper model as multi-objective methods in feature selection problems. For example, Oliveira et al. [83] developed a modified wrapper model using a multi-objective GA based on the Pareto approach to generating the Pareto optimal front for handwritten digit recognition. Sensitivity analysis and neural networks were employed to evaluate the fitness. Experiments on the NIST database illustrated the effectiveness of the proposed strategy.

Waqas et al. [107] also developed a wrapper approach to feature selection using a multi-objective GA. In this method, a subset that was irrelevant with one class and might be relevant with another one was regarded as a non-dominated or pareto-optimal solution. ID3 was employed to evaluate the fitness of each individual. Experiments showed that selected subsets of features could achieve high classification accuracy.

These two multi-objective approaches are more powerful in applications than single objective approaches to feature selection. However, more benchmarks datasets should be applied to illustrate the performance and the generalisation of these two techniques.

2.4.2.2 Genetic Programming (GP) for Feature Selection

GP is an evolutionary computation technique inspired by biological evolution to find computer programs that perform a user-defined task. GP evolves computer programs, traditionally represented in memory as tree structures [53]. Basically, in a GP based feature selection method, there are a function set F and a terminal set T including the original features and randomly generated constants. Each tree for each individual (classifier) is initialised with a subset of features using F and T. The population evolves to search for the optimal feature subset using genetic operations iteratively. GP based approaches have been proposed in recent years, including both filter and wrapper models for feature selection.

Muni et al. [74] developed a wrapper feature selection model based on multi-tree GP (GPmtfs), which simultaneously searched for a good feature subset and designed a classifier using the selected features. In GPmtfs, each individual in a c-class problem had c trees and each tree was initialised with a random feature subset. In GPmtfs, two new crossover operations: homogeneous crossover and heterogeneous crossover were introduced to suit the feature selection process. Comparisons with other results available in the literature showed that this method produced consistently better results.

Based on the two crossover operations introduced by Muni et al. [74], Purohit et al. [88] introduced another crossover operator to GP to reduce its randomness when used in a feature selection model. The crossover operator intended to select a subtree from the first parent and find its best place in the second parent. GP with multi-trees was used to design classifiers with feature selection for a multi-class classification problem. Experiments showed that proposed GP performed better than GPmtfs [74].

Ramirez and Puiggros [91] applied multi-tree GP to solve a multi-class problem, which was to classify the instantaneous cognitive state of a person. The performance measure,

the fitness function and initialisation of the classifiers were similar with [88]. Experiments showed the proposed method could accurately classify different cognitive states.

Chien and Yang [15] proposed a feature selection algorithm based on GP and rough sets. Rough membership was used to transform nominal data into numerical values. After transformation, new features and training sets were produced, and then GP was applied to search for the optimal feature subset and learn classification functions. Experiments showed that the proposed method outperformed other different features selection approaches in terms of the number of selected features and the classification accuracy.

Neshatian and Zhang [78] proposed a feature selection approach based on GP and a variation of NB. Bit-mask representation was used for feature subsets and a set of set operators were used as primitive functions. GP combined these feature subsets and set operators together to find the optimal subset of features. Experiment on a highly unbalanced face detection problem showed that the proposed algorithm could achieve a significant reduction in dimensionality and processing time.

Neshatian and Zhang [79] proposed a GP based filter model as a multi-objective approach for feature selection in binary classification problems. Unlike most filter methods that usually could only measure the relevance of single features to the class variables [59], the proposed algorithm could discover the hidden relationships between subsets of features and the target classes. In this method, an inexpensive binary relevance fitness function was defined to measure the relevance of a GP program tree (a subset of features) to the classification task. In order to explore large feature subsets and at the same time avoid overfitting and bloating, the standard GP was modified by adopting a run-time mechanism for depth control along with an overfit monitoring system. Moreover, a Pareto front archive was proposed as a multi-objective approach to maximising the relevance of subsets while minimising their sizes. So the result of the proposed method was a vector of Pareto front points serving as a trade-off matrix. Experiments showed that an inexpensive linear search over this vector could improve the classification performance of classifiers while decrease their complexity. However, the proposed method might not be quite appropriate for the problems where the best feature subset was expected to have a very large number of features.

Based on [79], Neshatian and Zhang [80] proposed a GP relevance measure (GPRM) to evaluate and rank feature subsets in binary classification tasks. GPRM extended the concept of a feature relevance measure function by proposing a virtual structure for GP program trees. Through case studies, it was found that the proposed method could detect relevant subsets of features in different situations including multimodal class distributions and mutually correlated features, where other methods had difficulties.

2.4.2.3 Ant Colony Optimisation (ACO) for Feature Selection

Ant colony optimisation (ACO) as an evolutionary computation technique has also been applied to feature selection problems. Typically, in an ACO based feature selection model, features are represented as nodes in the graph. Edges between the nodes indicate the possible choices of the next feature. Ants traverse through this graph to add nodes (features) until the stopping criterion has been satisfied. Feature selection problem is thus transformed to the problem of ant finding the best path on the graph [27].

Gao et al. [27] proposed an ACO based wrapper feature selection approach to network intrusion detection. In this wrapper model, least square based SVM was applied as the classifier to evaluate the feature subset generated by ants. Fisher discrimination rate was adopted as the heuristic information for ACO. Experiments on three datasets showed that the proposed method could be an effective approach to intrusion feature selection and detection.

Jensen [39] proposed a filter feature selection model based on ACO and fuzzy-rough theory. The proposed approach was examined on classification of web content and complex systems monitoring and good results were achieved. Experiments compared the proposed method with other five benchmark techniques. Results illustrated that hill-climbing feature selection approaches often failed to find minimal subsets even in small and medium-sized datasets. The proposed approach and a simulated annealing (SA) based feature selection algorithm achieved similar results and both of them outperformed other three benchmark techniques. However, the performance of the proposed approach has not been compared with that other evolutionary computation technique based feature selection methods, such as PSO or GAs based feature selection approaches.

Ke et al. [41] developed a Pareto-based multi-objective ACO for feature selection based on rough set theory. It adopted elite strategy to speed up the convergence performance, used the non-dominated solutions to add pheromone so as to reinforce the exploitation and applied crowding comparison operator to maintain the diversity of the constructed solutions. In addition, it intended to avoid premature convergence by imposing limits on pheromone values. Compared with a modified non-dominated sorting GA, the proposed method could obtain competitive solutions for rough feature selection. However, only three datasets were used in the experiments, which could not confirm the generalisation of the proposed approach.

2.5 Summary

This chapter reviewed the main concepts of machine learning and classification. The challenges in feature selection and construction were discussed in detail. Evolutionary computation techniques were also reviewed in this chapter, especially PSO. PSO is a heuristic search technique, which searches for the optimal solution of the problem based on social interactions between individuals in the population. Compared with other evolutionary computation techniques such as GAs and GP, PSO is computationally less expensive, easier to implement, has fewer parameters and can converge more quickly.

This chapter reviewed different modified versions of BPSO, and the related work of using conventional methods and evolutionary computation approaches to feature selection, single feature ranking and feature construction. The limitations of the existing work that form the motivations of this research were also discussed, which can be summarised as follows.

- Not all important characteristics of the PSO algorithm are completely present in BPSO, so improving the performance of BPSO is still an open issue. Research needs to be conducted to propose a novel encoding scheme, a new topology and effective update equations.

- In feature selection problems, neither conventional methods nor evolutionary computation approaches could address all the challenges discussed in Section 1.2. For example, the Relief algorithm does not deal with redundant features, hill-climbing approaches easily become stuck in local optima, and most of these approaches are computationally expensive. Meanwhile, not much work has been conducted solving a feature selection task as a multi-objective problem. Therefore, although BPSO has been applied successfully to feature selection problems, development of a highly accurate and efficient feature selection algorithm using BPSO is still an open issue.

- In single feature ranking, most of existing approaches rank features without considering the absence or presence of other features, which would not achieve good results in

problems with feature interactions (the second challenge in Section 1.2). Meanwhile, PSO has never been applied to single feature ranking problems. Therefore, development of a criterion dealing with feature interactions (e.g. epistasis) using BPSO to deal with single feature ranking problems is an open issue.

