VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

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

Improving Continuous PSO for Feature Selection using Statistical Clsutering and Genetic Operators

Hoai Bach Nguyen

Supervisors: Bing Xue, Mengjie Zhang

Submitted in partial fulfilment of the requirements for
Bachelor of Science with Honours in Computer Science.

Abstract

Feature selection is an important problem in classification tasks. However, it is a difficult task due to the large search space in the original dataset as well as the complex interaction between features. Statistical clustering methods considers feature interaction to group similar features in the same feature cluster. This work firstly introduces a new representation for continuous particle swarm optimisation (PSO), which takes the advantage of statistical clustering to select a small number of features while still achieves high classification performance. A new hybrid PSO based algorithm is then proposed in this project, which embeds genetic operators into standard PSO to avoid premature convergence problem. Experimental results show that by using the statistical clustering information in PSO for feature selection, small subsets of features are evolved, which achieve significant improvement over using all features in terms of the classification performance. Furthermore, the hybrid PSO algorithm is found to better explore the search space than the standard PSO for feature selection.

Acknowledgments

I wish to acknowledge the research into the statistical clustering of data undertaken by Ivy Liu, which has been used to cluster features that are used in the PSO algorithm presented in this project. I would like to express the deepest appreciation to my two supervisors, Bing Xue and Mengjie Zhang, who spend poured countless hours on encouraging and supporting me during this project.

Contents

Chapter 1

Introduction

1.1 Introduction

Classification is one of the most important tasks in machine learning, which aims to predict the class label of an instance based on the value of instance's features. In the learning process, a set of instances, called the training set, is used to train a classification algorithm, which is tested on an unseen dataset, called the test set. In many problems, a large number of features is used to well describe the instances. Unfortunately, due to "the curse of dimensionality"[12], the larger a set of features is, the longer time the training process takes. In addition, relevant features are often unknown without prior knowledge. Therefore, a large number of features often contains irrelevant or redundant features, which are not useful for classification. Those features might lower the quality of the whole feature set [46], because they usually conceal the useful information from the relevant features. Feature selection methods [16] are used to remove those redundant and irrelevant features, which will not only speed up the learning/classification process but also maintain or even increase the classification performance over using all features. However, due to the complex interaction between features and the huge search space, it is hard to develop a good feature selection approach.

The main goal of feature selection is finding a small feature subset from a large set of original features to achieve similar or even better classification performance than using all features. In feature selection, suppose there are n features introduced, then the total number of possible subsets is 2^n. It can be seen that over the large search space, the exhaustive search is too slow to perform in most situations. In order to reduce the searching time, some greedy algorithms such as sequential forward selection [39] and sequential backward selection [26] are developed. However, these methods easily get stuck at the local optima. Because of the global search ability, evolutionary computation (EC) techniques, such as genetic programming (GP) [28], genetic algorithm (GAs) [45] and particle swarm optimization (PSO) [37, 44], have been applied to solve the feature selection problem. Compared with GA and GP, PSO is more preferable because it is simple and easy to implement. In addition, PSO not only uses fewer parameter but also converge more quickly.

Feature selection can be viewed as a multi-objective problem because it needs to maximize the classification accuracy and simultaneously minimize the dimensionality of the selected subset. However, with fewer features being used for classification, the classification accuracy is likely decreased. Therefore, those two objectives often conflict with each other and the searching process needs to consider the trade-off between them. Furthermore, statistical clustering methods (SCm) [22] can be applied as a pre-processing step of selecting process, where the similar features are grouped together in the same cluster. There are two main kinds of PSO, which are continuous PSO [18, 34] and binary PSO [20]. Most of

PSO-based feature selection algorithms use binary PSO since its binary representation can naturally present a feature subset. But the research in [40] shows that continuous PSO can achieve better performance than binary PSO.Therefore, this work will develop a new representation in continuous PSO to utilise the benefits of feature clustering information.

1.2 Objectives

The overall goal of this works is to develop a new PSO based feature selection algorithm, which selects a small feature subset while achieving similar or even better classification performance than using all features. The performance of feature selection methods will be evaluated on a number of datasets with different numbers of features, classes and instances. In particular, the overall goal of this project can be divided into two main objectives:

- Objective 1: to develop new representation and updating methods for continuous PSO (CPSO), which can effectively utilise the statistical clustering information to select a smaller number of features and achieve better classification performance than using all features.
- Objective 2: to combine genetic operators with continuous PSO. By introducing genetic operators such as mutation or crossover, the diversity of swarm is expected to be ensured to better explore the search space to further improve the performance.

1.3 Major Contributions

The project has three major contributions. Firstly, a new representation of PSO is developed, which utilises statistical clustering information to select a small set of features from each cluster. The first research in PSO for feature selection based on statistical clustering information was conducted by Lane, et al. [23]. However, in that project [23] the number of selected features from each cluster is predefined, which might lead to selecting redundant features or skipping the best complimentary features subset. In addition, most of the existing PSO-based feature selection approaches use binary PSO, which is not as good as continuous PSO [40]. Compared to the algorithm proposed by Lane [23], this algorithm lets PSO to dynamically determine the number of selected features from a certain cluster, which is expected to achieve a better subset of features. The work in this project represents the first research which combines the continuous PSO and statistical clustering information to solve feature selection problems. Part of the work has been accepted by the 10th International Conference on Simulated Evolution and Learning (SEAL 2014) [2] and will be published by Springer in the Lecture Notes in Computer Science (LNCS).

The second contribution is to further improve the representation proposed in the first contribution by applying a Gaussian distribution to PSO to better explore the continuous search space. The combination of Gaussian distribution and statistical clustering was proposed by Lane [23]. In that work [23], the distribution is only used to determine the number of features to be selected from each cluster. According to that predefined number, features are selected from their clusters by other mechanism. The combination approach in this project is significantly different. In particular, the Gaussian distribution is applied to the representation to directly determine which features are selected from a certain cluster. This contribution addresses the limitation the new representation to achieve better performance. The work are under preparation for submission to the Evostar 2015 [1]

The last contribution is a new hybrid PSO algorithm, in which genetic operators are embedded into the traditional PSO to avoid premature convergence in PSO. There exists other known papers [11, 33] that combine GA and PSO to take the advantages from both algorithms. However, the integration between two algorithms is still at low level. In addition none of these combination is used to solve feature selection problems. In this work, the genetic operators are deeply applied in the updating process of PSO, which is expected to help PSO better explore the search space to achieve higher classification accuracy in feature selection problems. This work will be investigated more in the future to prepare for an international journal.

1.4 Organisation

The remainder of this report is organised as follows. Chapter 2 provides background information. Objective 1 is discussed in Chapter 3 and 4. Chapter 5 presents a new hybrid PSO based algorithm, which aims to solve the second objective. Chapter 6 provides a discussion of the major conclusions drawn from this project and some remaining future work.

Chapter 2

Background

2.1 Machine Learning and Classification

Machine learning is a major field of Artificial Intelligence, which aims to construct a system that is capable to learn from data rather than explicitly follows programmed instructions. So a machine learning can automatically improve its performance for a certain problem when it gains more experience. Based on the desired output of the algorithm, machine learning algorithms can be grouped into three main categories: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, all instances (inputs) given to the system are labelled by desired outputs. The task of supervised learning is generating a general rule that maps inputs to outputs. Unlike supervised learning, the learner in unsupervised learning is learning with unlabelled instances. It attempts to find out the way to determine groups that consist of similar inputs. In reinforcement learning, the system interacts with a dynamic environment, in which it must perform a certain goal.

Classification is one of the most important tasks in supervised learning, which is to assign a class label to a given input instance.Each classification algorithm aims to learn a classifier, which is expected to correctly predict the class label for unseen instances. In classification problem, a training set consists of instances, which are used to induce the classifier. The performance of the learnt classifier is then evaluated on the test set, which is a set of instances in the same problem domain as the training set. The instances within the test set are never used during the training process, which means that the test set plays a role as unseen data.

2.2 Feature Selection

In classification problems, a large number of features is usually used to well describe the instances of datasets. However, due to "the curse of dimensionality"[12], having a large number of features results in a long classifier training time, a complex classifier structure and poor predictive performance. In addition, a large set of features might contain redundant and/or irrelevant features, which would not improve or even decrease the classification accuracy. Feature selection is a technique, which selects a small subset of features from the original features. It can not only improve the classification performance but also reduce the training time.

According to the evaluation criterion, existing feature selection methods can be fallen into two categories: wrapper approaches and filter approaches [9, 21]. In a wrapper approach, a learning algorithm is used to calculate the fitness value of the selected features. Meanwhile, a filter approach is done in an independent way of learning algorithms. Therefore, wrapper methods usually can achieve better classification accuracy than filters. But wrappers may produce a feature subset with poor generality, which is only good for specific a learning algorithm. In addition, compared with wrappers, filter methods are usually less expensive in terms of the computation complexity. This work focuses mainly on wrapper feature selection.

2.3 Evolutionary Computation

In Artificial Intelligence, Evolutionary Computation (EC) techniques are well known because of their global search ability. These techniques are inspired by the principles of biological mechanisms, social interaction or swarm intelligence. Compared to other machine learning techniques such as Neural Network that optimises a single solution, EC can be viewed as population based meta-heuristic algorithms. The population is a set of candidate solutions, which are evolved with respect to a user's predefined fitness function. At the end of the evolutionary process, the solution with the highest fitness is selected as the best evolved solution. In general, EC consists of evolutionary algorithms (EA), swarm intelligence (SI) and other techniques. Genetic Algorithm (GAs) is one of the most important algorithm in EA, while PSO is known as a member of SI.

2.3.1 Genetic Aglgorithms (GAs)

Genetic Algorithms [14] are evolutionary algorithms that mimic the process of natural selection, which consists of genetic operators, e.g. reproduction, mutation, selection or crossover. In standard GAs, each candidate solution (chromosome) of the population is represented by a fixed-length array of bits (genes), but other encodings are also possible. The evolution usually starts from a population of a randomly generated individuals. After each generation, the individuals are evolved by using genetic operators to achieve better fitness value. Compared to other optimisation methods like gradient based optimisation, GAs can avoid local optima. However, their computation cost is expensive.

Among the genetic operators, mutation and crossover are used to generate the next generation in the evolutionary process. In particular, for each new solution to be produced, a pair of "parent" solutions is selected for breeding. Mutation alters one or more gene values in a chromosome from its initial state to produce a new chromosome. An example of mutation operator is shown in Figure 2.1.

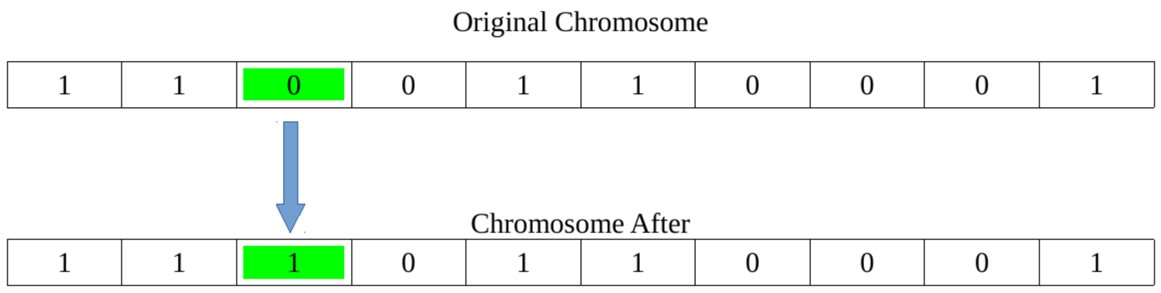


Figure 2.1: Example of mutation operator.

While mutation operators work on one parent solution, the crossover operator takes more than one parent solutions to produce a child solution. In crossover, the parent solutions exchange their genes to produce a new chromosome. There are many ways to do crossover including single point crossover, two point crossover or uniform crossover. An example of two point crossover is given in Figure 2.2. Firstly, two crossover points are selected. A binary string from beginning of chromosome to the first crossover point is copied from one parent. The part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent. In uniform crossover, the child's genes are randomly copied from either the first parent or the second parent.

