Evolutionary Computation for Feature Manipulation in Classification on High-dimensional Data

by

Binh Ngan Tran

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2018

Abstract

More and more high-dimensional data appears in machine learning, especially in classification tasks. With thousands of features, these datasets bring challenges to learning algorithms not only because of the curse of dimensionality but also the existence of many irrelevant and redundant features. Therefore, *feature selection* and *feature construction* (or feature manipulation in short) are essential techniques in preprocessing these datasets. While feature selection aims to select relevant features, feature construction constructs high-level features from the original ones to better represent the target concept. Both methods can decrease the dimensionality and improve the performance of learning algorithms in terms of classification accuracy and computation time.

Although feature manipulation has been studied for decades, the task on high-dimensional data is still challenging due to the huge search space. Existing methods usually face the problem of stagnation in local optima and/or require high computation time. Evolutionary computation techniques are well-known for their global search. *Particle swarm optimisation* (PSO) and *genetic programming* (GP) have shown promise in feature selection and feature construction, respectively. However, the use of these techniques to high-dimensional data usually requires high memory and computation time.

The overall goal of this thesis is to investigate new approaches to using PSO for feature selection and GP for feature construction on high-dimensional classification problems. This thesis focuses on incorporating a variety of strategies into the evolutionary process and developing new PSO and GP representations to improve the effectiveness and efficiency of PSO and GP for feature manipulation on high-dimensional data.

This thesis proposes a new PSO based feature selection approach to high-dimensional data by incorporating a new local search to balance global

and local search of PSO. A hybrid of wrapper and filter evaluation method which can be sped up in the local search is proposed to help PSO achieve better performance, scalability and robustness on high-dimensional data. The results show that the proposed method significantly outperforms the compared methods in 80% of the cases with an increase up to 16% average accuracy while reduces the number of features from one to two orders of magnitude.

This thesis develops the first PSO based feature selection via discretisation method that performs both multivariate discretisation and feature selection in a single stage to achieve better solutions than applying these techniques separately in two stages. Two new PSO representations are proposed to evolve cut-points for multiple features simultaneously. The results show that the proposed method selects less than 4.6% of the features in all cases to improve the classification performance from 5% to 23% in most cases.

This thesis proposes the first clustering-based feature construction method to improve the performance of single-tree GP on high-dimensional data. A new feature clustering method is proposed to automatically group similar features into the same group based on a given redundancy level. The results show that compared with standard GP, the new method can select less than half of the features to construct a new high-level feature that achieves significantly better accuracy in most cases. The combination of the single constructed feature and the selected ones achieves the best performance among different feature sets created from a single tree.

This thesis develops the first class-dependent multiple feature construction method using multi-tree GP for high-dimensional data. A new GP representation and a new filter fitness function that combines two filter measures are proposed to evaluate the whole set of constructed features more effectively and efficiently. The results show that in 83% of the cases, with less than 10 constructed features, the class-dependent method increases up to 32% average accuracy on using all the original thousands of features and 10% on using those constructed by the class-independent method.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, **Prof Mengjie Zhang** and **Dr Bing Xue** for their academic guidance and spiritual support during my PhD course. Prof Zhang has spent dedicated time and efforts in helping to develop my research skills, reading this thesis and my research articles. He provided not only constructive but also challenging feedbacks to improve my research work. Without his encouragement and support, this thesis would not have been achieved. I would like to reserve my special thanks to Dr Xue who is always nice to discuss with, regardless of academic or personal life. I am indebted to her for the dedicated hours, valuable comments, suggestions and persistent encouragement that she gave to me.

I am deeply grateful to Can Tho University for giving me the opportunity to follow my doctoral journey. I need to thank Vietnam International Education Development and Victoria University of Wellington for granting me the scholarship that enables the completion of my study in New Zealand.

I would like to thank staff and students in the Evolutionary Computation Research Group for their valuable feedbacks. I gratefully acknowledge Dr Su Nguyen for his valuable help. Discussions with him are great sources of novel ideas. I thank my officemates, especially Dr Harith Al-Sahaf, for sharing experience and suggestions. I also want to thank my Parkvale's friends for sharing my stressful moments with lots of laughs and lovely jokes.

Last but not least, I would like to thank my beloved mum for her unconditional love and caring. I owe a very large debt to my beloved husband and dearest sons for their love, patience and support.

List of Publications

1. Binh Tran, Bing Xue and Mengjie Zhang, "A New Representation in PSO for Discretisation-Based Feature Selection," IEEE Transactions on Cybernetics 48(6), 2018, 1733–1746.

2. Binh Tran, Bing Xue, Mengjie Zhang, and Su Nguyen, "Investigation on Particle Swarm Optimisation for Feature Selection on High-dimensional Data: Local Search and Selection Bias," *Connection Science* 28(3), 2016, 270–294.

3. Binh Tran, Bing Xue and Mengjie Zhang, "Genetic Programming for Feature Construction and Selection in Classification on High-dimensional Data," Memetic Computing, 8(1), 2016, 3–15.

4. Binh Tran, Bing Xue, and Mengjie Zhang, "Class Dependent Multiple Feature Construction Using Genetic Programming For High-Dimensional Data," in *Proceedings of the 30th Australasian Joint Conference on Artificial Intelligence (AI2017)*, Vol. 10400 of *Lecture Notes in Computer Science*. Springer, 2017, 182–194.

5. Binh Tran, Stjepan Picek, and Bing Xue. "Automatic Feature Construction for Network Intrusion Detection." in *Proceedings of the 11th International Conference on Simulated Evolution and Learning (SEAL 2017)*, Vol. 10593 of *Lecture Notes in Computer Science*. Springer, 2017, 569–580.

6. Binh Tran, Mengjie Zhang, Bing Xue, "A PSO Based Hybrid Feature Selection Algorithm for High-Dimensional Classification," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, 2016, 3801–3808.

7. Binh Tran, Mengjie Zhang, Bing Xue, "Bare-Bone Particle Swarm Optimisation for Simultaneously Discretising and Selecting Features For High-Dimensional Classification," in *Proceedings of the 19th European Conference on the Applications of Evolutionary Computation (EvoIASP)*, vol. 9597 of *Lecture Notes in Computer Science*. Springer, 2016, 701–718.

8. Binh Tran, Bing Xue, and Mengjie Zhang, "Using Feature Clustering for GP-Based Feature Construction on High-Dimensional Data," in *Proceedings of the 20th European Conference on Genetic Programming (EuroGP)*, vol. 10906 of *Lecture Notes in Computer Science*. Springer, 2017, 210–226.

9. Binh Tran, Mengjie Zhang, and Bing Xue, "Multiple Feature Construction in High-Dimensional Data Using Genetic Programming," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, 1-8. DOI:10.1109/SSCI.2016.7850130.

10. Binh Tran, Bing Xue, and Mengjie Zhang, "Improved PSO for Feature Selection on High-Dimensional Datasets," in *Proceedings of the 10th International Conference on Simulated Evolution And Learning (SEAL2014)*, vol. 8886 of *Lecture Notes in Computer Science*. Springer, 2014, 503–515.