- In feature construction problems, GP has been widely used to solve the tasks, but GP based approaches usually suffer from expensive computation. As a powerful technique, the potential of BPSO for feature construction has not been explored.
- There has not been research conducted in predicting the best fitness evaluation function for a given learning algorithm in feature selection problems, which is an open issue.

This research aims to address the above-mentioned issues. The next chapter will focus on the initial work conducted in investigating PSO for feature selection.

Chapter 3

Preliminary Work

This chapter presents the initial work conducted in investigating PSO for feature selection. A set of experiments have been conducted to compare the performance of continuous PSO (CPSO) with that of binary PSO (BPSO) for feature selection. Two wrapper based feature selection approaches are proposed, which are single feature ranking and BPSO based feature subset ranking. In the first approach, individual features are ranked according to their classification accuracy so that feature selection can be achieved by using only a few top-ranked features for classification. In the second approach, BPSO is applied to feature subset ranking to search for optimal feature subsets. The two proposed approaches are compared with two conventional feature selection approaches, which are linear forward selection (LFS) [31] and greedy stepwise backward selection (GSBS) [12].

3.1 Comparisons between CPSO and BPSO

Both two versions of PSO (CPSO and BPSO) have been applied previously in feature selection problems [60, 36, 110, 100]. In order to compare the performance of CPSO and BPSO, a set of experiments have been conducted using both of them to search for optimal feature subsets in a wrapper feature selection model. The goal here is to investigate which version of PSO is better appropriate to feature selection problems so that it can be chosen for further development for this work.

3.1.1 Datasets and Parameter Settings

3.1.1.1 Datasets

This research will assume that the datasets are balanced datasets. Twelve datasets chosen from the UCI machine learning repository [26] are used in this research. Table 3.1 summarises the main characteristics of these datasets. The twelve datasets were selected to have different numbers of features, classes and instances as the representative samples of different kinds of classification problems that could be addressed by this research.

Eight of the twelve datasets in Table 3.1 were selected when conducting the preliminary works, which were Vowel, Wine, Australian, Zoo, Vehicle, German, WBCD, and Sonar. The other four datasets were selected recently, which will be used in subsequent works. Artificial datasets will be produced as the representative samples of the problems to test the performance of the methods that will be proposed in this work.

These datasets were chosen with an increasing number of features/dimensions. Tasks (datasets) with more than 500 features would not be preferred, as they belong to a different

Table 3.1: Datasets, representative samples of the classification tasks which are expected to be addressed by this research

Dataset	Number of features	Number of classes	Number of instances
Vowel	10	11	990
Wine	13	3	178
Australian	14	2	690
Letter	16	26	20,000
Zoo	17	7	101
Vehicle	18	4	846
Segmentation	19	7	2310
German	24	2	1000
World Breast Cancer -Diagnostic (WBCD)	30	2	569
Ionosphere	34	2	351
Satellite	36	6	6435
Sonar	60	2	208

Table 3.2: Parameter settings

Parameter	Value	Parameter	Value
c_1	1.49618	Population size	30
c_2	1.49618	Maximum iterations	100
w	0.768	Topology	Star
v_{max}	6.0	Runs	30

type of problem classification, i.e. large scale, which is which is beyond the scope of this work.

Eight datasets in Table 3.1, which are Vowel, Wine, Australian, Zoo, Vehicle, German, WBCD, and Sonar, were used in the experiments. The eight datasets were selected to have different numbers of features, classes and instances as the representative samples of the problems that the two proposed approaches could address.

3.1.1.2 Parameter Settings

A wrapper feature selection model needs a learning algorithm to evaluate the classification accuracy of the selected features. There are many learning algorithms that can be used here, such as K-nearest neighbour (KNN), naive bayes (NB), and decision tree (DT). One of the simplest learning algorithms, KNN, was selected as the learning algorithm in the wrapper model in this research and 5 nearest neighbours are used in KNN (K=5). 5NN with 10-fold cross-validation implemented in Java machine learning library (Java-ML) [1] is employed to calculate the classification error rate on both the training set and the test set. A detailed discussion of why and how n-fold cross-validation is applied in this way is given by Kohavi [50].

The parameter settings of BPSO are shown in Table 3.2. These values are chosen based on the common settings in the literature [104]. For both CPSO and BPSO, in each dataset, the instances are divided into two sets: 70% as the training set and 30% as the test set. The encoding scheme in CPSO is the same as the scheme described in Section 2.4.1 and the encoding scheme in BPSO is the same as the scheme described in Section 2.4.1.2.

During the search process of CPSO and BPSO on the training set, the fitness function is to minimise the classification error rate of the selected feature subset, which is evaluated by Java-ML according to Equation 3.1:

$$Error\ Rate = 1 - accuracy = \frac{FP + FN}{TP + TN + FP + FN} \tag{3.1}$$

where TP, TN, FP and FN stand for true positives, true negatives, false positives and false negatives, respectively.

3.1.2 Experimental Results

This research aims to compare the search ability of CPSO with BPSO on feature selection problems. Therefore, only the results that show the difference in fitness (classification error rate) of the search ability are shown in this section.

Figure 3.1 shows the change of average fitness during the search process of CPSO and BPSO on the training set in eight datasets. Each plot in the figures corresponds to one of the eight datasets used in the experiments. In each plot, the horizontal axis shows the iterations in the search process. The vertical axis shows the average fitness values in each generation over the 30 runs, which is also the training error rate. Therefore, the plots actually show the decrease of the training error rate during the search process.

According to Figure 3.1, both CPSO and BPSO could converge in a small number of generations, which is 60 in most cases. BPSO could achieve better performance (lower classification error rate) than CPSO. Therefore, this PhD thesis will investigate research on BPSO as a potentially appropriate search technique to solve feature selection problems.

3.2 Wrapper Based Single Feature Ranking

A wrapper based single feature ranking approach is now proposed, where the relative importance of each feature is measured by its classification accuracy. The goal is to investigate whether the combination of top-ranked features generated by this algorithm can achieve better performance than using all features and can outperform conventional approaches.

Algorithm 2 shows the pseudo-code of the proposed wrapper based single feature ranking approach. In this approach, each dataset is divided into two sets: a training set and a test set. In both the training set and the test set, KNN with n-fold cross-validation is employed to evaluate the classification accuracy [50]. In this algorithm, firstly, in order to make sure n-fold cross-validation is always performed on the n fixed folds, both the training set and the test set are divided into n folds when the algorithms starts. Secondly, every feature is used for classification in the training set individually and its classification accuracy is calculated by a loop of n-fold cross-validation on the fixed n folds of training data (from Line 4 to Line 7 in Algorithm 2). Thirdly, the features are ranked according to the classification accuracies they achieve. Finally, based on the order of the ranked features, successive numbers of the top-ranked features are selected for classification to show the utility of single feature ranking in feature selection and the classification accuracy is calculated by KNN with n-fold cross-validation on the fixed n folds of the test data (from Line 9 to Line 12 in Algorithm 2).

In each dataset, the aim of this algorithm is to determine the number of successive top-ranked features that can achieve classification accuracy close to or even better than the classifier with all features as input.