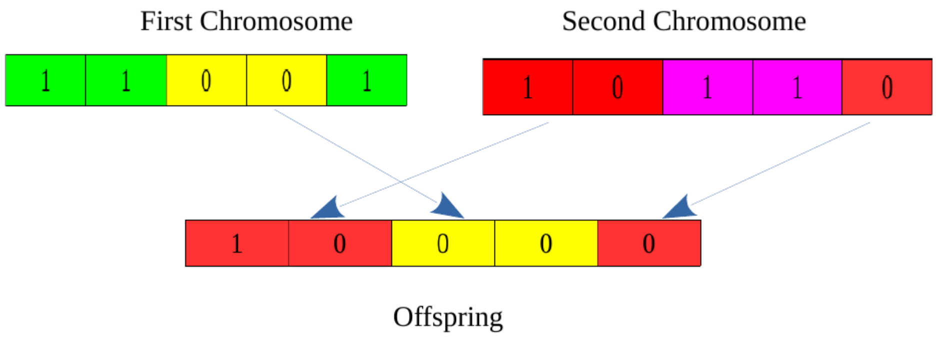


Figure 2.2: Example of two point crossover.

2.3.2 Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation (PSO) [19] is an evolutionary computation method, which is inspired by social behaviours such as bird flocking and fish schooling. In PSO, a problem is optimized by using a population, called swarm, of candidate solutions, which are called particles. In order to find the optimal solution, each particle moves around the search space by updating its position as well as velocity. Particularly, the current position of particle is represented by a vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D is the dimensionality of the search space. These positions are updated by using another vector, called velocity $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, which is limited by a predefined maximum velocity, v_{max} and $v_{id} \in [-v_{max}, v_{max}]$. During the search process, each particle maintains a record of the position of its previous best performance, called *pbest*. The best position of its neighbours is also recorded, which is called *gbest*. The position and velocity of each particle are updated according to the following equations:

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{i1} * (p_{id} - x_{id}^{t}) + c_2 * r_{i2} * (p_{gd} - x_{id}^{t}) \tag{2.1}$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{2.2}$$

where t denotes the t^{th} iteration in the search process, d is the d^{th} dimension in the search space, i is the index of particle, w is inertia weight, c_1 and c_2 are acceleration constants, r_{i1} and r_{i2} are random values uniformly distributed in [0,1], p_{id} and p_{gd} represent the position entry of *pbest* and *gbest* in the d^{th} dimension, respectively.

PSO was originally developed to address optimisation problems, which have continuous search spaces. Therefore, the representation for both position and velocity of a particle in PSO is a vector of real numbers. However, this representation is not suitable for many optimisation problems, which occur in a discrete search space. For example, feature selection problem has a discrete search, where features are either included or excluded in the solution. To address the discrete problem, in 1997 Kennedy and Eberhart [20] developed a binary particle swarm optimisation (BPSO). In BPSO, the position of each particle is a vector of binary numbers, which are restricted to either 0 or 1. Each entry v_i of the velocity represents the probability of the corresponding element in the position vector, p_i, taking value

1. The velocity of each particle is still updated by using Equation 2.1. A new equation 2.3, which uses a sigmoid function, is applied to update the particle position.

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

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

where $rand()$ is a random number selected from a uniform distribution in [0,1].

2.4 Related Work on Feature Selection

2.4.1 Traditional Feature Selection Methods

A basic version of feature selection is feature ranking [9], where a score is assigned to each feature according to an evaluation criterion. Feature selection can be performed by selecting the features with the highest scores. However, this type of algorithm ignores the interaction between features. Additionally, the features with the highest scores are usually similar. Therefore, these algorithms tend to selecting redundant features.

Sequential search techniques are also applied to solve feature selection problems. In particular, sequential forward selection (SFS) [39] and sequential backward selection (SBS) [26] are proposed. At each step of selection process, SFS (or SBS) adds (or removes) a feature from an empty (full) feature set. Although these local search techniques achieve better performance than the feature ranking method, they might suffer "nesting" problem, in which once a feature is added (or removed) from the feature set, it cannot be removed (or added) later. In order to avoid nesting effect, Stearns [35] proposed a "plus-*l*-takeaway-*r*" method in which SFS was applied *l* times forward and then SBS was applied for *r* back tracking steps. However, it is challenge to determine the best values of (*l*,*r*). This problem is addressed by sequential backward floating selection (SBFS) and sequential forward floating selection (SFFS), proposed by Pudil, et al.[31]. In SBFS ad SFFS, the values (*l*, *r*) are dynamically determined rather than being fixed in the "plus-*l*-takeaway-*r*" method.

2.4.2 EC Approaches(Non-PSO) for Feature Selection

EC techniques are well known because of their global search ability. EC algorithms have been applied to feature selection problems, such as GAs [47], GP [29]. Zhu et al.[47] proposed a hybrid feature selection approach, which combines both local search and GAs. In this algorithm, a filter method is used to rank features individually. Basing on the ranking information, GAs delete or add a feature to achieve better fitness value, which is the classification accuracy. The experiments showed that this algorithm outperforms the GAs alone and other algorithms.

Yuan, et al. [45] proposed a two-phase feature selection approach based on GAs. The proposed algorithm combined both filter and wrapper approach. Particularly, the first phase is a filter approach, which uses inconsistency criterion to remove irrelevant features. The second phase is a wrapper approach, which uses a feedforward neural network to remove the redundant features. The proposed algorithm intended to reduce the computation cost at wrapper step by reducing the size of feature set in the filter step. However, due to the the complex interaction between features, the first phase might remove features, which should be included in the best feature set.

Neshatian and Zhang [29] proposed a wrapper GP-based approach, which evaluates and rank feature subset in binary classification tasks. Experiments show that the proposed methods detected subset of relevant features in different situations, where other methods had difficulties.

2.4.3 PSO-based Feature Selection Methods

Many EC algorithms have been used for feature selection, such as GAs, GP or PSO. PSO is preferable because it is easier to implement and uses fewer parameters than GAs and GP. Two PSO based filter feature selection algorithms were proposed in [6], where mutual information and entropy are used in the fitness function to evaluate the relevance and redundancy of the selected feature subset. The experiments show that the proposed methods significantly reduce the number of features whilst achieve similar or better classification than using all features.

In PSO, premature convergence is a common problem, in which the swarm converges quickly to a local optima. To avoid premature convergence, Chuang, et al. [7] proposed a new *gbest* updating mechanism, which resets *gbest* elements to zero if it maintains the same value after several iterations. However, the performance of this algorithm is not compared with other PSO based algorithms. Another binary PSO based algorithm, which also aims to avoid premature convergence, is proposed by Bin et el. [4]. At each iteration of this algorithm, the swarm is divided into two groups, named "leaders" and "followers". The "leaders" have better fitness value. The "followers" update their positions and velocities based on "leaders"' update. The experimental results showed that the proposed update strategy better utilises the social behaviour phenomenon than the standard binary PSO. Another new *gbest* updating mechanism is developed by Xue, et al. [43], which regards not only classification accuracy but also the number of selected features. The proposed algorithms can increase the classification accuracy and simultaneously reduce both the number of selected features and the computation time.

Based on the statistical clustering method and PSO, Lane, et al. [22] proposed a feature selection approach, which selects one feature from each cluster. Although the number of features is significantly reduced to be the number of clusters, the classification accuracy is still improved. The results demonstrate that feature clusters, provided by the statistical clustering method , is useful information for feature selection. Therefore, this work will take the advantage of such information to further develop a new PSO based approach for feature selection.

2.5 Summary

Classification is an important task of machine learning, which aims to correctly assign instances with their correct label by building a classifier. During the training process, the classifier is trained on the training instances to achieve better performance. However, in many problems, the instances are described by a large number of features. Those features might be redundant or irrelevant, which not only cause a long training time ("curse of dimensionality") but also reduce the accuracy of the classifier. Feature selection is a technique that aims to remove those irrelevant or redundant features. However, due to the large search space and complex interaction between features, feature selection is considered a complex task.

Because of the global search ability, many EC techniques have been applied to solve feature selection problems. In which, PSO is widely chosen since it is easier to implement and computationally less expensive. However, there are some limitations in the existing PSO-based feature selection algorithms, which are summarised as follows.

- There is little research aims to utilise statistical clustering information in PSO to solve feature selection problems.
- Most of the existing PSO-based feature selection algorithms base on binary PSO, while it is shown that continuous PSO can achieve better performance than binary PSO [40].
- There is no research investigating the benefits of embedding genetic operators within the feature selection process.

Objective 1 and 2 aims towards addressing these limitations, through development of new continuous PSO-based feature selection approaches that use statistical clustering information and genetic operators.

Chapter 3

A New Representation in PSO for Feature Selection

This chapter addresses Objective 1, which is to develop a new representation in PSO for feature selection that can make use of the statistical clustering information for feature selection. Since the statistical clustering method groups similar features in one feature cluster, selecting a small number of representative features from each cluster is expected to provide the same or similar information to using all features in that cluster. Hence, the number of features is reduced while the classification performance would be maintained or even improved. Additionally, in most of current PSO-based feature selection methods, each feature is encoded as one dimension in search space, which results in a high dimensional search space. This chapter introduces a new representation, which has a lower dimensionality to utilise the statistical clustering information to select a small number of relevant features from the large original set of features.

3.1 Development of A New Representation for PSO-based Feature Selection Approaches

With standard PSO for feature selection, the dimensionality of each particle equals to the number of features, which is large in most situations. Due to the "curse of dimensionality", it might take a long time to select a good subset of features from such a high dimensional search space. In addition, in the original set of features, many features are similar, which can be considered redundant features. Those features are likely to be selected together since they have the same effects on the classification process. However, if those redundant features are selected, the classification performance will not be improved meanwhile the training time might be longer due to the large number of features. A new representation, based on statistical feature clustering, is introduced to address the redundancy problem as well as to improve the classification performance.

3.1.1 Feature Clustering Information

Removing redundant features is an essential task in feature selection problems. In the traditional representation, all features are treated equally which allows the presentation of redundant features in the solution. The statistical clustering algorithms proposed by Pledger and Arnold [30] and Matechou, et al. [27] are used to group features into different groups, called feature clusters. Features in the same cluster are considered similar and features in different clusters are dissimilar to each other. The technical detail of statistical clustering methods is not described here due to the page limit and the scope of this project.

The statistical clustering information can be utilised to reduce the chance of selecting redundant features. Particularly, in the new representation, all features which are in the same cluster, have to compete with each other to be selected. Each cluster possesses a finite number of entries in the position of each particle. Those entries are indices of the selected features from that cluster.

In the new representation, the length is determined according to the number of clusters and the total number of features. There are a limited number of entries, which correspond to a feature cluster to determine the selection of features from this cluster. In particular, the number of entries belonged to the j^{th} cluster is the maximum number of selected features from a cluster, which is called mSN_j. This important parameter will be defined in the later section. Being smaller than the number of features in a cluster is one of the conditions to select mSN_j, which is $mSN_j \leq N_j$, where N_j is the total number of features in the j^{th} cluster. By applying the clustering information, the new representation introduces a shorter position vector for each particle. In particular, the length of the new representation is $\sum\limits_j mSN_j$, which is smaller than $\sum\limits_j N_j$. On the other hand, $\sum\limits_j N_j$ is the total number of original features, which is also the length of the traditional representation in PSO for feature selection. So the new representation not only reduces the chance of selecting redundant features, but also shorters the position vector, which hopefully leads to shorter computation time.

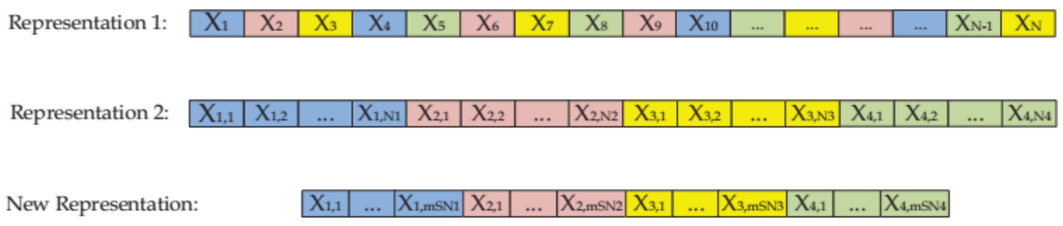


Figure 3.1: Example of N features that are grouped into 4 clusters with N_1, N_2,N_3 and N_4 features, respectively, then $N = N_1 + N_2 + N_3 + N_4$. mSN_j is the predefined maximum number of features selected from cluster j and $mSN_1 < N_1, \ldots, mSN_4 < N_4$

The comparison between the standard representation and the new representation is given in the Figures 3.1. Note that all features from the same cluster are represented by the same colour. Representation 1 shows the traditional way of using PSO for feature selection without considering the feature clustering information. Representation 2 considers the feature clustering information. Representation 2 is different from Representation 1 by putting features in the same cluster together, which is proposed by [23]. The new representation is shown in Representation 3, which is different from Representations 1 and 2 in two main aspects. Firstly, the dimensionality of the new representation is smaller than the two other representations. The second difference is the meaning of each element in the position vector. In Representations 1 and 2, each element (e.g. x_i or $x_{j,k}$) determines whether the corresponding feature is selected or not. In the new representation, each element shows which feature is selected from the corresponding cluster. The details are in next section.