11. Binh Tran, Bing Xue, and Mengjie Zhang, "Overview of Particle Swarm Optimisation for Feature Selection in Classification," in *Proceedings of the 10th International Conference on Simulated Evolution And Learning (SEAL2014)*, vol. 8886 of *Lecture Notes in Computer Science*. Springer, 2014, 605–617.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Statement

In machine learning and data mining, classification is an important task that aims to classify an instance into its corresponding category or class [34]. A classification algorithm learns a classifier or model based on an input collection of instances or examples. Each instance is typically described by a set of features and a class label. The quality of the input feature set is a key factor influencing the performance of a classification algorithm [221]. If the collected features are relevant to the class labels, a learning algorithm could induce the relationship between them. However, in many cases, which features are relevant is often unknown, especially when the domain knowledge is unavailable or incomplete. Therefore, the input data usually contains as many features as one can collect to present the problem. By retaining all potential important information, this could mean datasets may have many irrelevant and redundant features that are not actually useful in learning the target concept. The presence of these features may obscure the effect of the

relevant features on showing the hidden pattern of the data, and thereby reduce the representativeness of the whole feature set [267]. This can hinder learning algorithms from inducing a good classifier. Therefore, to optimise the performance of the learnt classifier, it is vital to eliminate these irrelevant and redundant features [57].

Feature manipulation [140] refers to the process of transforming the input space of a machine learning task in order to improve the learning quality and performance. It includes feature selection and feature construction, which are also known as *dimensionality reduction* methods. While *feature selection* chooses relevant features from the original set of features, *feature construction* creates new high-level features by combining the original low-level features using predefined operators or functions. The selected and constructed features are expected to better represent the target concept [140].

Meanwhile, feature ranking is a relaxed version of feature selection. It ranks features based on their relevance to the target concept. The top ranked features will be chosen to form the best feature subset. These techniques help the learning algorithms improve their classification performance, simplify the learnt classifier and reduce the computation time [57].

Feature manipulation is an important data preprocessing step, especially for classifying high-dimensional data. Recently, there has been an immense increase in *high-dimensional data* such as microarray gene expression, proteomics, images, text, and web mining data [72]. These datasets usually have thousands to tens of thousands of features. When working on these datasets, traditional learning algorithms that were developed for low-dimensional data may suffer from the curse of dimensionality [78], which is various phenomena arising when analysing or organising data in high-dimensional spaces. In machine learning, as the number of features grows linearly, the amount of data we need to generalize an accurate model grows exponentially. However, this requirement is usually not satisfied in reality. Furthermore, these datasets can have a large amount of irrelevant and redundant information, which may significantly limit the performance of learning algorithms [140]. Therefore,

dimensionality reduction is crucial for efficient learning in problems with a very large number of features.

Feature manipulation methods can be divided into three broad categories, namely *filter, wrapper*, and *embedded methods* [44, 91]. Filter methods evaluate the manipulated features based on the intrinsic characteristics of the data, which can be determined via different types of measures such as distance, information, dependence and consistency measures [57]. In comparison to wrapper and embedded methods, filter methods are usually computationally less expensive. Furthermore, since filters are independent of any classification algorithm, the performance of manipulated features typically does not bias to any learning algorithm [42, 259]. This property is referred to as the generality of a feature manipulation method. Wrapper methods, on the other hand, use a classification algorithm to evaluate the manipulated features. Therefore, a learning algorithm is typically employed in the evaluation procedure. As compared to filter methods, wrapper methods usually achieve better classification performance. However, their generality may be limited and the computation time is typically long, especially when the number of features is large [194]. Unlike filter and wrapper methods, embedded methods simultaneously learn a classifier and choose a subset of features. Traditional machine learning algorithms like decision trees or support vector machines are typical examples of embedded approaches [91].

Although feature manipulation has been studied for decades, applying it to high-dimensional data is still challenging. Feature selection is actually a combinatorial optimisation problem. An exhaustive search for the optimal subset from 2^N possible combinations of N original features is impractical, especially when N is large. Feature construction has an even larger search space comprising not only original features but also possible operators used to combine these features. In high-dimensional datasets, traditional methods such as sequential forward selection (SFS) [240], sequential backward selection (SBS) [153], sequential floating forward selection (SFFS), and sequential floating backward selection (SFBS) [193] easily get stuck into local optima.

Therefore, it is necessary to have a more efficient global search approach.

Evolutionary Computation (EC) techniques have been proposed for feature manipulation thanks to their ability to perform a global search with a population of individuals searching in the solution space. EC techniques such as *Particle Swarm Optimisation (PSO)* and *Genetic Programming (GP)* have been applied to feature manipulation [174, 255]. While PSO is a swarm intelligence method which simulates the social behaviours of birds flocking, GP is an evolutionary algorithm which works based on Darwinian evolution principles. In PSO, a swarm of particles communicate their knowledge to optimise their positions or solutions. On the other hand, in GP, fittest individuals are selected to evolve new individuals using crossover or mutation. Although both methods have been applied to feature manipulation, most of the current works focus on solving problems with a relatively small number of features (a few hundred) [51, 174, 252]. Applying these evolutionary computation methods on high-dimensional problems is still a challenge in this field. This gap is the primary motivation for this research project.

1.2 Motivations

This section first describes challenges of feature manipulation on high-dimensional data, current approaches to addressing these problems and their limitations. An explanation is then followed to show why PSO and GP are chosen for feature manipulation along with their current limitations as well as potentials in this field.

1.2.1 Challenges of Feature Manipulation on High-Dimensional Data

Feature manipulation is a difficult combinatorial problem [258, 256] especially on high-dimensional data due mainly to the challenges of large search space, feature interaction and evaluation criteria.

In feature selection, the size of the search space growths exponentially with the number of available features (2^N possible combinations for N features). Therefore, exhaustive search becomes impractical when working on high-dimensional data with thousands to tens of thousands of features. Different heuristics or strategies have been proposed to search for good solutions. However, existing methods still suffer from the problem of being stuck in local optima. For feature construction, the size of the search space is even larger. Feature construction methods need to choose a good set of not only features but also operators as well as a suitable structure to combine them.

Feature interaction is another important issue in feature manipulation. There can be two-way, three-way or multi-way complex interactions among features [107]. A feature which is individually irrelevant to the target concept may become useful if it is complementary to other features. On the other hand, an individually relevant feature may be redundant or even noisy in the context of other features. With a large number of features, high-dimensional datasets have a higher probability of having these types of feature interactions. The existence of feature interaction hinders the strategy of sequentially generating a feature subset from obtaining the optimal solution.

Most evaluation criteria in the current filter approaches can only handle two-way interactions between features, i.e. evaluation applies to two features only. Meanwhile, evaluation functions in wrapper approaches can be expensive and may face the problem of overfitting when dealing with high-dimensional data. Note that given a fix number of training samples, the density of these samples decreased exponentially when we increased the dimensionality of the problem. In other words, high-dimensionality introduces sparseness to the training data. Due to this sparsity, it becomes much easier for learning algorithms to find a separable hyperplane to separate the training data very well, but it may perform poorly on unseen test data. This phenomenon is called overfitting. To avoid this sparsity, classification algorithms usually require an enormous amount of samples in order to avoid overfitting to the training data. However, this requirement is not always satisfied especially

in domains where collecting data is expensive such as gene expression data [220] and mass spectrometry data [8]. For example, AML-ALL [271], a gene expression dataset differentiating two subtypes of leukemia, has 7129 features and 72 examples. Therefore, feature subsets evaluated based on the results of these classification algorithms may not work well for unseen data.