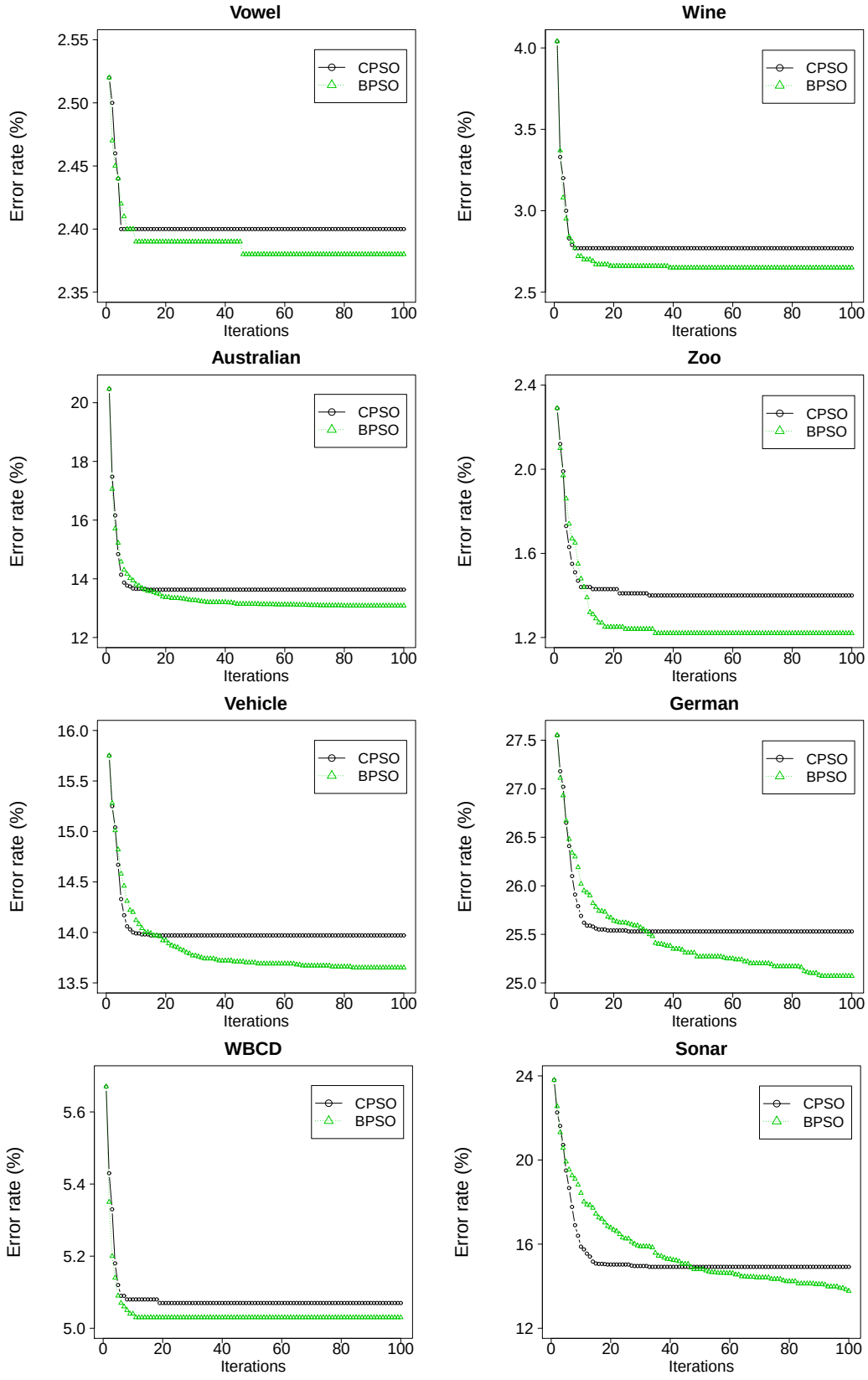


Figure 3.1: Comparisons between CPSO and BPSO: the difference in fitness (classification error rate) of the search ability.

Algorithm 2: The wrapper based single feature ranking algorithm

```
1  begin
2      divide the training set to n folds;                    // n-fold cross-validation
3      divide the test set to n folds;
4      for d=1 to number of features do
5          keep feature d and remove all the other features from training set ;
           // training set only contains feature d
6          use KNN with n-fold cross-validation to evaluate the classification accuracy of
           feature d for the training set;
7      end
8      rank the features according to the classification accuracy;
9      for d=1 to number of features do
10         keep d top-ranked features and remove the others from the test set;
11         use KNN with n-fold cross-validation to evaluate the classification accuracy of
           d top-ranked features for the test set;
12     end
13     return classification accuracy achieved by each feature;
14     return the order of features;
15     return the classification accuracies achieved by the successive numbers of the
       top-ranked features;
16 end
```

3.3 BPSO Based Feature Subset Ranking

The top-ranked feature set resulting from the single feature ranking algorithm might contain potential redundancy. For example, the combination of the two top-ranked features might not perform as well as the combination of one top-ranked feature and a low-ranked feature if the two top-ranked features are highly dependent. To overcome this problem, a feature subset ranking algorithm is proposed based on BPSO. Different feature subsets are evolved and ranked according to the classification accuracy on the training set. The goal is to investigate whether this algorithm can outperform the method of using all features, conventional approaches and the single feature ranking algorithm (Section 3.2).

If a dataset includes D features, the possible number of features that a feature subset contains will be in $[1, D]$. Therefore, according to the number of features they contain, all the possible feature subsets can be categorised into D groups. Each group includes many feature subsets that contain the same number of features and this algorithm aims to evolve the best feature subset from each group. So, for a dataset including D features, D feature subsets will be evolved and ranked. The dth feature subset is saved to contain the best feature subset that includes d features in this method, where d is a positive integer from 1 to D.

Algorithm 3 shows the pseudo-code of BPSO for feature subset ranking. In this approach, each dataset is firstly divided into two sets: a training set and a test set. KNN with n-fold cross-validation is employed to evaluate the classification accuracy [50] in both of the training set and the test set, which are divided into n folds, respectively. The feature subset search process starts from finding the best subset including one feature and ends with the feature subset with D features.

The process of selecting a certain feature subset is one step in this approach. For a dataset including D features, D feature subsets will be evolved and D steps are needed. Each step can be regarded as a process of using BPSO to select a certain number of the most relevant

Algorithm 3: The BPSO based feature subset ranking algorithm

```
1  begin
2      divide the training set to n folds                    // n-fold cross-validation
3      divide the test set to n folds;
4      initialise a feature subset S by randomly selecting 1 feature;
5      for d=1 to number of features do
6          initialise half of the swarm in BPSO with S;
7          initialise the other half of the swarm with a subset randomly selecting d
           features;
8          while maximum iteration or fitness=1 is not met do
9              for p=1 to number of particles do
10                 calculate sum (number of the selected features by particle p);
11                 if sum > d then
12                     randomly exclude (sum − d) features;
13                 end
14                 else if sum < d then
15                     randomly include (d − sum) features;
16                 end
17                 use KNN with n-fold cross-validation to evaluate the fitness of particle
                   p        // classification accuracy of d features selected by
                   particle p for the training set
18             end
19             for p=1 to number of particles do
20                 update Pbest_p and Gbest;
21             end
22             for p=1 to number of particles do
23                 update the velocity of particle p (Equation 2.2);
24                 update the position of particle p (Equations 2.3 and 2.4);
25             end
26         end
27         record the evolved feature subset and the corresponding classification
           accuracy;
28         S ← the recorded feature subset in Line 27;
29     end
30     rank the learnt feature subsets;
31     use KNN with n-fold cross-validation to calculate the classification accuracy of the
       ranked feature subsets for the test set;
32     return the order of feature subsets with classification accuracies;
33 end
```

features (from Line 8 to Line 26 in Algorithm 3). The dth step is actually the process of using BPSO to search d most relevant features and the fitness function of BPSO is to maximise the classification accuracy. During the search process of BPSO, if a particle selects more than d features, a deletion strategy is employed to randomly exclude features to reduce the number of features to d. On the other hand, if the number of selected features is smaller than d, an addition strategy is applied to randomly include features to increase the number of the selected features to d.

During the search process, when searching for the dth feature subset, half of the pop-

ulation in BPSO is initialised with the $(d-1)$th feature subset achieved in the $(d-1)$th step. This is due to the expection that some of the features in the $(d-1)$th subset are useful and should be retained in the dth subset. Meanwhile, each particle in the other half of the population is initialised with a feature subset that randomly selects d features to ensure the diversity of the swarm.

All the evolved feature subsets are ranked according to the classification accuracy on the training set and then their classification performance is evaluated by KNN with n-fold cross-validation on the test set. In each dataset, the aim is to determine the number of top-ranked feature subsets that can achieve classification accuracy close to or even better than the classifier with all features.