3.1.2 Indexing Features in Feature Clusters

Feature clusters will be used as an input to get rid of the redundant features or irrelevant features in a new PSO-based algorithm. In order to allow a particle to refer to features within

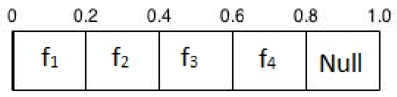


Figure 3.2: Interval for selection features (Not PSO positions)

a cluster, all features in the cluster need to be indexed. The chance of selecting a feature is affected by its index, the indexing method ensures fairness between features. This requirement is done by assigning sub-intervals with identical length to all features in the same cluster. In addition, a virtual feature, called "null" feature, is introduced to each cluster, which allows the selection of zero feature from a cluster, in case all features in that cluster are irrelevant or redundant.

Suppose the j^{th} cluster C_j contains N_j features, which is $C_j = \{f_1, f_2, \ldots, f_{N_j}\}$. A close interval [0,1] is split into $(N_j + 1)$ sub-intervals that are assigned to features within C_j by the following rule:

$$\begin{cases} \text{Feature } f_i \ : [\frac{i-1}{N_j+1}, \frac{i}{N_j+1}] \text{ where } i \in [1, N_j] \\ \text{Null Feature : } [\frac{N_j}{N_j+1}, 1] \end{cases} \tag{3.1}$$

So each feature is assigned to a sub-interval, whose length is $\frac{1}{N_j+1}$. For example, suppose a cluster C_j consists of 4 features, $C_j = \{f_1, f_2, f_3, f_4\}$, each feature is indexed as shown in Figure 3.2. As can be seen in Figure 3.2, the interval [0,1] is further divided into five intervals, where four of them correspond to the four features while the last interval corresponds to the *Null* feature, i.e. no feature is selected. Suppose that its $mSN_j = 2$ and the position values are $\{x_{1,1} = 0.5, x_{1,2} = 0.96\}$. As $x_{1,1} \in [0.4, 0.6]$, which is the interval of feature f_3, f_3 will be selected. Similarly $x_{1,2} \in [0.8, 1.0]$ that belongs to *Null* feature, which means that no feature is selected. So that entry values are interpreted as selecting only feature f_3 from the cluster. As can be seen from the above example, the use of "Null" feature allows the algorithm to choose less than mSN_j features, so mSN_j plays a role as an "upper limit"number of the selected features. Equation 3.2 shows a general case of how to determine which feature or no feature is selected from cluster j, where x is the position value in a dimension.

$$Feature = \begin{cases} f_k, \text{ if } x \in [\frac{k-1}{N_j+1}, \frac{k}{N_j+1}] \text{ where } k \in [1, N_j] \\ Null \text{ feature, if } x \in [\frac{N_j}{N_j+1}, 1] \end{cases} \tag{3.2}$$

3.1.3 Define Upper Limit: $\mathbf{mSN_j}$

The "upper limit", mSN_j, is an important parameter for this algorithm, which controls the number of selected features from each cluster. Basically, mSN_j needs to satisfy the following conditions:

1. $mSN_j \leq |cluster|$
2. mSN_j increases with respect to $|cluster|$, because the larger the cluster is, the more information it provides.

According to the above criteria, the upper limit number of selected features from each cluster is calculated by the following equation:

$$mSN_j = \sqrt{|cluster|} \tag{3.3}$$

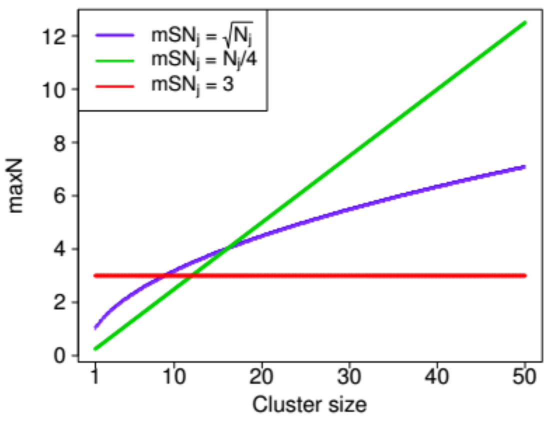


Figure 3.3: Three different ways of determining mSN_j

Figure 3.3 compares three different ways to determine mSN_j which are Equation 3.3, a constant function and a liner function, respectively plotted by using green, blue and red lines. As can be seen from the figure, Formula 3.3 allows selecting more features from a cluster that contains a large number of features, which cannot be done by the constant function (blue line). On the other hand, Equation 3.3 is preferred over linear scaling, since it defines a smaller number of selected features from large feature clusters, which likely consists of redundant features. The smaller mSN_j will reduce the chance of selecting those redundant features.

3.1.4 Fitness Function

In most of PSO based feature selection algorithms, the fitness function is set to the classification performance, which guides particles to achieve high classification accuracy. This fitness function's formula is given in Equation 3.4

$$Fitness_1 = ErrorRate \tag{3.4}$$

where

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN} \tag{3.5}$$

where TP, TN, FP and FN are short for true positive, true negative, false positive and false negative, respectively.

However, feature selection is a multi-objective problem, which aims to minimize the number of selected features while maximize the classification accuracy. In order to address this problem, a new fitness function is proposed in [41], which is shown in Equation 3.6

$$Fitness_2 = \alpha * \frac{\#Features}{\#AllFeatures} + (1-\alpha) * \frac{ErrorRate}{Error_o} \tag{3.6}$$

where $\alpha \in [0,1]$ shows the relative importance between the number of selected features and the classification error rate. $ErrorRate$ is the classification error rate obtained by the selected feature subset. $Error_o$ is the error rate obtained by using all original features. However, because $Error_o$ is not changed, introducing this value causes more computation. In addition, the fraction $\frac{ErrorRate}{Error_o}$ might be greater than 1, which is not fair because $\frac{\#Features}{\#AllFeatures} \in [0,1]$. Therefore, a similar fitness function (see Equation 3.7) is used in this work, which eliminates the $Error_o$ component.

$$Fitness_3 = \alpha * \frac{\#Features}{\#AllFeatures} + (1-\alpha) * ErrorRate \tag{3.7}$$

3.1.5 Summary of the Proposed Algorithm

Overall, the pseudo-code of the new algorithm with the new representation (PSOCC), which bases on continuous PSO based algorithm and feature clustering information, is shown in Algorithm 1.

Algorithm 1 : Pseudo-code of PSOCC

```
1:  begin
2:  indexing features in each cluster;
3:  define $mSN_j$ for each cluster according to Equation 3.3;
4:  randomly initialise the position and velocity of each particle;
5:  while Maximum iterations is not reached do
6:      evaluate the fitness of each particle according to its classification accuracy;
7:      for $i = 1$ to Population size do
8:          update pbest and gbest of particle $i$;
9:      end for
10:     for $i = 1$ to Population size do
11:         update $v_i$ of particle $i$ according to Equation 2.1;
12:         update $x_i$ of particle $i$ according to Equation 2.2;
13:     end for
14: end while
15: calculate the training and testing classification accuracy of the selected feature subset;
16: return the position of gbest, the training and testing classification accuracies;
17: end
```

3.2 Experimental Design

3.2.1 Benchmark Techniques

In order to examine the performance of the proposed representation, a traditional wrapper feature selection approaches (LFS [15]) and two PSO based feature selection algorithm (PSOFS [42] and PSO42 [43]) are used for comparison purposes in the experiments. The traditional algorithm, LFS, limits the number of features being considered during the forward selection process, which is quite similar to the upper limit number of selected features from each cluster. The algorithm PSOFS selects features by using standard continuous PSO. The other PSO-based algorithm, PSO42, introduces a new initialisation strategy and updating mechanism. Both PSOFS and PSO42 use the traditional representation.

Dataset	#features	#clusters	#classes	#instances
Wine	13	6	3	178
Vehicle	18	6	4	846
Ionosphere	34	11	2	351
Sonar	60	12	2	208
Musk1	166	14	2	476
Arrhythmia	279	15	16	452
Madelon	500	11	2	4400
Multiple Features	649	15	10	2000

Table 3.1: Datasets.

3.2.2 Datasets and Parameter Settings

Eight datasets (Table 3.1) chosen from the UCI machine learning repository [3] are used in the experiments. These datasets have different numbers of features, classes and instances. For each dataset, all instances are randomly divided into a training set and a test set, which contains 70% and 30% of the instances, respectively. Up to 500 training instances are used in the statistical clustering method to group features into different clusters, where the number of clusters are listed in the second column in Table 3.1. In the experiments, the classification/learning algorithm is K-nearest neighbour (KNN) where K = 5.
The parameters of PSO are set as follows [38]: $w = 0.7298$, $c_1 = c_2 = 1.49618$, $v_{max} = 6.0$, population size is 30, the maximum number of iterations is 100. The fully connected topology is used. The algorithm POCC is run 30 independent times on each dataset. A statistical significance test, Wilcoxon signed-rank test, is performed to compare between the algorithm's classification accuracies of different algorithms. The significance level of the Wilcoxon test was set as 0.05.

3.3 Experimental Results

Table 3.2 shows the experimental results of the PSOCC algorithm. In this table, "All" means that all available features are used, "Ave-size" shows the average number of selected features, "Best", "Ave-Test-Acc", "Std-Test-Acc" illustrate the best, average and standard deviation of the testing accuracies over the 30 independent runs. T represents the results of the significance tests between the testing accuracy of PSOCC and other algorithms. "+" or "-" means that the algorithm PSOCC achieved significantly better or worse classification performance than other algorithms, "=" means there is no significant difference between them.

As can be seen from Table 3.2, on the eight datasets, PSOCC achieves similar classification accuracy to all features on two datasets and significantly higher accuracy than all features on six datasets. Furthermore, PSOCC also selects a small feature subsets from the original one. On most of the datasets, the number of features is reduced by 70%. Especially, on the large datasets such as Madelon and Multiple Features, nearly 90% of the features are removed from the original feature sets meanwhile the classification accuracy is significantly increased. The results show that PSOCC successfully removes redundant/irrelevant features and simultaneously maintains or even improves the classification performance.

According to Table 3.2, comparing with PSOFS, PSOCC selects a smaller number of features. On five of the eight datasets, PSOCC achieves similar or higher classification accuracy than PSOFS. Although on two datasets (Vehicle and Multiple Features), the classification accuracy of PSOCC is lower than PSOFS, PSOCC selects fewer features than PSOFS. Especially, on the dataset Multiple Features, PSOCC only picks 51 features from the original 649 features, which is six times smaller than PSOFS. Meanwhile the classification accuracy is reduced only 0.16%. Comparing with PSO42, on most of the datasets, PSOCC selects a smaller number of features, and the classification performance is similar or even better. Only on two datasets (Sonar and Ionosphere), PSOCC selects a larger number of features, but PSOCC achieves significantly higher classification accuracy than PSO42. The results show that PSOCC with the new representation can use the clustering information to evolve better features subsets with a smaller number of features and similar or even higher classification accuracy than two existing PSO-based algorithms, PSOFS and PSO42.