Another common problem associated with these high-dimensional and small sample size datasets is feature selection bias. Because of the small sample size of these datasets, many studies [2, 5, 13, 20, 50, 158, 161, 162, 205, 210, 261] used the whole dataset to train their feature selection systems (i.e. to evaluate feature subsets during the training process), leaving no data left for testing. The training results are then used to evaluate the system performance, and thus leading to feature selection bias. This problem is quite common especially for wrapper-based feature selection methods, where a cross-validation is usually used to evaluate the feature subset [120]. This is even more complicated when EC-based techniques are used since many independent runs should be conducted to eliminate statistical variance. Therefore, a careful design of experiments and a good evaluation method are needed to deal with these high-dimensional data.

With these challenges, feature manipulation on high-dimensional data requires a global search technique and a good evaluation method to efficiently and effectively explore the huge solution space to find optimal or near-optimal solutions in a reasonable time. EC techniques are well-known for their global search ability thanks to their population-based beam search. In this thesis, we will investigate the ability of PSO in feature selection and GP in feature construction.

1.2.2 Why PSO for Feature Selection

PSO is an EC technique that has been successfully applied in many different fields [189]. Among these areas, more and more studies in feature selection use PSO [256] due to the following reasons:

- **Natural representation.** PSO has a vector representation which provides a natural encoding scheme for the feature selection task, where each element in the vector represents the likelihood of the corresponding feature being selected.

- **Efficiency.** Compared with other EC methods that are also used in feature selection such as genetic algorithms (GAs) and ant colony optimisation (ACO), PSO has less computation time and a faster convergence speed [117].

- **Simplicity.** Compared with other EC methods, PSO is simple and easy to implement.

- **The potential in feature selection.** PSO has shown promise in feature selection [248].

Although many PSO based feature selection methods have been proposed [230, 248], its application to high-dimensional data is still limited. This thesis aims to investigate PSO's potential in feature selection on high-dimensional problems.

1.2.3 Why GP for Feature Construction

GP is also a global search technique under the umbrella of EC. Among many EC techniques, it is an excellent choice for feature construction because of the following characteristics:

- **Flexible representation.** With a flexible evolutionary mechanism, GP does not require a fixed representation. It can be a tree or a graph whose structure will be dynamically evolved during the evolutionary process. In feature construction, a tree is usually used to represent a constructed feature with internal nodes being operators and terminal nodes being features. This enables GP to construct new features that are linear or non-linear combinations of the original features via different

operators. These new features can better represent the target concept than the original features.

- **Automatically evolving models.** GP is a very flexible technique that can automatically evolve mathematical models from a predefined set of terminals and operators. A GP evolved model can be executed by applying operators to terminals to produce a result. Therefore, GP provides a natural encoding scheme for feature construction where the terminal set comprises of original features, and operators are any function that can be applied to features to generate new feature.

- **Built-in feature selection.** A GP tree does not use all the original features to construct high-level features, which means GP implicitly performs feature selection. The features that are used in terminal nodes of the tree, i.e. terminal features, are potentially useful or informative features.

- **Interpretability.** Unlike in other feature transformation methods such as Principle Component Analysis and Linear Discriminant Analysis, the features constructed by GP are easier to understand and interpret in terms of both their origin and formation.

PSO and GP have shown promise in feature selection and construction, respectively; however, their capabilities for feature selection and construction on high-dimensional data have not been fully investigated and the limitations are shown in the following subsections.

1.2.4 Limitations of PSO for Feature Selection on High-Dimensional Data

PSO has been applied and shown its high potential to feature selection [25, 249]. However, due to the large search space, PSO based methods on high-dimensional data still face the problem of stagnation in local optima.

Different strategies have been proposed to address this problem, such as changing the leader particle [50, 258], reinitialising parts of the population [51], improving updating mechanism [161]. However, the achieved performance of these studies still needs to be improved. Cooperative coevolution [136, 137], new updating mechanism [48, 49, 108], and differential grouping [183] were also proposed and have shown promise in solving high-dimensional or also called large-scale optimisation problems. Nonetheless, most of these methods are tested on benchmarks of function approximation problems, where the characteristics of the search space are very different from feature selection. Furthermore, due to the complex interactions between features and the high computation cost of feature evaluation, these large-scale optimisation methods might not function well on high-dimensional feature selection. However, the success of these methods on large-scale function approximation problems motivates us to propose novel search strategies to improve PSO based feature selection method on high-dimensional data.

Feature selection via discretisation is a promising approach that has never been investigated in PSO. In data preprocessing, discretisation is also an important and popularly used technique especially for high-dimensional data. Many feature selection methods can only be applied to discrete data such as those based on mutual information [263]. High-dimensional datasets usually include continuous features that are automatically collected at the interval or ratio level such as images and gene expression data. In additions, discretisation techniques aim at finding a discrete representation of each feature so that it contains enough information for the learning task while eliminating the minor fluctuations that may be noisy in the original data [63]. Thanks to this ability, feature selection via discretisation has been proposed to select features based on the result of discretisation process [142, 105]. This approach has shown promise. However, the discretisation methods used in these studies are univariate, which means that only one continuous feature is discretised at a time [79]. For the sake of efficiency, these methods work with an assumption that each feature independently influences the task.

However, this assumption is not valid in most problems in which feature interdependency occurs [45]. Therefore, when using data discretised by a univariate discretisation method, feature selection methods may miss relevant features since information showing feature interaction may be destroyed in the discretisation process. Combining discretisation and feature selection into a single stage may obtain a better feature set for the learning task. PSO can represent all features in a single solution, which provides a good chance for multi-variate discretisation, but PSO has been used for feature selection and has never been proposed for feature selection via discretisation.

1.2.5 Limitations of GP for Feature Construction on High-Dimensional Data

Apart from feature selection, feature construction is also an effective technique to reduce the number of features by constructing new high-level features that can represent the original feature set which can then be removed and improve the classification accuracy [170]. Feature construction is a means to enhance the representation quality of the data, where the original features may not provide enough discrimination for learning algorithms to learn a good classifier [170]. In this case, feature construction aims at combining sets of features to obtain new features with stronger discriminating power. Therefore, the capability of a learning algorithm can be improved.

Based on its ability to automatically evolve free-form mathematical models, GP has been successfully proposed as a feature construction method using the single-tree representation [170] or the multiple-tree representation [125]. However, its applications to high-dimensional data are still limited [7, 9] due to a number of challenges. One of the main challenges of GP for feature construction on high-dimensional data is the huge search space. Since GP needs to search not only a good combination of features from thousands of features but also an appropriate set of operators to combine them. Although GP has a built-in feature selection capacity, its performance is still affected

when the number of features increases to thousands. Another challenge is how to design a good evaluation measure to avoid overfitting especially when the number of samples or instances is significantly smaller than the dimensionality of the problem [149]. In these cases, GP can easily find a model that perfectly fit the training data but has a poor performance on unseen test data. In general, feature construction has a larger search space and is easier to overfit than feature selection as the model can be more precisely tuned to fit the data.