3.4 Results: Single Feature Ranking vs Feature Subset Ranking

3.4.1 Datasets and Parameter Settings

Eight datasets in Table 3.1, which are Vowel, Wine, Australian, Zoo, Vehicle, German, WBCD, and Sonar, were used in the experiments. The eight datasets were selected to have different numbers of features, classes and instances as the representative samples of the problems that the two proposed approaches could address. For two proposed approaches, in each dataset, the instances are divided into two sets: 70% as training set and 30% as test set. Classification accuracy is evaluated by 5NN with 10-fold cross-validation implemented in Java-ML. The classification accuracy is determined according to Equation 3.2:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

where TP, TN, FP and FN have the same meaning with that in Equation 3.1.

The parameter settings of BPSO are shown in Table 3.2.

For BPSO based feature subset ranking, the experiment has been conducted for 30 independent runs. The results achieved in different runs are similar to each other in terms of the classification accuracy of the evolved feature subsets. Therefore, the results from a typical run and the best results from 30 independent runs are shown in Section 3.4.3.

3.4.2 Benchmark Techniques

Two conventional wrapper feature selection methods, linear forward selection (LFS) and greedy stepwise backward selection (GSBS), are used as benchmark techniques to examine the performance of the two novel proposed approaches. LFS was derived from on SFS and GSBS was derived from SBS.

LFS [31] is an extension of best first algorithm. The search direction can be forward, or floating forward selection (with optional backward search steps). In LFS, the number of features considered in each step is restricted so that it does not exceed a certain user-specified constant. More details can be seen in the literature [31].

Greedy stepwise [12] implemented in Waikato Environment for Knowledge Analysis (Weka) [33] is a steepest ascent search. It can move through the search space either in forward direction or in backward direction. Given that LFS performs a forward selection, a backward search is chosen in greedy stepwise to conduct a greedy stepwise backward selection. GSBS begins with all features and stops when the deletion of any remaining attribute results in a decrease in evaluation, i.e. the accuracy of classification.

Weka is used to run the experiments when using LFS and GSBS for feature selection. During the feature selection process, 5NN with 10-fold cross-validation in Weka is employed

to evaluate the classification accuracy. In order to make fair comparisons, all the feature subsets selected by LFS, GSBS and two proposed methods are tested by 5NN with 10-fold cross-validation in Java-ML on the test sets.

When using Weka to run the experiments, all the settings are kept to the defaults except that backward search is chosen in the greedy stepwise approach to performing GSBS for feature selection and 5NN with 10-fold cross-validation is selected to evaluate the classification accuracy in both LFS and GSBS.

3.4.3 Experimental Results

Figure 3.2 shows the classification accuracy of each feature achieved by the wrapper based single feature ranking on the training set. The eight charts correspond to the eight datasets used in the experiments. In each chart, the horizontal axis shows the feature index in the corresponding dataset. The vertical axis shows the classification accuracy.

Figure 3.3 compares the classification performance of the two proposed methods, LFS and GSBS on the *test set*. Each plot corresponds to one of the eight datasets. In each plot, the horizontal axis shows the number of features used for classification and the vertical axis shows the classification accuracy. "SFR" in the figure stands for the results achieved by the successive numbers of top-ranked features in the wrapper based single feature ranking. For the BPSO based feature subset ranking, "FSR-Best" shows the best results in 30 independent runs and "FSR" shows the results achieved in a typical run. Both LFS and GSBS produce a unique feature subset, so have a single result for each test set. The red star denotes the classification accuracy achieved by LFS and the blue dot presents the result of GSBS. In addition, the red star and the blue dot in the plot of the Vowel dataset are in the same position, which means both methods selected the same number of features and achieved the same classification accuracy.

3.4.3.1 Results of Wrapper Based Single Feature Ranking

According to Figure 3.2, the classification accuracy achieved by each feature varies considerably, which means that each feature is not equally important for classification. In most cases, the difference between the highest classification accuracy and the lowest one is more than 20%, but it varies with the datasets. For example, the difference in the WBCD dataset is around 50% while the difference is only about 3% in the Vowel dataset. This is caused by the different characteristics of the different datasets.

According to the results denoted by "SFR" in Figure 3.3, selection of a small number of top-ranked features achieves better results than using all features in all the datasets. In almost all cases, using more top-ranked features, not only does not increase the performance, but actually causes a deterioration, especially for the Wine and Zoo datasets. The results suggest that there are interactions between some features such that the relevance of a feature changes in the presence or absence of some the other features.

3.4.3.2 Results of BPSO based Feature Subset Ranking

According to the results ("FSR" and "FSR-Best") in Figure 3.3, in all the eight datasets, with many of the feature subsets evolved by BPSO the classifier can achieve higher classification accuracy than with all features. In most cases, the feature subset with which the classifier achieves the best performance contains a small number of features. For example, in the Australian dataset, the second feature subset evolved by BPSO only includes two features, but achieves the highest classification accuracy. This suggests that BPSO can select the relevant features and eliminate some noisy and irrelevant ones.

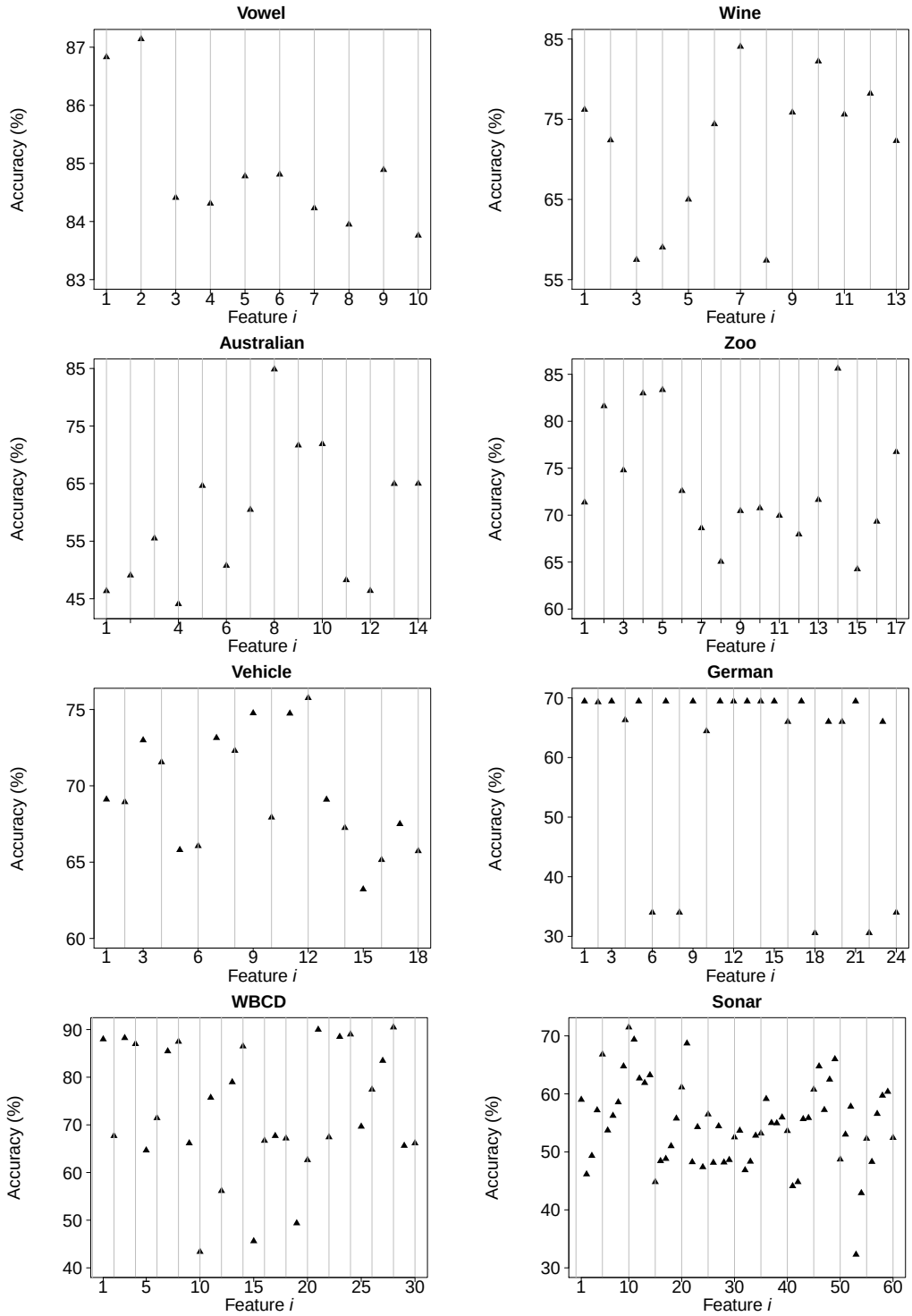


Figure 3.2: Results of single feature ranking: the classification accuracy of each feature.