Comparing between LFS and PSOCC, Table 3.2 shows that LFS selects a smaller number of features, but PSOCC achieves higher classification performance. The only exception is the dataset Musk1, where PSOCC achieves lower accuracy than LFS. But PSOCC's best

Dataset	Method	Ave-Size	Best	Ave-Test-Acc	Std-Test-Acc	T
Wine	All	13		76.54		+
	LFS	7		74.07		+
	PSOFS	7.93	98.77	95.6	1.7953	+
	PSO42	6.73	98.77	94.86	1.8628	+
	PSOCC	4.75	100	96.70	3.10	
Vehicle	All	18		83.86		+
	LFS	9		83.07		+
	PSOFS	9.5	87.01	85.03	0.8899	=
	PSO42	10.33	87.01	85.44	0.8372	-
	PSOCC	5.87	86.22	84.72	0.8720	
Ionosphere	All	34		83.81		+
	LFS	4		86.67		+
	PSOFS	12.47	93.33	88.41	2.3079	=
	PSO42	3.13	91.43	86.69	1.6444	+
	PSOCC	9.7	91.43	88.63	1.6765	
Sonar	All	60		76.19		+
	LFS	3		77.78		+
	PSOFS	26.1	84.13	77.3	3.5765	+
	PSO42	11.23	84.13	77.94	3.2104	=
	PSOCC	14.33	84.13	78.94	4.0185	
Musk1	All	166		83.92		=
	LFS	10		85.31		-
	PSOFS	85.93	88.81	84.61	2.0568	=
	PSO42	77.3	89.51	84.87	2.7042	=
	PSOCC	35.03	90.21	83.12	3.4196	
Arrhythmia	All	279		94.46		+
	LFS	11		94.46		+
	PSOFS	118.73	95.14	94.56	0.3517	=
	PSO42	69.77	95.59	94.77	0.4495	+
	PSOCC	44.17	95.59	94.96	0.38	
Madelon	All	500		70.9		+
	LFS	7		64.62		+
	PSOFS	259.07	78.97	76.35	1.0909	+
	PSO42	206.57	84.23	78.81	3.1171	+
	PSOCC	54.39	85.13	83.40	2.0368	
Multiple features	All	649		98.63		+
	LFS	18		99.0		+
	PSOFS	297.07	99.2	99.0	0.0934	-
	PSO42	314.5	99.2	99.0	0.0935	-
	PSOCC	51.07	99.23	98.84	0.1751	

Table 3.2: Experimental Results of PSOCC.

solution achieves higher accuracy than LFS. The results show that PSOCC with the new representation can better explore the search space to find the better subsets than the traditional method, LFS.

3.4 Summary

The goal of Objective 1 was to develop a new PSO representation, which utilises the feature clustering information to evolve a small subset of features that maintains or improves the classification accuracy. This goal has been achieved by introducing the upper limited number of selected features and indexing features method, which allowed to encode clustering information inside a particle's position. The algorithm with the new representation was compared with two PSO-based algorithms (PSOFS and PSO42) and a traditional methods (LFS). The results showed that the new algorithm successfully reduced the number of features and simultaneously maintained or increased the classification accuracy.

From the results in Table 3.2, it can be seen that for some datasets like Musk1 and Multiple Features, PSOCC selects a small number of features, which results relatively in poor classification accuracy. Although the upper limited number of features decreases the computation time, it also reduces the chance to select a good feature subset. For example, in case all features in a cluster are important, but the upper limited number of selected features will not allow selecting all features from that cluster. In the following chapter, an improved version of PSOCC is proposed, which uses Gaussian distribution to better explore the search space to further improve the performance.

Chapter 4

Applying Gaussian Distribution in PSO for Feature Selection

In Chapter 3, a new representation is proposed in PSO for feature selection to utilise the statistical clustering information to reduce the number of features selected and increase the classification performance. In that representation scheme, each element in the position vector indicates which feature is selected from its corresponding cluster. In particular, suppose a feature F in a cluster is assigned with an interval $[a, b]$, where $0 \leq a < b \leq 1$. If the element in the position vector has value c, which falls into the interval $[a, b]$, the feature F will be selected. This transformation rule introduces a limitation to the new representation. For instance, suppose that in two different particles p_1, p_2, the element's values are c_1 and c_2 respectively, where $a < c_1 < c_2 < b$. According to the transformation rule, both p_1 and p_2 selects the element's corresponding feature. Although the elements in p_1 and p_2 are assigned to two different values, the corresponding selected features are identical. This significantly affects on the search ability of PSO. Particularly, a small change of an element within the position vector might not lead to any change in the selected features as well as the fitness value. The purpose of using continuous PSO in the new representation scheme is to utilise the smooth movement of particle in the search space. However, the transformation rule accidentally diminishes smooth property of the search space. This chapter aims to solve this limitation by adding more meaning to the element of a position vector and creating a new transformation rule, which allows particles to move smoother in the search space. The new transformation rule is expected to make the fitness landscape smooth and achieve higher classification accuracy than the representation developed in Chapter 3.

4.1 Development of New Representation Schema in Continuous PSO

4.1.1 Gaussian Distribution for Feature Selection

Before introducing a new meaning of entries in the position vector, it is worth to have a short review on the new representation proposed in Chapter 3. In the new representation, the position value in a dimension determines which feature is selected from a certain cluster. In other word, the position value plays a role as the index of a feature in a cluster. For the cluster i that contains N_i features, an interval [0, 1] is equally divided into $(N_i + 1)$ sub-intervals. The length of a sub-interval is called step. Therefore, the step of a cluster can be calculated by the Equation 4.1.

$$step_i = \frac{1}{N_i + 1} \tag{4.1}$$

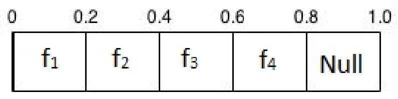


Figure 4.1: Indexing features

where N_i is total number of features in the i^{th} cluster. If the entry value falls into a feature's interval, this feature is selected. Although, the position value in a dimension could vary in the interval [0, 1], its meaning is still an index of a feature. Each position entry's value has exactly one corresponding feature. Suppose the i^{th} feature f_i in the j^{th} cluster is assigned to the interval [0.2,0.4]. Two different position entry's values 0.25 and 0.35 have exactly same interpretation, which is "feature f_i is selected". Therefore, when the position value is changed from 0.25 to 0.35, the set of selected feature is not changed. So even the search space is in continuous form, the particle does not move smooth.

In order to solve the above limitation, the position entry value is used to calculate the probability that a feature being selected. To achieve this, a Gaussian distribution is introduced in the interpreting process. This Gaussian distribution will determine the probability of a feature being selected within a certain cluster. The determining rule is shown later in the following section. By using the Gaussian distribution, all features in the cluster have chance to be selected, while in Chapter 3, only the corresponding feature will be selected. The parameters of the Gaussian distribution are defined, so that the corresponding feature has the biggest chance to be selected. The far from the corresponding feature a feature is, the less chance that feature is selected. Therefore, the position value is chosen as a mean for the Gaussian distribution, which ensures that the selection of the corresponding feature is given a higher probability than the selection of other features with in the cluster.

The standard deviation of the Gaussian distribution is set to $0.1 * step$, where $step$ is calculated using the Equation 4.1. This standard deviation ensures the probability that the corresponding feature being selected is more than 99%. The rest 1% is distributed to other features with respect to how far the feature is from the corresponding feature.

An example of applying Gaussian distribution on the new representation is outlined below. In this example, suppose a cluster contains 4 features f_1, f_2, f_3, f_4. Therefore, an interval [0,1] is divided into 5 sub-intervals, which are respectively assigned to 4 features as being shown in the Figure 4.1. The step value is $step = \frac{1}{4+1} = 0.2$.

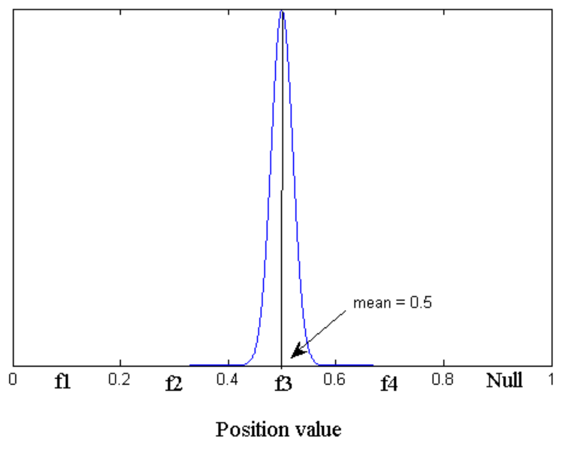


Figure 4.2: The Gaussian distribution when position value is 0.5.

Suppose that the position value of a dimension is 0.5, the corresponding feature is f_3.

The Gaussian distribution with this position value is shown in Figure 4.2. As can be seen from the figure, it is about more than 99% that feature f_3 is selected. In addition, f_2 and f_4 have more chance to be selected than f_1 or $Null$ (i.e. no feature is selected), because they are closer to f_3.

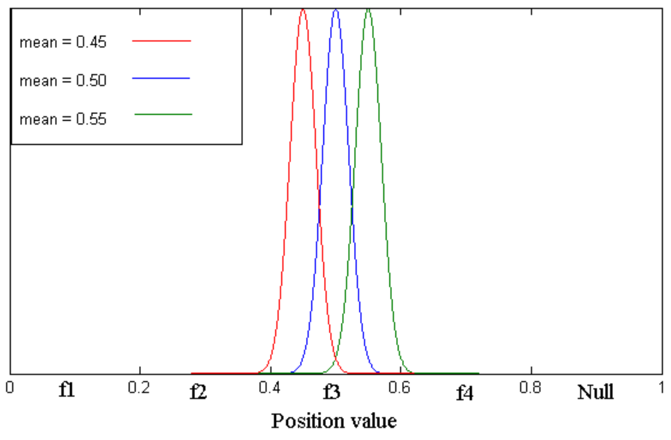


Figure 4.3: The differences introduces by different position values.

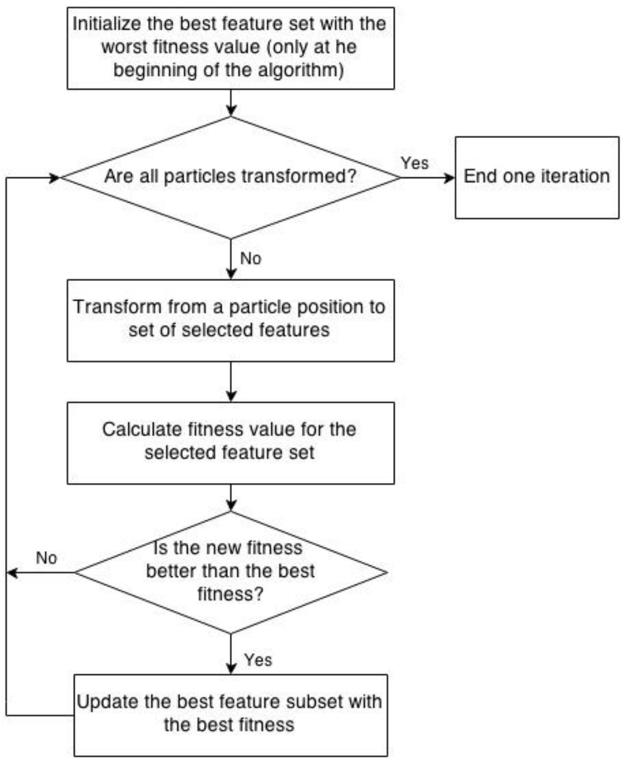
Since the position value relates to the probability that a feature is selected, a change in position entry results in the different probabilities. To better illustrate this assertion, the above example is reused, in which the position entry takes 3 different values 0.45, 0.5 and 0.55. According to the transformation rule in Chapter 3, these values indicate the same meaning, that is "feature f_3 is selected". However, by using Gaussian Distribution, these different values have different meanings. The differences are shown in Figure 4.3. At the beginning, the position value is 0.5, which is also a mean of the Gaussian Distribution. As can be seen, the chances that f_2 and f_4 being selected are equal. If the position value is updated to 0.55, which is still in the interval of feature f_3. The Gaussian distribution moves toward feature f_4. Therefore the feature f_4 is more likely to be selected than feature f_2. On the contrary, the chance of feature f_2 being selected is more than feature f_4 when the position value is updated to 0.45. Therefore, although these three different values are in the same feature's interval, they introduce different probability that features within the cluster being selected.