1.2.6 Limitations of Conventional Feature Manipulation Methods on High-Dimensional Data

To address the problem of high computational cost in a large search space, many filter feature selection methods have been proposed in the last decades. A popular technique used in this context is feature ranking, sometimes also called feature weighting [222], which ranks features based on their degrees of relevance to the target concept. Feature ranking assesses features individually using some measures such as the distance between instances of different classes [199], feature dispersion [72], or classification accuracy [254]. After features have been ranked, the final feature subset will be formed by choosing a predefined number of top-ranked features. These methods are computationally efficient because of their linear time complexities. However, all of them require a certain domain knowledge to determine the number of features to be selected. Another more serious issue is that they cannot discover redundancy or interactions among features. The top-ranked features may include redundant information, which can affect the speed and accuracy of learning algorithms. Therefore, in feature selection for high-dimensional data, purely relevance-based feature ranking methods do not meet the needs of feature selection in commonly found data domains.

To address these limitations, a two-stage approach was proposed where feature ranking is performed following by a heuristic search to remove redundant

features from the ranked list [63, 263]. In the second stage, a measure can be used to evaluate feature redundancy such as symmetric uncertainty [263] or feature dispersion [72]. There has been an increased amount of research into this approach for high-dimensional data in the last decade [243, 270]. In general, these methods have shown to be effective and efficient to remove redundant features. However, selecting an appropriate number of features from the first stage is not straightforward especially for high-dimensional data with thousands of features. Moreover, since the features are ranked individually in the first stage and the proposed redundancy measures can only detect redundancy between two features, they may fail to identify multiple feature interactions [263, 271]. A wrapper method was also proposed in the second stage [63] to evaluate the whole feature subset taking into account multiple feature interactions. However, because the interactions between features are not considered in the first stage, some relevant features may be left out in the second stage. Therefore, search techniques that can consider feature interaction are needed for better solutions.

For feature construction, techniques such as Principal Component Analysis and Fisher's Linear Discriminant Analysis have been tried on high-dimensional data [60, 195]. Although these methods are widely used in many areas, they are limited to only find the optimal coefficients in a limited number of predefined models (e.g. linear, polynomial). Recently, deep learning [26], e.g. deep neural networks and deep belief network, has shown promise in feature learning for speech recognition, natural language processing, computer vision, etc. [61]. However, large amounts of data and computational resources are crucial to their performance. Furthermore, solutions of these approaches also have low intelligibility or human readability.

1.3 Research Goals

The overall goal of this thesis is to investigate a new approach to using Evolutionary Computation techniques, i.e. PSO and GP, for feature manipulation

on high-dimensional classification problems. This approach is anticipated to effectively and efficiently solve feature selection and feature construction problems with thousands to tens of thousands of features, which may include a large number of irrelevant, redundant features and feature interactions. The specific objectives of this research are to:

1. Develop a new PSO based feature selection algorithm for high-dimensional data by proposing a new local search strategy, which can be integrated with PSO to balance the global and local search to achieve better performance. The local search needs to be designed in a way that the evaluation process can be sped up to avoid high computational cost since most of the computational cost in wrapper feature selection is caused by the evaluations. The proposed algorithm is expected to significantly reduce the dimensionality. The selected features are expected to achieve better classification performance than using the original full feature set as well as features selected by the conventional methods and other existing PSO based feature selection methods.

   PSO works based on collective intelligence which is implemented via simple communication between particles in searching for better solutions. Thanks to such swarming behaviours, PSO is capable of quickly detecting fruitful regions; however, once there, it cannot perform a refined local search in a complex search space to compute the optimum with high accuracy [18]. Local search has been combined with PSO to overcome this drawback [188]. This approach, which is also called memetic algorithm, has been shown effective in many feature selection algorithms [271, 272]. However, this strategy has not been investigated much in PSO based feature selection especially for high-dimensional data.

2. Develop a new PSO based feature selection via discretisation for high-dimensional data by proposing a new representation and a new evaluation method for PSO to perform discretisation and feature selection

simultaneously in a single stage. The selected discrete features are expected to have better performance than the original feature set and those generated by the commonly used two-stage approach where discretisation followed by feature selection [263].

PSO has been used for feature selection; however, it has never been applied to discretisation. With real-number vector representation and a global search ability, PSO has a high potential in multivariate discretisation where multiple features are discretised at once taking into account possible interactions between features. In this way, useful relationships between features can be maintained after discretisation. As a result, feature selection via multi-variate discretisation may provide better solutions.

3. Develop a new feature construction algorithm using single-tree GP for high-dimensional problems by proposing a new feature clustering technique to narrow the GP search space. The proposed GP based feature construction algorithm is expected to construct new high-level features that can improve the classification performance of the common classification algorithms on high-dimensional data. The performance of different combinations of selected and constructed features from a single tree are also investigated.

   Although GP has a built-in capability to select good features based on the guide of the fitness function, its performance is still affected when applied to high-dimensional data due to the huge search space [228]. Therefore, it is critical to narrow this search space for GP performance improvement. Results from a GP based feature selection method [7] and a classification algorithm [168] have shown that helping GP select appropriate features is critical in enhancing its performance. Furthermore, feature clustering has been proposed to group similar features into the same cluster so that feature selection can be effectively achieved by choosing one or several representative features from each

group [89, 109, 127, 200]. However, feature clustering has not been investigated in GP for feature construction.

4. Develop a new multiple feature construction algorithm using multi-tree GP for high-dimensional data by proposing a new representation to construct multiple features, each of which focuses on discriminating instances of one class from the remaining classes. A new fitness function is also developed to better evaluate the performance of the constructed features. The proposed approach is expected to construct a small set of features that has better discriminating ability than the original large feature set. The constructed features are expected to help common classification algorithms work more effectively and efficiently.

   In feature manipulation, evaluating features is a critical component. Identifying relevant features to the target class is very important. However, not all relevant features can distinguish instances from all classes. Some particular features may have better ability than other features to distinguish instances of a particular class from other classes. Therefore, evaluating features in the context of class-dependency may better reveal the hidden patterns of the data. Recently, this approach has been proposed in feature selection [77, 155, 237] and showed that evaluating features in a class-based context led to a better performance. However, the investigation of this strategy to feature construction is still limited.

1.4 Major Contributions

This thesis makes the following major contributions, each of which is extensively discussed in each of the contribution Chapters 3 to 6 as appropriate.

1. This thesis proposes a new PSO approach with local search to feature selection on high-dimensional data to significantly reduce the number of features while increasing the classification performance. Two new local search strategies are proposed for PSO based feature selection to balance

the global and local search of the algorithm. The proposed algorithm is also designed in such a way that it can avoid the large computational cost when using local search. Experimental results show that the proposed method achieved much smaller feature subsets with significantly better classification performance than using the original feature sets, those selected by standard PSO and traditional feature selection methods. The results and analyses have suggested that local search using general knowledge in feature selection can significantly improve the performance of PSO in feature selection on high-dimensional data. Further analyses also show the effectiveness and robustness of the proposed methods. In addition, a careful design of experiments is developed to avoid feature selection bias when using EC methods. This design can be applied to any feature selection or feature construction system. A comparison between the experimental results with and without feature selection bias demonstrates the necessity of the correct experiment design.

Parts of this contribution have been published in:

Binh Tran, Bing Xue, and Mengjie Zhang, "Improved PSO for Feature Selection on High-Dimensional Datasets," in *Proceedings of the 10th International Conference on Simulated Evolution And Learning (SEAL2014)*, vol. 8886 of *Lecture Notes in Computer Science*. Springer, 2014, 503–515.