3.4.3.3 Observations from the Two Proposed Methods

Comparing the two proposed methods for feature selection leads to the following observations. Firstly, using all features could not achieve the best performance in all the eight datasets. The two proposed methods could select a relatively small number of features with which the classifier could achieve higher classification accuracy than with all features. Secondly, in most cases, combing top-ranked features could not achieve the best performance

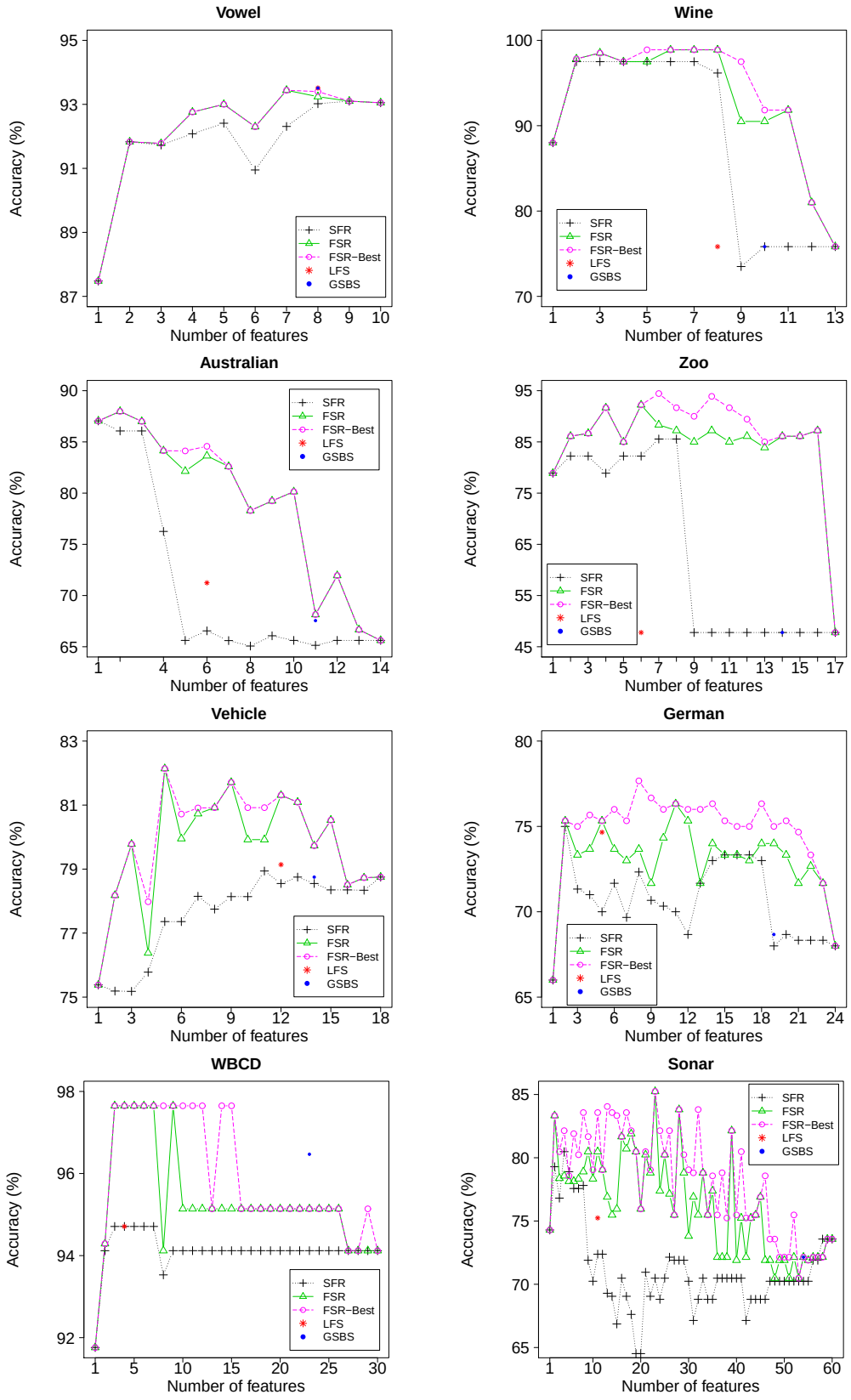


Figure 3.3: Comparisons between single feature ranking (SFR), feature subset ranking (FSR), the best results of FSR in 30 runs, linear forward selection (LFS) and greedy stepwise backward selection (GSBS).

because this combination still has redundancy. Thirdly, feature subset ranking provides an effective way for feature selection. Using the same number of features, BPSO based feature subset ranking can achieve higher classification accuracy than wrapper based single feature ranking. This suggests that BPSO could find a subset of complementary features to improve the classification performance.

3.4.3.4 Further Analysis

Results in Figure 3.3 show that in almost all cases, the feature subset evolved by BPSO is not the combination of the top-ranked features, but a subset of complementary ones.

Considering the Australian dataset as an example, as can be seen in Figure 3.2, the order of the ranked features is F8, F10, F9, F14, F13, F5, F7, F3, F6, F2, F11, F12, F1, F4, where Fi denotes the ith feature in the dataset. The second feature subset evolved by BPSO includes F8 and F12, which are not the two top-ranked features (F8 and F10). According to Figure 3.3, although with F8 and F10 the classifier can achieve higher classification accuracy than with all features, with F8 and F12 it can obtain better results than with F8 and F10. This suggests that the combination of the two top-ranked features contains redundancy while the combination of a top-ranked feature (F8) and a low-ranked feature (F12) is a subset of complementary features. Meanwhile, the other 11 (from the 3th to the 13th) feature subsets evolved by BPSO are not the combinations of the top-ranked features either. These results suggest that the BPSO based subset ranking algorithm has great potential to avoid redundant and/or noisy features and thus reduce the dimensionality of the classifier.

3.4.3.5 Comparisons Between Proposed Methods and Benchmark Techniques

The red stars and blue dots in Figure 3.3 show that the number of features selected by LFS is smaller than that of GSBS, but the classification accuracy achieved by LFS is close to or better than that of GSBS in most cases. This suggests that LFS starting with an empty feature subset is more likely to obtain some optimality of the small feature subsets than backward selection methods, but does not guarantee finding the larger feature subsets. GSBS starts with all features and a feature is removed only when its removal can improve the classification performance. The redundant features that do not influence the classification accuracy will not be removed. Therefore, the feature subset selected by GSBS is usually larger than the feature subset selected by LFS because of the redundant features.

Comparing the proposed wrapper based single feature ranking with the two conventional techniques, it can be observed that using the same number of features, LFS and GSBS could achieve higher classification accuracy than single feature ranking in most cases. This suggests that the combination of top-ranked features could not achieve the best performance because it contains redundancy or noise. However, in most cases, combing a relatively small number of top-ranked features could obtain higher accuracy than LFS and GSBS. The reseason might be that the feature subsets selected by LFS and GSBS still have redundancy.

Figure 3.3 shows that BPSO based feature subset ranking outperforms LFS and GSBS. In seven of the eight datasets, feature subsets obtained by feature subset ranking can achieve higher classification accuracy than the subsets obtained by LFS and GSBS (in the eighth one, the Vowel dataset, the results are almost the same). This suggests that BPSO could find subsets of complementary features that could achieve better classification performance than other combinations of features.