4.1.2 How to Select Features

Introducing the Gaussian distribution makes the position value has more meaning than just an index of selected feature in a cluster. This section discusses how to translate from the position value to the selected feature with Gaussian distribution. Firstly, a Gaussian distribution is built by using the position value as its mean and $0.1 * step$ as its standard deviation. After that, a random number is generated from the distribution. Notice that this number should fall into the range [0,1]. That is why the standard deviation is calculated as $0.1 * step$, which is small enough to ensure that the random number is in the interval [0,1] in most cases. This random number is used to determine which feature is selected from a certain cluster. The selection rule is similar to the selection rule in PSOCC, except that the random number is used instead of the position value. The chance that the random number falls into a feature's interval can be viewed as the probability that feature is selected.

Algorithm 2 : Pseudo-code of GPSOCC

```
1: begin
2: indexing features in each cluster;
3: define $N_{sc}$ for each cluster according to Equation 3.3
4: initialize the best feature set ($BFS$) with the worst fitness.
5: randomly initialise the position and velocity of each particle;
6: while Maximum iterations is not reached do
7:     for each particle in the swarm do
8:         transform from the position to a set of selected features
9:         evaluate the fitness based on the selected features
10:        if the fitness is better than $BFS$'s fitness then
11:            update $BFS$
12:        end if
13:    end for
14:    for $i = 1$ to Population size do
15:        update pbest and gbest of particle $i$;
16:    end for
17:    for $i = 1$ to Population size do
18:        update $v_i$ of particle $i$ according to Equation 2.1;
19:        update $x_i$ of particle $i$ according to Equation 2.2;
20:    end for
21: end while
22: calculate the training and testing classification accuracy using $BFS$
23: return $BFS$, the training and testing classification accuracies;
24: end
```



Initialize the best feature set with the worst fitness value (only at he beginning of the algorithm)
Are all particles transformed?
Yes
End one iteration
No
Transform from a particle position to set of selected features
Calculate fitness value for the selected feature set
Is the new fitness better than the best fitness?
No
Yes
Update the best feature subset with the best fitness

Figure 4.4: Update the best feature set in one iteration.

Since the transformation process from the position to the selected future set uses a Gaussian distribution, the future set is not deterministic. In PSO, *gbest* is the final solution, but the same position might result in different feature sets at different times. Therefore, it is necessary to keep track of the best selected feature set, which is used as a solution at the end of each run. This selected set is updated during the fitness calculation process. The updating process is shown in Figure 4.4.

4.1.3 Fitness Function

The fitness function used in GPSOCC is the same fitness function as PSOCC, which is calculated according to Equation 3.7.

4.1.4 Summary of the Proposed Algorithm

The pseudo-code of the proposed algorithm, GPSOCC, which combines both the Gaussian distribution and the statistical clustering information, is shown in Algorithm 2.

4.2 Experimental Design

To examine the performance of the proposed algorithm, GPSOCC, a set of experiments have been conducted. The experimental design is the same as in Chapter 3. In addition, the performance of GPSOCC is compared with PSOCC, which is the proposed algorithm in Chapter 3 and a binary PSO based feature selection algorithm, GPSO1 [23], which also uses statistical feature clustering information and Gaussian distribution.

4.3 Experimental Results

Table 4.1 shows the experimental results of the GPSOCC algorithm, where "All" means that all the available features are used for classification. "Ave-size" shows the average number of selected features over the 30 runs. "Ave-Train", "Std-Train", "Ave-Test", "Std-Test" illustrate the average and standard deviation of the training and testing accuracies over the 30 independent runs. T shows the results of the statistical significant tests between the accuracy of GPSOCC and other algorithms. "+" or "-" means that the algorithm GPSOCC achieved significantly better or worse classification performance than other algorithms, "=" means there is no significant difference between them.

From Table 4.1 , it can be seen that the number of features selected by GPSOCC is much smaller than the total number of features, but using the selected features only, the 5NN classification algorithm achieved significantly better or similar classification accuracy to using all features. For example, on the Madelon dataset, GPSOCC selects on average 51 features from the original 500 features, but achieves a significant increase in classification accuracy of 14%. The results suggest that GPSOCC can be successfully used for feature selection to reduce the dimensionality of the data and significantly increase the classification performance over using all features.

In comparison with PSOCC, GPSOCC achieves a higher classification performance in three datasets and has similar performance on four others. The only dataset that the PSOCC outperforms GPSOCC is the Ionosphere datatset. However, the number of features selected by GPSOCC is about three times smaller than the number of features selected by PSOCC. In addition, on all datasets, GPSOCC achieves higher training accuracy than PSOCC. For example, on the Madelon dataset, GPSOCC selects less features than PSOCC but GPSOCC

Dataset	Method	Ave-Size	Ave-Train ± Std-Train	Ave-Test ± Std-Test	T
Wine	All	13		76.54	+
	GPSO1	5.4	96.71 ± 7.77E-14	96.59 ± 2.76	+
	PSOCC	4.75	95.05 ± 0.58	96.70 ± 3.10	+
	GPSOCC	4.60	97.37 ± 0.42	97.70 ± 2.52	
Vehicle	All	18		83.86	+
	GPSO1	8.94	86.11 ± 0.20	84.30 ± 0.62	+
	PSOCC	5.87	84.61 ± 0.56	84.72 ± 0.87	=
	GPSOCC	7.30	90.10 ± 0.40	84.74 ± 0.49	
Ionosphere	All	34		83.81	+
	GPSO1	7.66	91.59 ± 0.47	89.50 ± 1.68	-
	PSOCC	9.7	90.04 ± 0.99	88.63 ± 1.68	-
	GPSOCC	3.17	93.90 ± 0.67	86.89 ± 1.8	
Sonar	All	60		76.19	+
	GPSO1	17.64	86.74 ± 0.94	78.19 ± 4.14	=
	PSOCC	14.33	87.01 ± 2.00	78.94 ± 4.02	=
	GPSOCC	10.17	90.67 ± 1.6	78.25 ± 2.96	
Musk1	All	166		83.92	+
	GPSO1	39.64	90.02 ± 0.60	84.95 ± 2.73	-
	PSOCC	35.03	89.78 ± 1.25	83.12 ± 3.41	=
	GPSOCC	38.93	93.22 ± 1.37	83.29 ± 2.48	
Arrhythmia	All	279		94.46	+
	GPSO1	45.5	94.87 ± 0.09	94.85 ± 0.34	+
	PSOCC	44.17	95.11 ± 0.20	94.96 ± 0.38	+
	GPSOCC	42.03	95.75 ± 0.18	95.12 ± 0.34	
Madelon	All	500		70.9	+
	GPSO1	36.08	85.45 ± 0.73	85.68 ± 1.10	-
	PSOCC	54.39	83.73 ± 1.74	83.40 ± 2.00	+
	GPSOCC	51.17	89.20 ± 1.41	84.06 ± 1.65	
Multiple features	All	649		98.63	+
	GPSO1	91.4	99.38 ± 0.38	99.01 ± 0.13	=
	PSOCC	51.07	99.17 ± 0.09	98.84 ± 0.18	=
	GPSOCC	51	99.36 ± 0.07	98.86 ± 0.17	

Table 4.1: Experimental Results of GPSOCC

achieves higher training accuracy, 89.2%, which is about 6% better than PSOCC's training accuracy. The results suggests that the Gaussian distribution can help PSO to better explore the continuous search space by applying smooth movement for each particle.

Comparing GPSOCC with GPSO1, GPSOCC selects fewer features than GPSO1 on most of datasets, except on the Madelon dataset. In terms of testing accuracy, GPSOCC achieves higher classification performance on 3 datasets and has similar performance on two other datasets. On all datasets, GPSOCC'ss training accuracies are much better than GPSO1's training accuracy, which illustrates that GPSOCC explores the search space better than GPSO1.

As can be seen from experimental results, in terms of training accuracy, on all datasets, GPSOCC achieves better performance than both PSOCC and GPSO1. The results show that introducing the Gaussian distribution can guide the particle better explore the continuous space. In addition, it is an evidence to show that continuous PSO is better than binary PSO. However, on most of datasets, the testing accuracy of GPSOCC is similar to GPSO1. This is an indication of overfitting problem.

4.4 Summary

The goal of this chapter was to further improve the new PSO representation, proposed in Chapter 3. The goal is successfully achieved by introducing a Gaussian distribution, which uses the position values as its mean to determine the probability of each feature in a certain cluster being selected. A number of experiments have been conducted to compare the new algorithm GPSOCC with its predecessor, PSOCC and GPSO1, which also uses statistical cluster information and Gaussian distribution. The results show that by using the Gaussian distribution, GPSOCC can explore the search space better than two other PSO algorithms. However, on some datasets, GPSOCC sacrifices its classification accuracy to achieve smaller set of selected features. In the following chapter, a new updating mechanism with genetic operators (crossover and mutation) is developed to help PSO explore the search space better.

Chapter 5

A Hybrid PSO for Feature Selection

In recent years, several evolutionary algorithms have been developed for feature selection problems in which Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) are the two most popular algorithms. GA [13] is a heuristic search technique that mimics the process of natural evolution, which includes some pseudo-biological operators such as inheritance, mutation, selection and crossover. In term of searching ability, GA is very good at exploring the entire search space because of the pseudo-biological operators. However, GA is not really good at precisely locate local optimal solution in the search region. Another drawback of GA is its expensive computation cost.

Some comparisons between GA and PSO have been done in [10, 5, 17], which show that PSO usually achieves the same performance as GA but with much cheaper computation cost. However, premature convergence is a drawback of PSO in which the entire population is trapped in local optima. This problem appears frequently in solving high dimensional problems such as feature selection, whose search space contains too many local optima. Some studies have been done to overcome this limitation. One way is increasing the exploration ability by applying dynamic coefficient, which is proposed in [36, 32]. Another solution is introduced in [8], which resets *gbest* if the *gbest* value does not change after three iterations. In [25], a multi-swarm PSO is proposed to avoid premature convergence problem, in which the population is split into many small sized sub-swarms to achieve better performance. Every fixed number of generations, the population is grouped randomly to exchange the information as well as to prevent early convergence inside each sub-swarm. Some hybrid techniques, which combine GA and PSO, have been done with low level of integration. For example, in [33], the combination of GA and PSO is done by using the result of one algorithm as an input of the other algorithm. Another stronger co-operation of GA and PSO is proposed by Gandelli [11], where the integration is done during the entire run. Particularly, the whole population is split into two sub-populations, which are evolved by GA and PSO respectively. After that, these sub-populations are merged to exchange their achievement before being randomly split again in the next iterations.

In this chapter, a new PSO based feature selection algorithm, which integrates two genetic operators: crossover and mutation in the updating process, is developed. The proposed approach is examined and compared with the original PSO and another PSO based feature selection proposed by Lane [23] to investigate whether the new approach can better explore the search space to avoid the premature convergence problem to improve the performance.

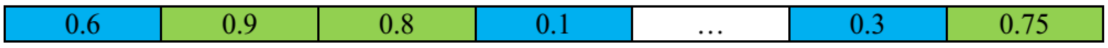


Figure 5.1: Standard PSO Representation.

5.1 Continuous PSO for Feature Selection

5.1.1 Particle's Representation

In this chapter, a continuous PSO is applied to solve feature selection problems, where each position value in the position vector is a real number and corresponds to a feature from the original feature set. In particular, each particle is represented by a vector of real numbers, $x_i = (x_{i1}, x_{i2}, \ldots, x_{iN})$ where N is the total number of features. $0 \leq x_{in} \leq 1$ shows the probability of the n^{th} feature being selected. A threshold θ is introduced to determine whether or not a feature is selected. Particularly, if $x_{in} \geq \theta$, the n^{th} feature is selected. Otherwise, the n^{th} feature is not selected. So θ is an important parameter which can be used to control the number of selected features. The higher θ is, the less chance a feature being selected. According to initial experience, 0.7 is a good value for θ, which balances between the number of selected features and the classification accuracy. The visualisation of particle's representation can be shown in Figure 5.1, where the green entries indicate the selected features and the blue ones correspond to the ones which are not selected.

5.1.2 Fitness function

The fitness function used in the proposed algorithm is the same fitness function as PSOCC, which is a combination of error rate and ratio between the number of selected features and the total number of features. Formula of the fitness function is shown in Equation 3.7.