Binh Tran, Mengjie Zhang, Bing Xue, "A PSO Based Hybrid Feature Selection Algorithm for High-Dimensional Classification," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, 2016, 3801–3808.

Binh Tran, Bing Xue, Mengjie Zhang, and Su Nguyen, "Investigation on Particle Swarm Optimisation for Feature Selection on High-dimensional Data: Local Search and Selection Bias," *Connection Science* 28(3), 2016, 270–294.

2. This thesis proposes the first PSO approach to feature selection via

discretisation for high-dimensional data. To achieve discretisation, two methods with two new PSO representations are proposed to evolve cut-points for multiple features simultaneously. A cut-point is a real value within the feature range used to partition that range into two intervals. Feature selection is accomplished by removing features that are discretised into only one interval. Experiment results show that the proposed methods can effectively discretise multiple features to significantly improve or maintain the classification performance in most cases. Through discretisation, a much smaller number of relevant features is selected at the same time. Comparisons with two-stage (one for discretisation and one for feature selection) approaches using PSO or conventional methods show that conducting discretisation and feature selection in a single stage is more effective than applying these techniques in two separate stages.

Parts of this contribution have been published in:

Binh Tran, Mengjie Zhang, Bing Xue, "Bare-Bone Particle Swarm Optimisation for Simultaneously Discretising and Selecting Features For High-Dimensional Classification," in *Proceedings of the 19th European Conference on the Applications of Evolutionary Computation (EvoIASP)*, vol. 9597 of *Lecture Notes in Computer Science*. Springer, 2016, 701–718.

Binh Tran, Bing Xue and Mengjie Zhang, "A New Representation in PSO for Discretisation-Based Feature Selection," IEEE Transactions on Cybernetics, 2017, DOI: 10.1109/TCYB.2017.2714145

3. This thesis proposes the first feature clustering based GP approach to feature construction on high-dimensional data. A new clustering method is proposed to group similar or redundant features into a particular group. Features from these groups are chosen to construct high-level features. The proposed clustering technique can automatically determine the number of clusters based on a given correlation or redundancy level

between features. Experiment results show that the proposed method selects a smaller number of features to construct a better discriminating feature than the standard GP. Feature clustering helps GP construct features with better discriminating ability than those constructed by GP using the whole feature set. Investigation on different combinations of selected and constructed features from a single GP tree shows that the combination of the single constructed feature and the selected ones achieves the best performance among them. This is because the selected features help to alleviate the overfitting problem in the constructed feature. Further analysis also shows that GP can cope with small sample size and skew distribution to obtain a good generalisation.

Parts of this contribution have been published in:

Binh Tran, Bing Xue and Mengjie Zhang, "Genetic Programming for Feature Construction and Selection in Classification on High-dimensional Data," Memetic Computing, 8(1), 2016, 3–15.

Binh Tran, Bing Xue, and Mengjie Zhang, "Using Feature Clustering for GP-Based Feature Construction on High-Dimensional Data," in *Proceedings of the 20th European Conference on Genetic Programming (EuroGP)*, vol. 10906 of *Lecture Notes in Computer Science*. Springer, 2017, pp.210–226.

4. This thesis proposes the first class-dependent feature construction method using GP for high-dimensional data. The proposed representation enables each constructed feature to focus on distinguishing one class from other classes. A new filter-based fitness function is proposed to evaluate the whole set of constructed features more effectively and efficiently. Experiment results show that the class-dependent constructed features help different classification algorithms achieve better performance than using the original full feature set and using features constructed by the class-independent approach. Results also show that the proposed method can construct one or two high-level features for

each class to obtain comparable or even better results than the selected features.

Parts of this contribution have been published in:

Binh Tran, Mengjie Zhang, and Bing Xue, "Multiple Feature Construction in High-Dimensional Data Using Genetic Programming," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, 1-8. DOI:10.1109/SSCI.2016.7850130.

Binh Tran, Bing Xue, and Mengjie Zhang, "Class Dependent Multiple Feature Construction Using Genetic Programming For High-Dimensional Data," in *Proceedings of the 30th Australasian Joint Conference on Artificial Intelligence (AI2017)*, Vol. 10400 of *Lecture Notes in Computer Science*. Springer, 2017, pp.182–194.

1.5 Organisation of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 introduces the essential background and related work. The major contributions of the thesis are presented in Chapters 3-6. Chapter 7 concludes the thesis.

Chapter 2 presents the essential background of machine learning and classification, the basic concepts in feature manipulation, an overview of evolutionary computation and an introduction of PSO and GP techniques. It also reviews the related work in feature selection and feature construction using the conventional as well as evolutionary computation technique methods.

Chapter 3 proposes two new local search strategies to integrate into PSO for feature selection on high-dimensional data. A fast fitness evaluation is also proposed and illustrated. It then describes the details of the conducted experiments to compare the proposed methods against the conventional feature selection methods and existing PSO based algorithms. The results of the compared methods on a set of commonly-used high-dimensional data with varying difficulty are then presented and discussed. Further analysis of

computation time, the robustness and learning capability of the proposed methods are also shown at the end of the chapter.

Chapter 4 proposes the first PSO approach to feature selection via discretisation with two new representations. The two proposed algorithms are described, tested and compared with the two-stage approach, where traditional discretisation method is combined with either conventional or PSO based feature selection methods. The results of all the compared methods on ten high-dimensional data are displayed and examined. The final section of this chapter provides further discussion on the generalisation, robustness and learning capability of the proposed approach.

Chapter 5 develops a new feature clustering based GP approach to feature construction on high-dimensional data. The proposed redundancy based feature clustering algorithm is described along with the overall structure, representation and fitness function of the proposed GP based feature construction method. It then describes the design and results of the experiments conducted to compare the proposed method with a traditional feature construction method and standard GP. It also investigates the performance of different combinations of the constructed and selected features from the single GP tree. Further analysis of the returned clusters, the constructed features and the generalisability of the proposed algorithm are discussed at the end of the chapter.

Chapter 6 proposes a novel class-dependent feature construction method using GP for high-dimensional data. Two representations for class-independent and class-dependent multiple feature construction are proposed and presented. Then, it describes the two corresponding GP based methods and compares their results. The proposed class-dependent method is also compared with the single feature construction method proposed in Chapter 5 and an existing class-dependent feature construction method using single-tree GP. The constructed features are also visualised to show their discriminating ability.