Chapter 4

Proposed Contributions and Project Plan

4.1 Proposed Contributions

This work will contribute to the fields PSO and feature selection. The proposed contributions are listed follows.

1. This work will present a novel version of discrete PSO with a new particle encoding scheme, a new topology structure and new update equations. This novel version of discrete PSO is expected to improve the performance of the PSO algorithm.

2. This work will present a new PSO based feature selection approach to solving feature selection as a single objective problem. This will be accomplished by developing a feature selection algorithm based on PSO and developing a strategy to improve its computational efficiency. The new PSO based feature selection approach is expected to achieve better performance than existing feature selection algorithms in terms of computational efficiency.

3. This work will present a PSO based single feature ranking algorithm for feature selection. A new criterion will be developed to evaluate the relative importance of each feature in the presence or absence of other features and then a PSO based single feature ranking approach will be proposed. This single feature ranking algorithm is expected to achieve better performance than existing single feature ranking approaches in terms of the classification accuracy of the successive top-ranked features.

4. This work will show how PSO can be used to solve feature selection tasks as a multi-objective problem. A multi-objective feature selection approach will be developed by proposing a multi-objective discrete PSO to maximise the classification accuracy and minimise the number of selected features. The proposed multi-objective feature selection approach is expected to achieve better performance than existing multi-objective feature selection algorithms and PSO based single objective approaches.

5. This work will show how PSO can be used for feature construction problems by utilising discrete PSO to select original features that are used to construct new high-level features. PSO has never been applied to feature construction problems because it does not have the construction function. In this work, new search operators will be introduced to discrete PSO thus it could evolve feature construction functions. The proposed feature construction approach is expected to automatically construct high-level

Table 4.1: Phases of project plan (Phase 6 will be conducted if time allowed).

Phase	Task	Duration (Months)
1	Reviewing literature, overall design, selection of datasets and writing the proposal	12 (Complete)
2	Developing a new version of discrete PSO for feature selection	5
3	Using discrete PSO to select a single subset of features	4 (Partly)
4	Investigating multi-objective discrete PSO for feature selection	4
5	Utilising discrete PSO for feature construction	5
6*	Predicting the best fitness evaluation function with which a feature selection algorithm could obtain the best feature subset for a given learning algorithm	3*
7	Writing the thesis	6

features which could improve the quality of the search space and achieve better performance than existing feature construction algorithms.

6. The performance of feature selection is bound to the classification algorithm in wrapper based approaches, but this link has not been throughly investigated. This work intends to present a method to predict the fitness evaluation function with which a feature selection algorithm could obtain the best feature subset for a given learning algorithm. This is expected to help users select the best fitness evaluation function for a desired learning algorithm to obtain the best classification accuracy in unseen data and/or reduce the training time.

4.2 Overview of Project Plan

The initial plan for this project includes eight overall phases of research as shown in Table 4.1. The first step has been completed and part of the rest of the steps have been done partly during the provisional registration period (12 months). Phases two to six will cover the major research objectives addressed by this project. Work in phase six is optional, which will be conducted if one of phases two to five (one of the first four objectives) can not be completed, or there is still enough time available when other work (except phase seven) is completed. The last phase involves writing the final thesis.

4.3 Project Timeline

An estimation of the approximate timetable for the remaining phases of the work for the next 24 months of research is presented in Table 4.2. The first column shows the research phase from Table 4.1, the second column the detailed research tasks, and the next subsequent columns represent two-month periods. Note that phase six is not included in the timetable because it is optional and three months may be needed to complete it if possible.

4.4 Thesis Outline

This thesis plans to have nine chapters, where *chapter 7* is optional work. The outline of the final thesis will be written as follows.

- *Chapter 1: Introduction*

Table 4.2: Project timeline for the next 24 months

		Time in Months											
Phase	Task	2	4	6	8	10	12	14	16	18	20	22	24
n/a	Updating the literature review	x	x	x	x	x	x	x	x	x	x	x	x
2	Developing a new encoding scheme for discrete PSO	x	x										
2	Designing a new topology for discrete PSO		x										
2	Investigating new update equations and add search operators to discrete PSO			x									
3	Developing a strategy to improve the computation efficiency of discrete PSO for feature selection			x	x								
3	Using discrete PSO to select a single subset of features				x								
3	Using discrete PSO for single feature ranking for feature selection					x							
4	Investigating multi-objective discrete PSO for feature selection					x	x	x					
5	Utilising discrete PSO for feature construction							x	x	x			
7	Writing the first draft of the thesis										x	x	
7	Editing the final draft											x	x

This chapter will introduce the academic thesis. It will outline the problems, the motivations, research goals, contributions and organisation of the thesis.

- *Chapter 2: Literature Review*

  This chapter will present a review of the literature on evolutionary computation techniques as main approaches to solving feature selection problems. It will cover essential background and basics concepts of evolutionary computation and machine learning, particularly PSO, feature selection and construction, and then it will review typical related work in feature selection and construction problems using conventional methods and evolutionary computation techniques. It thus highlights the main limitations of these algorithms and current challenges that form the motivations of the thesis.

- *Chapter 3: Improved Discrete PSO*

  In this chapter, a new version of discrete PSO will be proposed, which is based on a new encoding scheme, a new topology structure, new update equations and newly introduced search operators. The performance of the proposed PSO will be compared with that of BPSO with a standard encoding scheme, commonly used topology structures and standard update equations on feature selection problems.

- *Chapter 4: Discrete PSO for Feature Selection and Single Feature Ranking*

  In this chapter, a strategy will be proposed to improve the computational efficiency of discrete PSO. A feature selection approach will then be developed based on discrete PSO, which is a single objective technique aiming to maximise the classification accuracy. Meanwhile, a new criterion will be proposed to evaluate the relative importance of each feature in the presence or absence of other features. Then a discrete PSO based

single feature ranking approach will be developed for feature selection problems. The performance of the proposed algorithms will be compared with that of conventional approaches and standard PSO based feature selection methods.

- *Chapter 5: Multi-objective Discrete PSO for Feature Selection*

  In this chapter, a multi-objective discrete PSO will be proposed based on which a feature selection approach will be developed with the objectives of maximising the classification accuracy and minimising the number of features. The performance of the developed approach will be compared with that of existing multi-objective feature selection algorithms and the single objective PSO developed in the previous chapter.

- *Chapter 6: Discrete PSO for Feature Construction*

  In this chapter, a feature construction approach will be developed. Discrete PSO will be utilised to select original features for constructing new high-level features. New search operators will be introduced to discrete PSO to evolve a feature construction function. The performance of the proposed approach will be compared with that of benchmark feature construction algorithms.

- *Chapter 7*: Prediction of a Good Fitness Evaluation Function for Given Classification Algorithms*

  This chapter will investigate the best fitness evaluation function that can be used in the training process of a feature selection approach, which will obtain the best feature subset for a given learning algorithm. Some commonly used learning algorithms will be covered, such as KNN, NB, DT, SVM and LCS. Experiments will be conducted to test the predictions.

- *Chapter 8: Discussions*

  In this chapter, relevance and impact of the novel work in the PSO and feature selection fields will be discussed in detail.

- *Chapter 9: Conclusions and Future Work*

  In this chapter, the conclusions and findings from the experiments in each phase will be presented and summarised. It will also describe the main future research directions arising from the contributions of this work.

4.5 Resources Required

4.5.1 Computing Resources

This research will need to run large computational experiments, which consume much memory space and processing power. The School's grid computing facilities can meet the requirement. Other IT resources are also available in the School's system.

4.5.2 Library Resources

Access to the respective publications in this area is expected. Most of the required literature is already subscribed by the university library and also available on-line.

4.5.3 Conference Travel Grant

Publications to the respective conference (i.e. CEC and GECCO) in this area are expected. Therefore a travel grant from the university is needed to support important conference travels.

Bibliography

[1] T. Abeel, Y. V. de Peer, and Y. Saeys. Java-ml: A machine learning library. *Journal of Machine Learning Research*, 10:931–934, 2009.

[2] T. J. Ai and V. Kachitvichyanukul. Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers and Industrial Engineering*, 56(1):380– 387, 2009.