5.2 Integrating Crossover and Mutation into PSO

5.2.1 Crossover

In order to avoid premature convergence, the crossover operator is performed between a number of particle pairs in each iteration. A roulette wheel selection based on the fitness value is used to select particles, which then are used as the parents of the crossover operator. A uniform crossover is applied to the selected particles to derive a new particle, called a child. This child's fitness value is calculated and compared with the *pbest*. If its fitness is better than the parent's current fitness value, this child will replace the parent in the swarm. If the child's fitness is even better than the parent's personal best fitness, the parent's personal best position is then replaced by the child's position. The improvement of those parents is then propagated through the swarm during the sharing process.

Since the particles within the swarm will be more similar near the end of the run, the crossover will have more impact at the beginning, when the population is just randomly initialized. Therefore, the number of particles , which is used to apply the crossover operation, is reduced with respect to the increase of the number of iterations (see Equation 5.1). By doing so, the computation cost will be reduced, while the crossover performance is maintained.

$$P_i = \lfloor N - i * \frac{N-2}{I} \rfloor \tag{5.1}$$

where P_i is the number of selected particles for crossover at i^{th} iteration, I is the total number of iterations.

The pseudo-code of crossover in each iteration is shown in Algorithm 3

Algorithm 3 : Pseudo-code of Crossver in iteration i^{th}

```
1: begin
2: calculate $P_i$ according to Equation 5.1
3: $noCrossover = \lfloor \frac{P_i}{2} \rfloor$
4: while noCrossover is not reached do
5:     select a pair of particles as parents;
6:     perform uniform crossover between these parents to get a child;
7:     calculate child's fitness value;
8:     if child's fitness is better than parent's current fitness then
9:         parent's current position $\leftarrow$ child's position
10:        parent's current fitness $\leftarrow$ child's fitness
11:    end if
12:    if child's fitness is better than parent's personal fitness then
13:        parent's personal best position $\leftarrow$ child's position
14:        parent's personal best fitness $\leftarrow$ child's fitness
15:    end if
16:    remove the two particles from candidate parents list
17: end while
18: end
```

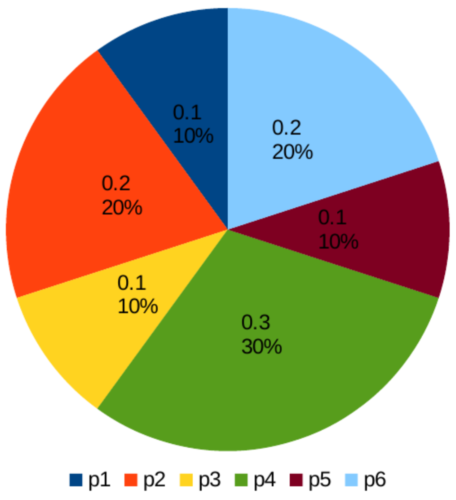
An example of typical run-through of selecting particles for crossover operator is outlined below. In this example, suppose the swarm contains 6 particles $\{p_1, p_2, p_3, p_4, p_5, p_6\}$ with these respective fitness values $\{0.1, 0.2, 0.1, 0.3, 0.1, 0.2\}$. We are going to do the crossover operation 2 times, which means 4 particles being selected. So according to Algorithm 3, $P_i = 4$ and $noCrossover = 2$. The process of performing crossover are shown as belows.

- Step 1: The probabilities of selecting each particle is shown in Figure 5.2a. Two particles are chosen via a roulette wheel selection. Since we aim to minimise the fitness value, the lower a particle's fitness value is, the better that particle is. As can be seen from the figure, the particle, which has higher fitness value than the others, has more chance to be chosen. Therefore, the worse particle will be more likely to be improved via the crossover operation. In this example, we assume that the roulette wheel selection determines that the particle 1 and particle 6 are selected. This step is the first iteration in Algorithm 3. After that, the process inside the loop of Algorithm 3 is applied to do crossover operation between particle 1 and 6.

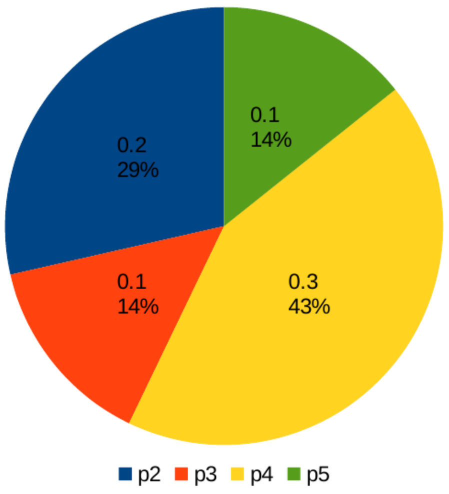
- Step 2: After being chosen, particles 1 and 6 are removed from the wheel, so other particles will have chance to improve its fitness via crossover operation. This step corresponds to the second iteration of Algorithm 3. In this iteration, another two particles are chosen as parents of the crossover operation.

5.2.2 Mutation

Although the crossover operation can be used to better explore the search space, it has less impact near the end of run because particles in the swarm become very similar. Opposite to crossover, mutation operation has less impact at the beginning of a run, and more near the end [10]. Therefore, integrating a mutation operation into the PSO algorithm would also improve the exploration ability.



(a) The probability of a particle being selected to be a parent of the crossover operation



(b) The probability of a particle being selected to be a parent of the crossover operation after feature p_1 and p_6 being selected

Figure 5.2: Selecting parents for the crossover operation basing on fitness value

Each particle within the swarm records a global best position i.e. *gbest*, which is the best position so far being discovered by the particle and its neighbours. If the *gbest*'s fitness value is not improved after a number of iterations, the particle is probably trapped in local optima. This is when a mutation operation needs to be applied on the *gbest* position.

Belief in a feature

The global best is represented by a vector of real numbers, which are continuous values in the interval [0,1]. Suppose the *gbest*'s position is encoded as $(g_{i1}, g_{i2}, \ldots, g_{iN})$. These values not only indicate which features are selected but also show the "belief" of that feature. For example, $g_{i1} = 0.8$ and $g_{i2} = 0.9$ shows that both feature 1 and feature 2 are selected, since both values are greater than the threshold $\theta = 0.7$. However, it is believed that feature 2 is more deserved to be selected than feature 1, because g_{i2} is far from θ than g_{i1}. Similarly, if $g_{i2} < g_{i1} < \theta$ then the second feature is more likely not being selected than the first one. In other word, the further distance between the feature's value and the threshold, the more belief that feature is selected or not selected.

Mutation on *gbest*

Taking the idea about belief in a feature, a new mutation method is proposed for *gbest* position, where the less confident feature is more likely to be mutated. Firstly, the confident rate (CR) of each feature is calculated based on the distance from its value to the threshold θ, which is shown in Equation 5.2

$$CR_i = \begin{cases} \frac{f_i-\theta}{1-\theta} & \text{if } f_i > \theta \\ \frac{\theta-f_i}{\theta} & \text{if } f_i \leq \theta \end{cases} \tag{5.2}$$

where CR_i is the confident rate of i^{th} feature, f_i is the i^{th} feature's value in *gbest*.

A temporary position, called a child, is generated by applying mutation on the *gbest* position. After calculating the confident rate for a feature, a random number r is generated. If $r < CR_i$, then the i_{th} child's position entry is set to the corresponding entry in *gbest*.

Otherwise, the i_{th} child's position entry is a mutated value of *gbest*'s corresponding entry, which is determined by Equation 5.3.

$$child_i = 1 - gbest_i \tag{5.3}$$

where $child_i$ is the i^{th} position entry value of the *child*, $gbest_i$ is the corresponding entry value in *gbest*.

It can be seen that if a feature is selected, it will not be selected after being mutated and vice versa. More important, the confident rate CR plays a role in the mutation rate. The higher the CR is, the lower chance that its corresponding feature being mutated.

Mutation example

Suppose *gbest* fitness value of a particle p is not changed over 3 iterations, we are going to mutate this *gbest*. Assume that:

- The threshold is θ = 0.7
- The total number of features is 4
- The *gbest* position is [0.14, 0.76, 0.91, 0.21]

According to the Equation 5.2, the confident rate of each features can be calculated as below:

- $CR_1 = \frac{0.7-0.14}{0.7} = 0.8$
- $CR_2 = \frac{0.76-0.7}{1-0.7} = 0.2$
- $CR_3 = \frac{0.91-0.7}{1-0.7} = 0.7$
- $CR_4 = \frac{0.7-0.21}{0.7} = 0.7$

Suppose the first random number is generated, $r_1 = 0.45$, which is smaller than CR_1. So the child's first position entry, which corresponds to the first feature, is set to $gbest_1 = 0.14$. Similarly, another random numbers are generated for the other features to fully build the child, which is shown as below:

- $r_2 = 0.3 > CR_2$, so $child_2 = 1 - gbest_2 = 1 - 0.76 = 0.24$
- $r_3 = 0.65 < CR_3$, so $child_3 = gbest_3 = 0.91$
- $r_4 = 0.75 > CR_4$, so $child_4 = 1 - gbes_4 = 1 - 0.21 = 0.79$

The mutated child's position is [0.14, 0.24, 0.91, 0.79]

The pseudo-code of mutation in each iteration is shown in Algorithm 4

5.2.3 Overall Algorithm

The pseudo-code of the proposed algorithm, Crossover-Mutation PSO (CMPSO), which embeds genetic operators into a continuous PSO, is given in Algorithm 5.

Algorithm 4 : Pseudo-code of Mutation

```
1: begin
2: for each particle $p_i$ in the swarm do
3:     if $p_i$'s global fitness is not improved for 3 iterations then
4:         child ← $p_i$'s gbest
5:         for $i = 1$ to gbest's size (number of features) do
6:             calculate $CR_i$ of the $i^{th}$ feature according to Equation 5.2;
7:             generate a random number $r$
8:             if $r \geq CR_i$ then $child_i = 1 - gbest_i$
9:             end if
10:            if child's fitness is better than $p_i$'s current fitness then
11:                $p_i$'s current position ← child's position
12:                $p_i$'s current fitness ← child's fitness
13:            end if
14:            if child's fitness is better than $p_i$'s personal fitness then
15:                $p_i$'s personal best position ← child's position
16:                $p_i$'s personal best fitness ← child's fitness
17:            end if
18:            if child's fitness is better than $p_i$'s global fitness then
19:                $p_i$'s global best position ← child's position
20:                $p_i$'s global best fitness ← child's fitness
21:            end if
22:        end for
23:    end if
24: end for
25: end
```

Algorithm 5 : Pesudo-code of CMPSO

```
1: begin
2: randomly initialise the position and velocity of each particle;
3: while Maximum iteration is not reached do
4:     evaluate the fitness of each particle;
5:     for $i = 1$ to $PopulationSize$ do
6:         update the pbest of particle $i$;
7:     end for
8:     perform crossover operation on the swarm;
9:     update the gbest /*Ring topology*/;
10:    perform mutation operation on the swarm;
11:    for $i = 1$ to $PopulationSize$ do
12:        Update velocity of particle $i$;
13:        Update position of particle $i$;
14:    end for
15: end while
16: end
```

5.3 Experimental Design

5.3.1 Benchmark Techniques

To examine the performance of the proposed algorithm (CMPSO), a binary PSO based feature selection algorithms, GPSO1[23]), and the PSOCC algorithm (Chapter 3) are used as benchmark techniques in the experiment. In addition, to further analysis the evolutionary process, CMPSO is also compared with the original continuous PSO (PSO).

5.3.2 Datasets and Parameter Settings

Eight datasets (Table 3.1) chosen from the UCI machine learning repository are used in the experiments. Those datasets are the same ones, which are used in GPSO1 and PSOCC to ensure fair comparisons.

The parameters of the PSO algorithms are set as follow: $w = 0.7298, c_1 = c_2 = 1.49618$, $v_{max} = 0.2$, the population size is 30, the maximum iteration is 100 and the threshold θ is set as 0.7. The same parameters are set for GPSO1, except the maximum velocity v_{max} is set to 6.0.

5.4 Experimental Results

This section firstly discusses the performance of CMPSO and the other three PSO based feature selection algorithms (Table 5.1), then compares the search ability between CMPSO and PSO.