Table 1.1: Datasets

Dataset	Abbrev.	#F	#Inst.	#Class	Class Distribution	Ref.
Alizadeh-V1	Alizadeh	1,095	42	2	50%, 50%	[59]
Colon	Colon	2,000	62	2	35%, 65%	[271]
SRBCT	SRBCT	2,308	83	4	13%, 22%, 30%, 35%	[218]
Yeoh-V1	Yeoh	2,526	148	2	83%, 17%	[59]
DLBCL	DLBCL	5,469	77	2	25%, 75%	[218]
9_Tumors	9Tumor	5,726	60	9	3%, 10%, 10%, 10%, 12%, 13%, 13%, 13%, 15%	[218]
Leukemia1	Leukemia1	5,327	72	3	12%, 35%, 53%	[218]
Brain_Tumor1	Brain1	5,920	90	5	4%, 7%, 11%, 11%, 28%	[218]
ALL-AML	Leukemia	7,129	72	2	35%, 65%	[271]
Central nervous system	CNS	7,129	60	2	35%, 65%	[271]
Leukemia2	Leukemia2	11,225	72	3	28%, 33%, 39%	[218]
Brain_Tumor2	Brain2	10,367	50	4	14%, 28%, 28%, 30%	[218]
Prostate_Tumor	Prostate	10,509	102	2	49%, 51%	[218]
11_Tumors	11Tumor	12,533	174	11	3%, 4%, 5%, 6%, 7%, 8%, 8%, 13%, 15%, 15%, 16%	[218]
Lung_Cancer	Lung	12,600	203	5	3%, 8%, 10%, 10%, 69%	[218]
Ovarian_Cancer	Ovarian	15,154	253	2	36%, 64%	[271]

1.6 Benchmark Datasets

Throughout this thesis, the proposed PSO based feature methods and GP based feature construction methods are evaluated on a set of high-dimensional gene expression datasets of varying difficulty. These datasets are commonly used in the literature to evaluate the performance of feature selection or construction methods developed for high-dimensional data. Table 1.1 shows their names, abbreviations, numbers of features, instances, and classes, as well as the percentage of instances in each class. The datasets are listed in ascending order of their dimensionality.

As can be seen from Table 1.1, these datasets have quite different numbers

of features ranging from one to fifteen thousand. In contrast, their numbers of instances range from 42 to 253, which are quite small compared to the number of features. The number of classes in each dataset varies from 2 to 11. Many of them are also unbalanced with varying degrees. These characteristics make these datasets become challenging problems for learning algorithms to obtain their best performance. Ten datasets from [218] are publicly available at *http://www.gems-system.org*.

2

Literature Review

Before reviewing typical work on feature selection and feature construction, this chapter starts with an essential background about machine learning and classification, basic concepts in feature manipulation, an overview of evolutionary computation and an introduction of PSO and GP techniques. Finally, a summary of limitations of existing work and motivation of this project are discussed.

2.1 Machine Learning and Classification

By developing algorithms that allow computers to automatically learn from experience rather than being explicitly programmed to carry out a certain task, machine learning (ML) is a major research area in artificial intelligence. Learning from examples and making changes to improve their performance is the main characteristic of ML algorithms [160].

ML algorithms can be classified into supervised, unsupervised and reinforcement learning [221]. In supervised learning, the examples or instances

include the inputs and their desired outputs. Therefore, the aim of supervised learning algorithms is to generate a model or a function that can map inputs to desired outputs. On the other hand, desired outputs are not given in unsupervised learning. The algorithms have to infer inherent patterns or subsets of similar examples from the input. Different from supervised and unsupervised learning, reinforcement learning interacts with the environment through a sequence of actions, receives feedback on the goodness of its action via rewards or penalties to learn how to choose the best actions.

Classification is a typical supervised ML task, which uses a model or a classifier to assign an object to a given category based on its input variables [34]. Examples of classification would be assigning an email into spam or non-spam classes, and assigning a diagnosis to a patient based on observed characteristics of that patient. The main objective of ML algorithms in this area is to learn a classifier or a model that can correctly classify an input instance into a predefined class label [34]. Thereby, these learning algorithms are also called classifier inducers or classification algorithms.

2.1.1 Training and Testing

Typical supervised ML systems involve two processes: training and testing. While training refers to the learning process which induces a new classifier from given instances, testing denotes the process used to evaluate the performance of the learnt classifier on new or unseen instances [160]. Instances from problem domain are also called samples, examples or observations. Collections of instances used in training and testing process are referred to as the training set and the test set respectively. Each instance, which is usually presented as a row in a dataset, comprises of many values of features, or attributes, or input variables which are presented in different columns. The values of these features can be real or nominal (discrete or ordinal). In classification problems, each instance belongs to a single class. If the number of different classes in a dataset is two, it is called a binary-classification problem, otherwise, it is a multi-class classification problem.

To evaluate the performance of a learning algorithm, or to examine its generalisability, different ways of splitting datasets into training and test sets have been used [119]. The main principle here is to leave some instances in the dataset out of the training process so that the performance of the learnt classifier can be tested by checking if it can correctly identify unseen instances. *Holdout* is a method in which a dataset is divided into two disjoint sets for training and testing using a predefined proportion. This straightforward method is popularly used in many studies. However, it makes inefficient use of the data because a significant amount of data is not used for training the classifier [119]. In addition, it may be impractical in many real-world datasets that have a small number of instances, which is common in high-dimensional data such as gene expression data.

To deal with small sample size datasets, k-fold cross-validation (CV) is usually used. In this method, a dataset is randomly divided into k mutually exclusive subsets or folds. The inducer is trained k times. Each time, a different fold is used as the test set and the remaining folds form the training set. The results from k times will be averaged to produce a single estimate of classification performance. The advantage of this method is that all instances are used to assess the performance of a learning system. Each instance is also ensured to participate in testing exactly once. To maintain the performance stability of an inducer across different runs with different training and test sets, stratified k-fold CV is used. In this method, instances are randomly selected in such a way that the class distribution of each fold is the same as that of the original dataset. Leave-one-out-cross-validation (LOOCV) is a special case of k-fold CV where k is equal to the total number of instances in the dataset. This means that in each time, only one instance is used to test the classifier learnt from all the other instances.

0.632 bootstrap [67] is another resampling method that was proposed for small sample size datasets. Given a dataset of size n, a bootstrap sample is created by sampling with replacement n instances uniformly from the data. In this way, a bootstrap sample of size n contains on average $0.632 * n$ distinct

instances. The classifier is trained on the bootstrap sample and tested on the left-out instances. This process is independently replicated for B times, where B should be between 25 and 200 [67]. This replication requires a high computational time, which is one major drawback of this method. The final accuracy is calculated based on both training and testing accuracies. Therefore, this method fails to give the expected result for classifiers that have a perfect memory such as k-Nearest Neighbour, which always gives 100% accuracy on the training set.

2.1.2 Classification Algorithms

The past decades witnessed a significant development of ML algorithms including classification algorithms. This section presents some commonly used ones including k-Nearest Neighbour, Decision Tree, Support Vector Machines, and Naïve Bayes. Other classification algorithms, such as neural networks, random forest can be seen from [186].

k-Nearest Neighbour (KNN) [53] is a type of instance-based learning, which does not perform any explicit induction or learning process. KNN classifies a new instance based on its k nearest instances in the training set, where k is a positive and user predefined number. To determine k nearest neighbours, all the distances from the test instance to each instance of the training set are calculated. Then, the test instance is classified by a majority vote among these k nearest neighbours. Different proximity measures can be used to calculate the distance between instances such as Euclidean distance for continuous values and Hamming distance for discrete values. KNN is simple but works well in practice. It is also a non-parametric classifier which means no assumption about the probability distribution of the underlying data is needed. However, it is expensive in terms of time and memory due to the need of distance calculation from the query instance to all training instances. Furthermore, because the classification decisions are made locally, it is quite susceptible to noise, especially when k is small. Appropriate proximity measure and preprocessing steps have to be chosen for KNN to

produce good predictions.