[3] E. Alba, J. Garcia-Nieto, L. Jourdan, and E. Talbi. Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *IEEE Congress on Evolutionary Computation (CEC'07)*, pages 284–290, 2007.

[4] H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69:279–305, 1994.

[5] E. Alpaydin. *Introduction to machine learning*. The MIT Press, 2004.

[6] M. AlRashidi and M. El-Hawary. A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, 13(4): 913–918, 2009.

[7] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

[8] G. Azevedo, G. Cavalcanti, and E. Filho. An approach to feature selection for keystroke dynamics systems based on pso and feature weighting. In *IEEE Congress on Evolutionary Computation (CEC'07)*, pages 3577–3584, 2007.

[9] M. Banerjee, S. Mitra, and H. Banka. Evolutionary rough feature selection in gene expression data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(4):622–632, 2007.

[10] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees*. Wadsworth International Group, 1984. BREIMAN84.

[11] C. Cardie. Using decision trees to improve case-based learning. In *In Proceedings of the Tenth International Conference on Machine Learning (ICML)*, pages 25–32, 1993.

[12] R. Caruana and D. Freitag. Greedy attribute selection. In *International Conference on Machine Learning (ICML'94)*, pages 28–36, 1994.

[13] B. Chakraborty. Genetic algorithm with fuzzy fitness function for feature selection. In *IEEE International Symposium on Industrial Electronics (ISIE'02)*, volume 1, pages 315–319, 2002.

[14] B. Chakraborty. Feature subset selection by particle swarm optimization with fuzzy fitness function. In *3rd International Conference on Intelligent System and Knowledge Engineering (ISKE'08)*, volume 1, pages 1038–1042, 2008.

[15] B. C. Chien and J. H. Yang. Features selection based on rough membership and genetic programming. In *IEEE International Conference on Systems, Man and Cybernetics (SMC'06)*, volume 5, pages 4124–4129, 2006.

[16] L. Y. Chuang, H. W. Chang, C. J. Tu, and C. H. Yang. Improved binary pso for feature selection using gene expression data. *Computational Biology and Chemistry*, 32(29):29–38, 2008.

[17] L. Y. Chuang, S. W. Tsai, and C. H. Yang. Improved binary particle swarm optimization using catfish effect for feature selection. *Expert Systems with Applications*, 38:12699–12707, 2011.

[18] J. Cooper and C. Hinde. Improving genetic algorithms' efficiency using intelligent fitness functions. In *Developments in Applied Artificial Intelligence*, volume 2718, pages 636–643. Springer Berlin / Heidelberg, 2003.

[19] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(4): 131–156, 1997.

[20] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28 –39, 2006.

[21] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Sixth International Symposium on Micro Machine and Human Science (MHS'95)*, pages 39–43, 1995.

[22] A. P. Engelbrecht. *Computational intelligence: an introduction (2. ed.)*. Wiley, 2007.

[23] M. A. Esseghir, G. Goncalves, and Y. Slimani. Adaptive particle swarm optimizer for feature selection. In *international conference on Intelligent data engineering and automated learning (IDEAL'10)*, pages 226–233, Berlin, Heidelberg, 2010. Springer Verlag.

[24] P. Estevez, M. Tesmer, C. Perez, and J. Zurada. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, 20(2):189–201, 2009.

[25] A. O. Finley and R. E. McRoberts. Efficient k-nearest neighbor searches for multisource forest attribute mapping. *Remote Sensing of Environment*, 112(5):2203 – 2211, 2008.

[26] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[27] H. H. Gao, H. H. Yang, and X. Y. Wang. Ant colony optimization based network intrusion feature selection and detection. In *International Conference on Machine Learning and Cybernetics*, volume 6, pages 3871–3875, 2005.

[28] I. A. Gheyas and L. S. Smith. Feature subset selection in large dimensionality domains. *Pattern Recognition*, 43(1):5–13, 2010.

[29] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[30] Y. Guo, W. Li, A. Mileham, and G. Owen. Applications of particle swarm optimisation in integrated process planning and scheduling. *Robotics and Computer-Integrated Manufacturing*, 25(2):280–288, 2009.

[31] M. Gutlein, E. Frank, M. Hall, and A. Karwath. Large-scale attribute selection using wrappers. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09)*, pages 332–339, 2009.

[32] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11:931–934, 2009.

[34] M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18 –28, 1998.

[35] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification, 2003.

[36] C. L. Huang and J. F. Dun. A distributed pso-svm hybrid system with feature selection and parameter optimization. *Application on Soft Computing*, 8:1381–1391, 2008.

[37] C. L. Huang and C. J. Wang. A ga-based feature selection and parameters optimization for support vector machines. *Expert Systems with Applications*, 31(2):231– 240, 2006.

[38] A. Jain and D. Zongker. Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[39] R. Jensen. Performing feature selection with aco. In *Swarm Intelligence in Data Mining*, volume 34 of *Studies in Computational Intelligence*, pages 45–73. Springer Berlin / Heidelberg, 2006.

[40] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference (ICCCS'11)*, pages 121–129. Morgan Kaufmann Publishers, 1994.

[41] L. Ke, Z. Feng, Z. Xu, K. Shang, and Y. Wang. A multiobjective aco algorithm for rough feature selection. In *Second Pacific-Asia Conference on Circuits,Communications and System (PACCS)*, volume 1, pages 207–210, 2010.

[42] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *IEEE Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 1931–1938, 1999.

[43] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[44] J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation.*, volume 5, pages 4104–4108, 1997.

[45] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *IEEE Congress on Evolutionary Computation (CEC'02)*, volume 2, pages 1671–1676, 2002.

[46] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. Evolutionary Computation Series. Morgan Kaufman, San Francisco, 2001.

[47] M. Khanesar, M. Teshnehlab, and M. Shoorehdeli. A novel binary particle swarm optimization. In *Mediterranean Conference on Control Automation (MED'07)*, pages 1–6, 2007.

[48] K. Kira and L. A. Rendell. A practical approach to feature selection. *Assorted Conferences and Workshops*, pages 249–256, 1992.

[49] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, pages 129–134, 1992.

[50] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[51] D. Koller and M. Sahami. Toward optimal feature selection. In *International Workshop And Conference on Machine Learning*, pages 284–292. Morgan Kaufmann, 1996.

[52] I. Kononenko. Estimating attributes: Analysis and extensions of relief. *Lecture Notes in Computer Science*, 784:171, 1994.

[53] J. R. Koza. Introduction to genetic programming. In *Annual conference on Genetic and evolutionary computation (GECCO'07)*, volume 5, pages 3323–3365, 2007.

[54] K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3: 329–343, 2002.

[55] K. Krawiec. *Evolutionary Feature Programming: Cooperative learning for knowledge discovery and computer vision*. Number 385 in . Wydawnictwo Politechniki Poznanskiej, Poznan University of Technology, Poznan, Poland, 2004.

[56] N. Kwak and C. H. Choi. Input feature selection for classification problems. *IEEE Transactions on Neural Networks*, 13(1):143–159, 2002.

[57] P. Langley. Selection of relevant features in machine learning. In *In Proceedings of the AAAI Fall symposium on relevance*, pages 127–131. AAAI Press, 1994.

[58] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *IN PROCEEDINGS OF THE TENTH NATIONAL CONFERENCE ON ARTI CIAL INTELLIGENCE*, pages 223–228. MIT Press, 1992.

[59] M. Last, A. Kandel, and O. Maimon. Information-theoretic algorithm for feature selection. *Pattern Recognition Letters*, 22:799–811, 2001.

[60] S. W. Lin, K. C. Ying, S. C. Chen, and Z. J. Lee. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications*, 35(4):1817–1824, 2008.

[61] H. Liu and H. Motada, editors. *Feature extraction, construction and selection: A data mining perspective*. Kluwer Academic Publishers, Norwell, MA, 1998.

[62] H. Liu and H. Motoda. Feature transformation and subset selection. *IEEE Intelligent Systems and Their Applications*, 13(2):26— 28, 1998.