5.4.1 CMPSO versus GPSO1 and CPSO

Table 5.1 shows the experimental results of the CMPSO algorithm, where "All" means that all the available features are used for classification. "Ave-size" shows the average number of selected features over the 30 runs. "Ave-Train", "Std-Train", "Ave-Test", "Std-Test" illustrate the average and standard deviation of the training and testing accuracies over the 30 independent runs. T shows the results of the statistical significant tests between the accuracy of CMPSO and other algorithms. "+" or "-" means that the algorithm CMPSO achieved significantly better or worse classification performance than other algorithms, "=" means there is no significant difference between them.

As can be seen from Table 5.1, on all datasets, CMPSO successfully selects feature subsets, which achieve significantly higher classification accuracy than using all features. In addition, on all dataset, the number of selected features is always less than a third of the total number of features.

Compared to the PSOCC algorithm, CMPSO also achieves similar or higher classification accuracy, while the number of selected features remains similar. It is remarkable that on all datasets, CMPSO achieves much higher training accuracy than PSOCC. Especially, on Vehicle dataset, CMPSO's training accuracy is about 5.27% higher than PSOCC's one.

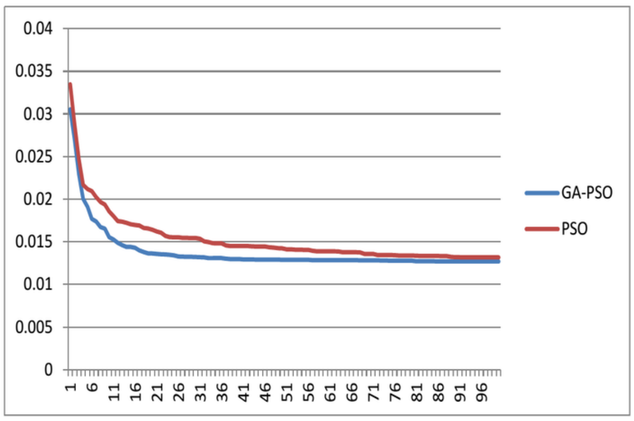
Comparing CMPSO with GPSOCC, in terms of testing accuracy, CMPSO outperforms GPSOCC on 3 datasets and achieves similar accuracy on 3 of the remaining 5 datasets. In addition, on all datasets, CMPSO also achieves higher or similar training accuracy than GPSOCC.

Compared to GPSO1, CMPSO achieves higher testing accuracy on 3 out of the datasets and similar accuracy on 4 of the remaining 5 datasets. However, GPSO1 performs better than CMPSO on the Ionosphere dataset (about 2%). The reason is that CMPSO selects only

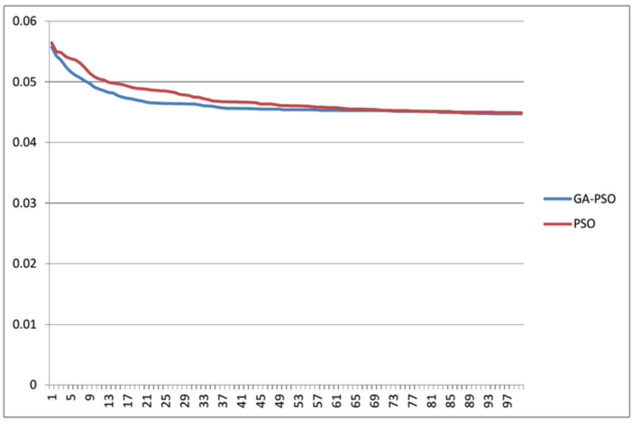
Dataset	Method	Ave-Size	Ave-Train ± Std-Train	Ave-Test ± Std-Test	T
	All	13		76.54	+
	GPSO1	5.4	96.71 ± 7.77E-14	96.59 ± 2.76	+
Wine	GPSOCC	4.60	97.37 ± 0.42	97.70 ± 2.52	=
	PSOCC	4.75	95.05 ± 0.58	96.70 ± 3.10	+
	CMPSO	4.70	97.28 ± 0.34	97.24 ± 2.89	
	All	18		83.86	+
	GPSO1	8.94	86.11 ± 0.20	84.30 ± 0.62	=
Vehicle	GPSOCC	7.30	90.10 ± 0.40	84.74 ± 0.49	-
	PSOCC	5.87	84.61 ± 0.56	84.72 ± 0.87	=
	CMPSO	7.57	90.25 ± 0.43	84.49 ± 0.44	
	All	34		83.81	+
	GPSO1	7.66	91.59 ± 0.47	89.50 ± 1.68	-
Ionosphere	GPSOCC	3.17	93.90 ± 0.67	86.89 ± 1.80	+
	PSOCC	9.7	90.04 ± 0.99	88.63 ± 1.68	=
	CMPSO	3.77	93.75 ± 0.86	87.94 ± 2.00	
	All	60		76.19	+
	GPSO1	17.64	86.74 ± 0.94	78.19 ± 4.14	+
Sonar	GPSOCC	10.17	90.67 ± 1.60	78.25 ± 2.95	+
	PSOCC	14.33	87.01 ± 2.00	78.94 ± 4.02	+
	CMPSO	11.60	91.59 ± 1.74	79.42 ± 2.48	
	All	166		83.92	+
	GPSO1	39.64	90.02 ± 0.60	84.95 ± 2.73	=
Musk1	GPSOCC	38.93	93.22 ± 1.40	83.29 ± 2.48	+
	PSOCC	35.03	89.78 ± 1.25	83.12 ± 3.41	+
	CMPSO	39.93	93.47 ± 1.12	85.06 ± 2.49	
	All	279		94.46	+
	GPSO1	45.5	94.87 ± 0.91	94.85 ± 0.34	=
Arrhythmia	GPSOCC	42.03	95.75 ± 0.18	95.12 ± 0.34	=
	PSOCC	44.17	95.11 ± 0.20	94.96 ± 0.38	=
	CMPSO	44.97	95.75 ± 0.19	95.07 ± 0.42	
	All	500		70.9	+
	GPSO1	36.08	85.45 ± 0.73	85.68 ± 1.10	-
Madelon	GPSOCC	51.13	89.20 ± 1.41	84.06 ± 1.64	-
	PSOCC	54.39	83.73 ± 1.74	83.40 ± 2.00	-
	CMPSO	107.2	89.4 ± 0.73	81.57 ± 1.46	
	All	649		98.63	+
Multiple	GPSO1	91.4	99.38 ± 0.38	99.01 ± 0.13	=
features	GPSOCC	51	99.36 ± 0.07	98.86 ± 0.17	+
	PSOCC	51.07	99.17 ± 0.09	98.84 ± 0.18	+
	CMPSO	110.77	99.53 ± 0.05	99.05 ± 0.01	

Table 5.1: Experimental Results of CMPSO

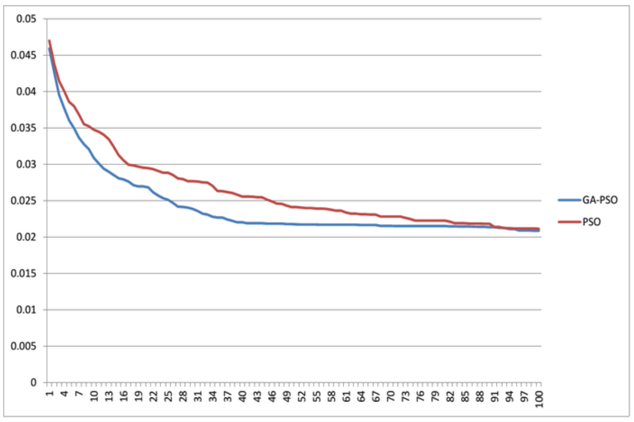
3.77 features, which is 2 times less than the number of selected features by GPSO1. Furthermore, on each dataset, CMPSO always achieves higher training accuracy than GPSO1. For example, on Sonar dataset, the training accuracy of CMPSO is 90.62%, which is 4% higher, while CMPSO selects even a smaller number of features than GPSO1.



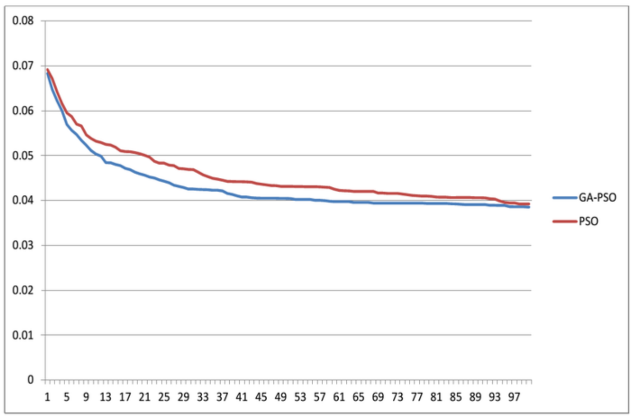
(a) Wine



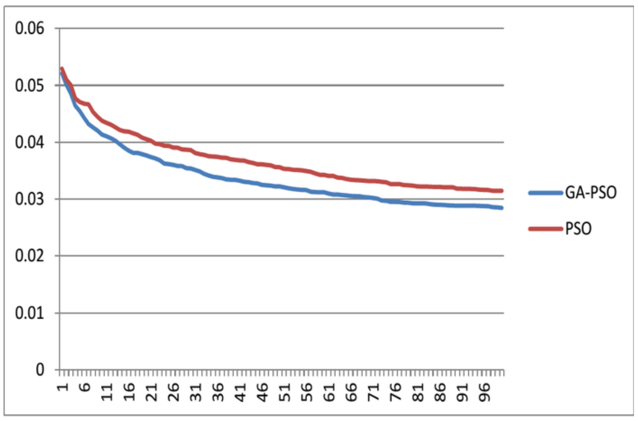
(b) Vehicle



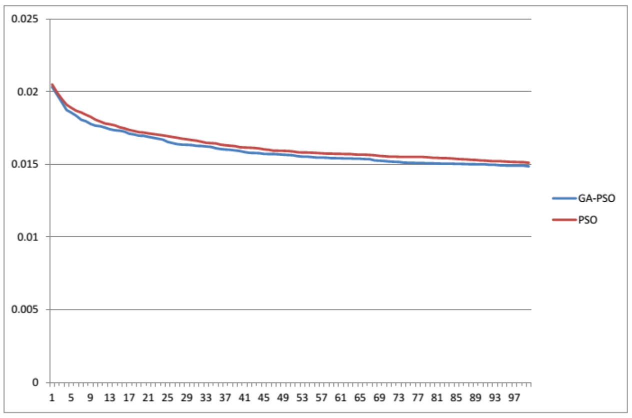
(c) Ionosphere



(d) Sonar



(e) Musk1



(f) Arrhythmia

Figure 5.3: Fitness evolution figures

The results suggest that integrating crossover and mutation operations into PSO improves the search ability of a PSO algorithm, which is clearly shown by the higher training accuracy on all datasets. However, someone might argue that this improvement is from the usage of continuous PSO, which usually has better exploration ability than binary PSO (GPSO1). In the next section, a comparison between standard PSO (PSO) and CMPSO is done to further analysis the effect of crossover and mutation operations.

5.4.2 Evolutionary Process

For each dataset, CMPSO and PSO are independently run 30 times. The fitness value of PSO is calculated by using the Equation 3.7. Each run contains 100 iterations, in which the fitness value is improved after each iteration. The average of *gbest*'s fitness value at each iteration is calculated by using 30 independent runs' results. Those average values are used to evaluate the fitness evolution of the above algorithms. The smaller fitness an algorithm achieves, the better the algorithm is. Figure 5.3 illustrates the fitness values in each iteration on 6 datasets, in which X-axis represents the iteration index and Y-axis represents the average fitness value at that iteration. CMPSO's fitness values are represented by blue lines, while red lines show the fitness values of PSO.

As can be seen from Figure 5.3, on all datasets, the blue lines are always under the red lines, which indicates that the positions discovered by CMPSO always have better fitness values then the ones discovered by PSO. On the small datasets like Wine, Vehicle, Ionosphere or Sonar, the difference between those 2 approaches are clearly shown in the middle iterations. Although at the end, the blue and red lines approach each other, the blue lines are still a little bit lower than the red lines, which indicates CMPSO still can better explores the search space than PSO.

The more significant difference is shown in a bigger dataset, Musk1. As can be seen in Figure 5.3e, despite of starting at the same position (same fitness value), the gap between the red and blue lines is maintained or even getting bigger with respect to the iteration order. At the end of the run, the difference of fitness values between these two approaches is still about 0.3%. On Arrhythmia, another dataset with a big number of features, the difference is not as significant as Musk1. The reason is that the starting position is good. At that position the fitness value is already 0.02, which is quite smaller, compared to the starting fitness of Musk1 (0.053). However, in Figure 5.3f, we still can recognise that the blue line is still under the red line.