Decision tree (DT) is another non-parametric classifier that is used for nominal, numeric, or mixture data [160]. A classifier in this method is presented as a tree in which each inner node is a decision stump or a split point and each leaf node is a class label. A query instance will traverse through the tree by testing its input values against the decision stumps and finally reaches its class label. To build a DT, a learning algorithm carries out a heuristic-based search using information gain as a criterion to choose the best split feature for each node. The best feature is the one that can split instances into separate groups that are as high homogeneous as possible. Different impurity measures were used to implement different DT algorithms such as C4.5 with information gain, CART with Ghini index, and CHAID with Chi-squared test. DT learning is computationally inexpensive and robust to noise. Moreover, the learnt classifier can be translated into comprehensible rules. However, since only one feature is tested in a node, DT does not perform well for cases where the boundary between classes are not parallel with coordinate axes [196]. These cases usually happen in problems that have two-way or multi-way relationships among features.

Support vector machine (SVM) [52] is a classification technique for continuous data that is rooted from statistical learning theory. SVM learns to find optimal hyperplanes in a higher dimensional space, which have maximal margins to their nearest instances of different classes. The original SVM method can only work with binary problems. Different versions of SVMs have been proposed for multi-class problems such as SVM one versus one, one versus rest methods [207]. SVM has shown promising empirical results in many practical applications. However, users must provide the type of the kernel function to use. Another drawback of SVM is its high computation time, especially when the number of dimensions is high.

Bayesian classifier is an approach to modelling probabilistic relationships between the feature set and the class variable. Naïve Bayes (NB) classifier is a simple implementation of Bayesian classifier where the class-conditional

probability is estimated by assuming that features or attributes are conditionally independent given the class label [160]. Although NB has been shown to be competitive with DT [156], its performance may be degraded in problems where assumption of features being conditionally independent given the class labels cannot be held.

2.1.3 Performance Measures

Prediction performance of a learnt classifier needs to be evaluated by some measures. The most popular measures are classification accuracy or error rate. While classification accuracy denotes the proportion of instances that are correctly classified to its category, error rate indicates the proportion of incorrectly classified instances. There are two types of errors: false negative and false positive. A false negative error is a case of misclassifying an instance belonging to the positive category, e.g. tumour, to the negative category, e.g. normal. In contrast, a false positive error is a case where an instance belonging to the negative class is misclassified into the positive class. In some cases, false negative errors may result in serious problems, e.g. patients of these cases will not be well treated. Therefore, more specific metrics are defined to show the relative frequencies of these errors. Table 2.1 shows the confusion matrix which defines four different relative frequencies of errors [186]. Receiver operating characteristic (ROC) curve and Area Under ROC (AUC) are popular methods used to display the trade-off between true positive and false positive rate. This method is applicable when there is a vary threshold in the classifier.

Unbalanced data is another issue that should be considered in evaluating the classification performance of a classifier. In these datasets, instances are not equally distributed into different classes. In a binary classification, if one class has more instances than the other, it is called the majority class. The other is the minority class. In this case, a high percentage of correctly identified instances may be contributed from the majority class only. Therefore, it cannot reflect the ability of the classifier in classifying instances

Table 2.1: A confusion matrix for a binary-class problem [186]

		Predicted Class	
		+	−
Actual Class	+	True Positive (TP)	False Negative (FN)
	−	False Positive (FP)	True Negative (TN)

from the minority class. To address this problem, a weighted average or balanced accuracy is proposed [187] as shown in Equation (2.1) where ω_i and TPR_i are the weight and the true positive rate of class i. Note that $\sum_{i=1}^{c} \omega_i = 1$. Therefore, if all classes are equally important, ω_i can be set to $1/c$.

$$balanced\_accuracy = \sum_{i=1}^{c} \omega_i * TPR_i \tag{2.1}$$

It is also important to note that in case of unbalanced data, creating training and test sets by splitting the whole dataset should consider stratification to maintain the original class distribution in the training and test sets. Otherwise, the classification performance may not reflect the effectiveness of the learned classifier when it is applied to unseen data in the future.

In general, classification performance of a classifier depends not only on the power of the learning algorithm but also the quality of the data. Therefore, data preprocessing is usually required to guarantee the successful application of a learning method to a real-world problem.

2.1.4 Discretisation

Discretisation is a data preprocessing technique that is used to transform features with continuous values into discrete, nominal, or ordinal values. It is often a crucial data preprocessing technique in ML because of several factors. First of all, many learning algorithms are applicable to or efficient on discrete data only. For instance, discretisation is either an embedded or external process required to discretise data into nominal values before applying DT or

NB. Furthermore, via discretisation, minor fluctuations or possible noise in the data can be ignored. In this way, discretisation helps learning algorithms improve their effectiveness and efficiency [65]. Last but not least, since discrete data is more compact than continuous data, it requires less memory and thus improving the efficiency of learning algorithms [31, 73].

Discretisation is a topic with a long research history. Many discretisation methods with different strategies have been proposed in this area. However, all of them share the same purpose which is to determine cut-points to partition features values into discrete values. Cut-points or split-points are real values within the range of the feature's values that are used to partition that range into several intervals. Existing discretisation methods can be categorised using different criteria [79, 123, 139]. In *direct* methods, intervals are generated based on a predefined parameter. On the other hand, *incremental* methods recursively split (or merge) intervals until some criterion is met. They are also known as *top-down* or *bottom-up* approaches, respectively. A discretisation method is *supervised* or *unsupervised* depending on whether class labels are used in the discretisation process or not. It is said to be *global* if the entire instance space is used in each discretisation step, or *local* if each discretisation step just uses a subset of instances. A method also belongs to *univariate* or *multivariate* depending on whether features are discretised individually or simultaneously taking into account interactions between features [79, 123].

Equal-width and equal-frequency (or equal-depth) binning are two simple unsupervised methods. They discretise features into a predefined number of m intervals with the same width (so-called equal-width) or the same number of instances (so-called equal-frequency), respectively. These simple methods are easy to implement but sensitive to the value of m which is usually difficult to determine, especially when features are not uniformly-distributed or contain outliers [38].

Using the class labels as a guide in searching cut-points, supervised discretisation usually performs better than the unsupervised counterpart. 1R [96] defines cut-points as feature values lying at the boundary of different

classes. Each bin needs to have at least a predefined number of instances in the same class except for the last bin. In addition to different searching techniques, different evaluation measures are proposed such as classification error rates [75, 197], information gain [70, 85], and statistical measures [37, 260]. More comprehensive reviews can be found in [79, 123, 139, 150].

Among supervised methods, minimum description length (MDL) proposed by Fayyad and Irani [70] is one of the most popular methods. It is an entropy-based incremental splitting discretisation method aiming to find the best splits so that the discretised features are as pure as possible in terms of class labels. This means that the majority of the values in one interval are preferred to have the same class label. If entropy $E(S)$ is used to measure the purity level of a set S, then according to this criterion, the cut-point with the highest information gain is the best one. Equation (2.2) is used to calculate information gain of a cut-point T for feature F given S as the set of feature F values. S_1 and S_2 are subsets of S after partitions.

$$Gain(T, F; S) = E(S) - \frac{|S_1|}{|S|}E(S_1) - \frac{|S_2|}{|S|}E(S_2) \tag{2.2}$$

Furthermore, not all cut-points are useful, especially when the features are noisy or irrelevant to the class label. Therefore, Fayyad and Irani [70] proposed to use the minimum description length principle (MDLP) as a criterion to accept a cut-point. Note that in this thesis, we use the abbreviation of MDLP to represent the minimum description length principle and MDL to represent the discretisation method proposed in [70].

$$Gain(T, F; S) > \frac{log_2(|S| - 1)}{|S|} + \frac{\delta(T, F; S)}{|S|} \tag{2.3}$$

where

$$\delta(T, F; S) = log_2(3^{k_S} - 2) - [k_S E(S) - k_{S_1} E(S_1) - k_{S_2} E(S_2)] \tag{2.4}$$

given $|S|$ as the size of set S, $E(S)$ as the entropy of S, and k_S as the number of classes in S.