[63] Y. Liu, G. Wang, H. Chen, H. Dong, X. Zhu, and S. Wang. An improved particle swarm optimization for feature selection. *Journal of Bionic Engineering*, 8(2):191–200, 2011.

[64] T. Marill and D. Green. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9(1):11–17, 1963.

[65] M. Menhas, M. Fei, L. Wang, and X. Fu. A novel hybrid binary pso algorithm. *Advances in Swarm Intelligence, Lecture Notes in Computer Science*, 6728:93–100, 2011.

[66] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors. *Machine Learning: An Artificial Intelligence Approach*. Tioga, 1983.

[67] D. Michie, D. J. Spiegelhalter, and C. Taylor. Machine learning, neural and statistical classification, 1994.

[68] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

[69] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[70] A. Modiri and K. Kiasaleh. Modification of real-number and binary pso algorithms for accelerated convergence. *Antennas and Propagation, IEEE Transactions on*, 59(1):214–224, 2011.

[71] A. Mohemmed, M. Zhang, and M. Johnston. Particle swarm optimization based adaboost for face detection. In *IEEE Congress on Evolutionary Computation (CEC'09)*, pages 2494–2501, 2009.

[72] A. M. Molinaro, R. Simon, and R. M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005.

[73] M. A. Muharram and G. D. Smith. The effect of evolved attributes on classification algorithms. In *Australian Conference on Artificial Intelligence (AI'03)*, volume 2903 of *Lecture Notes in Computer Science*, pages 933–941. Springer, 2003.

[74] D. Muni, N. Pal, and J. Das. Genetic programming for simultaneous feature selection and classifier design. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(1):106–117, 2006.

[75] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[76] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 26(9):917–922, 1977.

[77] K. Neshatian and M. Zhang. Genetic programming for performance improvement and dimensionality reduction of classification problems. In *IEEE Congress on Evolutionary Computation (CEC'08)*, pages 2811–2818, 2008.

[78] K. Neshatian and M. Zhang. Dimensionality reduction in face detection: A genetic programming approach. In *24th International Conference Image and Vision Computing New Zealand (IVCNZ'09)*, pages 391–396, 2009.

[79] K. Neshatian and M. Zhang. Pareto front feature selection: using genetic programming to explore feature space. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO'09)*, pages 1027–1034, New York, NY, USA, 2009.

[80] K. Neshatian and M. Zhang. Genetic programming for feature subset ranking in binary classification problems. In *European Conference on Genetic Programming*, pages 121–132, 2009.

[81] K. Neshatian, M. Zhang, and M. Johnston. Feature construction and dimension reduction using genetic programming. In *Australian Conference on Artificial Intelligence (AI'07)*, volume 4830 of *Lecture Notes in Computer Science*, pages 160–170. Springer, 2007.

[82] K. Neshatian, M. Zhang, and P. Andreae. Genetic programming for feature ranking in classification problems. In *Simulated Evolution and Learning*, volume 5361 of *Lecture Notes in Computer Science*, pages 544–554. Springer Berlin / Heidelberg, 2008.

[83] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen. Feature selection using multiobjective genetic algorithms for handwritten digit recognition. In *16th International Conference on Pattern Recognition (ICPR'02)*, volume 1, pages 568– 571, 2002.

[84] F. Otero, M. Silva, A. Freitas, and J. Nievola. Genetic programming for attribute construction in data mining. In *Genetic Programming*, volume 2610 of *Lecture Notes in Computer Science*, pages 101–121. Springer Berlin/Heidelberg, 2003.

[85] G. Pampara, N. Franken, and A. P. Engelbrecht. Combining particle swarm optimisation with angle modulation to solve binary problems. In *IEEE Congress on Evolutionary Computation (CEC'05)*, pages 89–96, 2005.

[86] A. Paoli, F. Melgani, and E. Pasolli. Clustering of hyperspectral images based on multiobjective particle swarm optimization. *IEEE Transactions on Geoscience and Remote Sensing*, 47(12):4175–4188, 2009.

[87] P. Pudil, J. Novovicova, and J. V. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[88] A. Purohit, N. Chaudhari, and A. Tiwari. Construction of classifier with feature selection based on genetic programming. In *IEEE Congress on Evolutionary Computation (CEC'10)*, pages 1–5, 2010.

[89] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[90] R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[91] R. Ramirez and M. Puiggros. An evolutionary computation approach to cognitive states classification. In *IEEE Congress on Evolutionary Computation (CEC'07)*, pages 1793–1799, 2007.

[92] R. Ruiz, J. C. R. Santos, and J. S. Aguilar-Ruiz. Fast feature ranking algorithm. In *Knowledge-Based Intelligent Information and Engineering Systems (KES'03)*, volume 2773 of *Lecture Notes in Computer Science*, pages 325–331. Springer, 2003.

[93] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (Second Edition)*. Pearson Education, 2003.

[94] J. Sadri and C. Suen. A genetic binary particle swarm optimization model. In *IEEE Congress on Evolutionary Computation (CEC'06)*, pages 656–663, 2006.

[95] R. K. Sivagaminathan and S. Ramakrishnan. A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert Systems with Applications*, 33(1):49– 60, 2007.

[96] M. G. Smith and L. Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, 2005.

[97] S. Stearns. On selecting features for pattern classifier. In *In Proceedings of the 3rd International Conference on Pattern Recognition*, pages 71–75, Coronado, CA, 1976.

[98] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003.

[99] E. G. Talbi, L. Jourdan, J. Garcia-Nieto, and E. Alba. Comparison of population based metaheuristics for feature selection: Application to microarray data classification. In *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA'08)*, pages 45–52, 2008.

[100] A. Unler and A. Murat. A discrete particle swarm optimization method for feature selection in binary classification problems. *European Journal of Operational Research*, 206(3):528–539, 2010.

[101] H. Vafaie and K. DeJong. Feature space transformation using genetic algorithms. *IEEE Intelligent Systems*, 13(2):57–65, 1998.

[102] H. Vafaie and I. Imam. Feature selection methods: Genetic algorithms vs. greedy-like search. In *International Conference on Fuzzy Systems and Intelligent Control Conference*, volume 1, pages 217–220, 1994.

[103] J. Valyon and G. Horváth. A WEIGHTED GENERALIZED LS–SVM, 2003.

[104] F. Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Pretoria, South Africa, 2002.

[105] L. Wang, X. Wang, and M. Fei. An adaptive mutation-dissipation binary particle swarm optimisation for multidimensional knapsack problem. *Journal of Modelling, Identification and Control*, 8(4):259–269, 2009.

[106] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen. Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*, 28(4):459–471, 2007.

[107] K. Waqas, R. Baig, and S. Ali. Feature subset selection using multi-objective genetic algorithms. In *IEEE 13th International Conference on Multitopic Conference (INMIC'09)*, pages 1–6, 2009.

[108] A. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, C-20(9):1100–1103, 1971.

[109] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2005.

[110] C. S. Yang, L. Y. Chuang, C. H. Ke, and C. H. Yang. Boolean binary particle swarm optimization for feature selection. In *IEEE Congress on Evolutionary Computation (CEC'08)*, pages 2093–2098, 2008.

[111] C. S. Yang, L. Y. Chuang, J. C. Li, and C. H. Yang. Chaotic maps in binary particle swarm optimization for feature selection. In *IEEE Conference on Soft Computing in Industrial Applications(SMCIA '08)*, pages 107–112, 2008.

[112] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004.

[113] H. Yuan, S. S. Tseng, W. Gangshan, and Z. Fuyan. A two-phase feature selection method using both filter and wrapper. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC'99)*, volume 2, pages 132–136, 1999.

[114] S. C. Yusta. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, 30:525–534, 2009.

[115] H. M. Zhao, A. P. Sinha, and W. Ge. Effects of feature construction on classification performance: An empirical study in bank failure prediction. *Expert Systems with Applications*, 36(2):2633–2644, 2009.

[116] Z. Zheng. Effects of different types of new attribute on constructive induction. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI'96)*, pages 254–257, 1996.

[117] Z. Zheng. Constructing x-of-n attributes for decision tree learning. *Machine Learning*, 40:35–75, 2000.