The results clearly show that crossover and mutation operators would help PSO to better explore the search space, which is shown by the better fitness values that CMPSO can achieve during an entire run. In addition, CMPSO could reach good points in the search space earlier than the traditional PSO.

5.5 Summary

The goal of this chapter is to develop a PSO-based feature selection approach that can better explore the search space. This goal was achieved by integrating two genetic operators: crossover and mutation into the PSO. The results show that, on almost datasets, the proposed algorithm (CMPSO) can achieve similar or better testing accuracy than the other three methods PSOCC, GPSOCC and GPSO1. In terms of training accuracy and fitness evolution, CMPSO outperforms these three approaches as well as the standard PSO.

In CMPSO, although the training accuracy is significantly improved, the testing accuracy is not improved much or even getting worse than other PSO based approach. There might be an overfitting problem in this proposed algorithm. It is worth to develop a mechanism to balance between the training accuracy and testing accuracy. In addition, the Objective 1 shows that statistical clustering information can help PSO achieve better accuracy. It is promising to combine both CMPSO and statistical clustering information to achieve better classification performance.

Chapter 6

Conclusions

The aim of this project is to improve the standard PSO by using statistical clustering information and embedding genetic operators into PSO-based feature selection approaches for general classification problem. In particular, two main objectives were focused upon. Firstly, a new representation for continuous PSO, which takes the advantage of the statistical clustering information, is developed to select a smaller number of features and achieve better classification performance. Secondly, a hybrid PSO algorithm, which combines genetic operators and standard continuous PSO, is developed to better explore the search space to further improve the performance.

The first objective was achieved by developing the PSOCC algorithm in which a new representation is introduced. In the new representation, the number of features selected from a certain cluster is predefined. Therefore, the features within the same cluster compete with each other to be appeared in the solution. Experimental results show that by using the new representation with statistical clustering, the algorithm can significantly reduce the number of features in the datasets while the classification performance is similar or even better than using all features. GPSOCC is an improvement of PSOCC, which uses a Gaussian distribution to help particles smoothly move in the search space. According to conducted experiments, GPSOCC achieves similar or higher classification performance than PSOCC and smaller number of features.

To achieve second objective, the new PSO-based feature selection algorithm is developed (CMPSO) that uses genetic operators, that are crossover and mutation to avoid premature convergence and better explore the search space. The experimental results show that by embedding genetic operators, CMPSO achieve higher training accuracy than other PSO-based algorithms on all datasets. In addition, on most dataset, CMPSO is able to evolve feature subsets which achieve higher classification accuracy with a smaller number of features than using all features.

6.1 Future Work

Although the proposed algorithms successfully solve feature selection problems, they still have some limitations. Firstly, the maximum number of features selected from a cluster is predefined before the evolutionary process. This limitation might cause PSO skipping the best features subset, which contains more feature than the predefined number. It would be better if there is a mechanism which dynamically defines the maximum number of features selected from each cluster. In addition, in PSOCC and GPSOCC, the order indexing features in a cluster does matter. For example, suppose a cluster contains 3 features $\{f_1, f_2, f_3\}$. If the features are indexed in the order $[f_1, f_2, f_3]$, then it would be easier to move from f_1 to f_2

than move from f_1 to f_3. This problem can be solved in two ways. The first way is detecting the interaction between features. Basing on that information, it would be possible to find the best order of features within a cluster. The second way is using a set-based PSO [24] in which the position is a set of values rather than a vector of values as in standard PSO. Due to the limitation of time, these methods are left for future work.

The combination of PSO and genetic operators witnesses very high training accuracy. However, the experimental results also show that in some datasets, the testing accuracy is still low. This is an evidence of overfitting problem. A deep analysis of the effects of the genetic operators could be conducted in the future to solve overfitting problem. In CMPSO, the crossover operation performs more likely on bad particles which aims to improve the bad particles. However, the crossover operator might be better if the good particles are selected as parents. In addition, for some large datasets like Madelon or Multiple features, CMPSO tends to select a much larger number of features than PSOCC or GPSOCC while the testing accuracy is improved a little. It would be promising to use statistical feature clustering within CMPSO to balance between the number of selected features and the classification accuracy. The combination method is left for future work.

Bibliography

[1] Evo* 2015. http://www.evostar.org/2015/.

[2] Seal 2014. http://seal2014.otago.ac.nz/.

[3] ASUNCION, A., AND NEWMAN, D. Uci machine learning repository, 2007.

[4] BIN, W., QINKE, P., JING, Z., AND XIAO, C. A binary particle swarm optimization algorithm inspired by multi-level organizational learning behavior. *European Journal of Operational Research 219*, 2 (2012), 224–233.

[5] BOERINGER, D. W., AND WERNER, D. H. Particle swarm optimization versus genetic algorithms for phased array synthesis. *Antennas and Propagation, IEEE Transactions on 52*, 3 (2004), 771–779.

[6] CERVANTE, L., XUE, B., ZHANG, M., AND SHANG, L. Binary particle swarm optimisation for feature selection: A filter based approach. In *Evolutionary Computation (CEC), 2012 IEEE Congress on* (2012), IEEE, pp. 1–8.

[7] CHUANG, L.-Y., CHANG, H.-W., TU, C.-J., AND YANG, C.-H. Improved binary pso for feature selection using gene expression data. *Computational Biology and Chemistry 32*, 1 (2008), 29–38.

[8] CHUANG, L.-Y., CHANG, H.-W., TU, C.-J., AND YANG, C.-H. Improved binary pso for feature selection using gene expression data. *Computational Biology and Chemistry 32*, 1 (2008), 29–38.

[9] DASH, M., AND LIU, H. Feature selection for classification. *Intelligent data analysis 1*, 3 (1997), 131–156.

[10] EBERHART, R. C., AND SHI, Y. Comparison between genetic algorithms and particle swarm optimization. In *Evolutionary Programming VII* (1998), Springer, pp. 611–616.

[11] GANDELLI, A., GRIMACCIA, F., MUSSETTA, M., PIRINOLI, P., AND ZICH, R. E. Development and validation of different hybridization strategies between ga and pso. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on* (2007), IEEE, pp. 2782–2787.

[12] GHEYAS, I. A., AND SMITH, L. S. Feature subset selection in large dimensionality domains. *Pattern recognition 43*, 1 (2010), 5–13.

[13] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[14] GOLDBERG, D. E., AND HOLLAND, J. H. Genetic algorithms and machine learning. *Machine learning 3*, 2 (1988), 95–99.

[15] GUTLEIN, M., FRANK, E., HALL, M., AND KARWATH, A. Large-scale attribute selection using wrappers. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on* (2009), IEEE, pp. 332–339.

[16] GUYON, I., AND ELISSEEFF, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research 3* (2003), 1157–1182.

[17] HASSAN, R., COHANIM, B., DE WECK, O., AND VENTER, G. A comparison of particle swarm optimization and the genetic algorithm. In *Proceedings of the 1st AIAA multidisciplinary design optimization specialist conference* (2005), pp. 18–21.

[18] KENNEDY, J., EBERHART, R., ET AL. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (1995), vol. 4, Perth, Australia, pp. 1942–1948.

[19] KENNEDY, J., EBERHART, R., ET AL. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (1995), vol. 4, Perth, Australia, pp. 1942–1948.

[20] KENNEDY, J., AND EBERHART, R. C. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* (1997), vol. 5, IEEE, pp. 4104–4108.

[21] KOHAVI, R., AND JOHN, G. H. Wrappers for feature subset selection. *Artificial intelligence 97*, 1 (1997), 273–324.

[22] LANE, M. C., XUE, B., LIU, I., AND ZHANG, M. Particle swarm optimisation and statistical clustering for feature selection. In *AI 2013: Advances in Artificial Intelligence*. Springer, 2013, pp. 214–220.

[23] LANE, M. C., XUE, B., LIU, I., AND ZHANG, M. Gaussian based particle swarm optimisation and statistical clustering for feature selection. In *Evolutionary Computation in Combinatorial Optimisation*. Springer, 2014, pp. 133–144.

[24] LANGEVELD, J., AND ENGELBRECHT, A. P. Set-based particle swarm optimization applied to the multidimensional knapsack problem. *Swarm Intelligence 6*, 4 (2012), 297–342.

[25] LIANG, J., AND SUGANTHAN, P. N. Dynamic multi-swarm particle swarm optimizer. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE* (2005), IEEE, pp. 124–129.

[26] MARILL, T., AND GREEN, D. M. On the effectiveness of receptors in recognition systems. *Information Theory, IEEE Transactions on 9*, 1 (1963), 11–17.

[27] MATECHOU, E., LIU, I., PLEDGER, S., AND ARNOLD, R. Biclustering models for ordinal data. In *Presentation at the NZ Statistical Assn. Annual Conference. University of Auckland* (2011).

[28] NESHATIAN, K., AND ZHANG, M. Dimensionality reduction in face detection: A genetic programming approach. In *Image and Vision Computing New Zealand, 2009. IVCNZ'09. 24th International Conference* (2009), IEEE, pp. 391–396.

[29] NESHATIAN, K., AND ZHANG, M. Genetic programming for feature subset ranking in binary classification problems. In *Genetic programming*. Springer, 2009, pp. 121–132.

[30] Pledger, S., and Arnold, R. Multivariate methods using mixtures: Correspondence analysis, scaling and pattern-detection. *Computational Statistics & Data Analysis 71* (2014), 241–261.

[31] Pudil, P., Novovičová, J., and Kittler, J. Floating search methods in feature selection. *Pattern recognition letters 15*, 11 (1994), 1119–1125.

[32] Ratnaweera, A., Halgamuge, S., and Watson, H. C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *Evolutionary Computation, IEEE Transactions on 8*, 3 (2004), 240–255.

[33] Robinson, J., Sinton, S., and Rahmat-Samii, Y. Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In *Antennas and Propagation Society International Symposium, 2002. IEEE* (2002), vol. 1, IEEE, pp. 314–317.

[34] Shi, Y., and Eberhart, R. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on* (1998), IEEE, pp. 69–73.

[35] Stearns, S. D. On selecting features for pattern classifiers. In *Proceedings of the 3rd International Conference on Pattern Recognition (ICPR 1976)* (Coronado, CA, 1976), pp. 71–75.

[36] Trelea, I. C. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters 85*, 6 (2003), 317–325.

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

[38] Van Den Bergh, F. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2006.

[39] Whitney, A. W. A direct method of nonparametric measurement selection. *Computers, IEEE Transactions on 100*, 9 (1971), 1100–1103.

[40] Xue, B., Zhang, M., and Browne, W. N. Multi-objective particle swarm optimisation (pso) for feature selection. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference* (2012), ACM, pp. 81–88.

[41] Xue, B., Zhang, M., and Browne, W. N. New fitness functions in binary particle swarm optimisation for feature selection. In *Evolutionary Computation (CEC), 2012 IEEE Congress on* (2012), IEEE, pp. 1–8.

[42] Xue, B., Zhang, M., and Browne, W. N. New fitness functions in binary particle swarm optimisation for feature selection. In *Evolutionary Computation (CEC), 2012 IEEE Congress on* (2012), IEEE, pp. 1–8.

[43] Xue, B., Zhang, M., and Browne, W. N. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing* (2013).

[44] Yang, C.-S., Chuang, L.-Y., Ke, C.-H., and Yang, C.-H. Boolean binary particle swarm optimization for feature selection. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on* (2008), IEEE, pp. 2093–2098.

[45] YUAN, H., TSENG, S.-S., GANGSHAN, W., AND FUYAN, Z. A two-phase feature selection method using both filter and wrapper. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on* (1999), vol. 2, IEEE, pp. 132–136.

[46] ZHAO, H., SINHA, A. P., AND GE, W. Effects of feature construction on classification performance: An empirical study in bank failure prediction. *Expert Systems with Applications 36*, 2 (2009), 2633–2644.

[47] ZHU, Z., ONG, Y.-S., AND DASH, M. Wrapper–filter feature selection algorithm using a memetic framework. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 37*, 1 (2007), 70–76.