According to MDLP, as shown in Equation (2.3), a cut-point T is only accepted if its information gain is greater than the cost of encoding the cut-point T and the classes of the instances in the intervals induced by T. If the

cut-point T induces impure intervals, the sum of these two costs may exceed its information gain. As shown in Equation (2.4), $\delta(T, F; S)$ will become larger if the entropy values of $E(S_1)$ and $E(S_2)$ are larger; in other words, when S_1 and S_2 are less pure. In noisy or irrelevant features, the cut-points usually cannot induce intervals that are pure enough to satisfy the condition in Equation (2.3). Therefore, using MDLP can deal with noise or irrelevant data.

2.2 Feature Manipulation

In this thesis, feature manipulation is a general term for feature selection and feature construction, which are also known as dimensionality reduction methods. This section introduces their definitions, the overview of these systems and details of their processes.

2.2.1 Feature Selection

In real-world situations, many candidate features are introduced to better represent the domain because which features are relevant is often unknown. Unfortunately, the presence of irrelevant and redundant features may obscure the effect of using the considered relevant features to show the hidden pattern of the data, and thereby reduce the representativeness of the whole feature set [267]. This can hinder learning algorithms from inducing a good classifier.

Feature selection is ideally defined as finding the smallest subset that is necessary and sufficient to describe the target concept [118]. Classically, feature selection is to select L features from N original features in such a way that the value of a subset evaluation function is optimised over all subsets of size L [169]. When considering the class distribution approximation, feature selection is defined as selecting a small subset while maintaining as much as possible the class distribution similarity between the original dataset and the dataset using only selected features [121]. On the other hand, when focusing on predictive accuracy improvement, feature selection is defined as choosing

a subset of features that can improve the classification accuracy or reduce the number of features without significantly decreasing the classification accuracy of the classifier learnt from the selected subset [121]. Although feature selection is differently defined by many researchers based on different viewpoints, most of them are similar in intuition and/or content [57].

With the same purpose, feature ranking or weighting is a relaxed version of feature selection. These methods rank or weight features based on their relevance using a specific measure. A predefined number of top-ranked features are then selected to form the final feature subset.

In general, feature selection is a data preprocessing technique searching for a minimal feature subset by eliminating irrelevant and redundant features. Its ultimate goal is to reduce the dimensionality and thus the size of the data and the running time of the learning algorithm, increase or not significantly decrease the classification accuracy, and simplify the learnt model [58].

2.2.2 Feature Construction

Many classification algorithms, especially those using symbolic classifiers such as decision trees and decision rules, cannot induce good classifiers when being confronted by problems where feature interactions exist due to their inability to construct new features from original features [126]. This limitation can be partially solved by feature construction.

Feature construction aims to automatically transform the original representation space into a new one that better represents the target concept [101]. Specifically, in the context of this thesis, feature construction is a process which combines features selected from the original features to create new high-level features in order to improve classification accuracy [267]. The newly constructed features are in fact mathematical expressions of the original features. They can be used to replace or augment the original feature set to improve the performance of the learning algorithms [125].

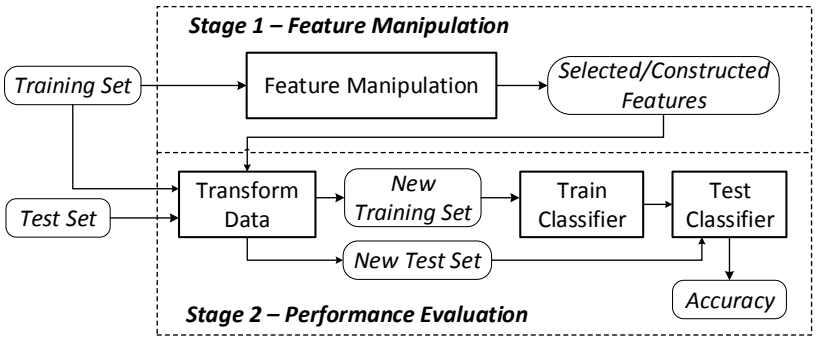


Figure 2.1: The typical structure of a feature manipulation system (adapted from [141]).

2.2.3 A Typical Feature Manipulation System

A typical feature manipulation system usually has two stages as shown in Figure 2.1, which is adapted from [141]. The first stage is feature manipulation, which takes training data as input and outputs the best feature set. Details of this stage will be described in the next section. The returned feature set will then be examined in the next stage to test if the feature selection method has been able to eliminate irrelevant and redundant features, or if the feature construction method has been able to create new features with higher discriminating power. In case of feature selection, if prior knowledge about the relevant features is available as in synthetic datasets, then a direct comparison can be made easily. However, in real-world applications, these prior knowledge is usually unknown. Therefore, we usually rely on some indirect methods by comparing the change in performance of a specific classifier from using all features to only the selected or constructed features.

In general, the performance evaluation in Stage 2 has three steps:

1. Transforming data: By removing the unselected features or constructing new features from the original training set and test set, this step outputs the corresponding new training and test sets.

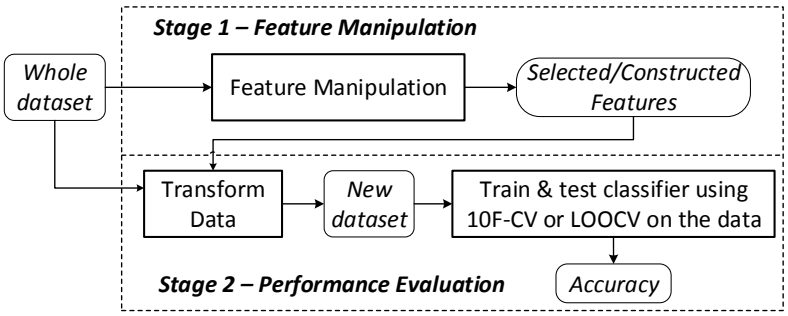


Figure 2.2: Structure of a feature manipulation system with feature selection bias.

2. Training a classifier: The new training set is fed into the learning algorithm to induce a new classifier.

3. Testing the classifier: The performance of the learnt classifier will be evaluated on the new test set and measured by using a performance measure (mentioned in Section 2.1.3). The results are then analysed to see if the selected or constructed features improve the performance of the learnt classifier.

2.2.3.1 Feature Selection Bias Issue

When evaluating the performance of a feature manipulation method, it is important to avoid an issue called feature selection bias [17], where the whole dataset is used in Stage 1 to train the feature manipulation system as shown in Figure 2.2. While this issue did not happen in many studies where datasets have a large number of instances to be divided into training and test sets, it can be seen in many studies where datasets have a small number of instances [2, 5, 13, 20, 50, 106, 158, 161, 162, 205, 210, 261].

According to Ambroise and McLachlan [17], there is a feature selection bias issue in these studies since there is no unseen data to test the generality of the selected features. Therefore, one cannot claim that the selected features can be used for future unseen data.

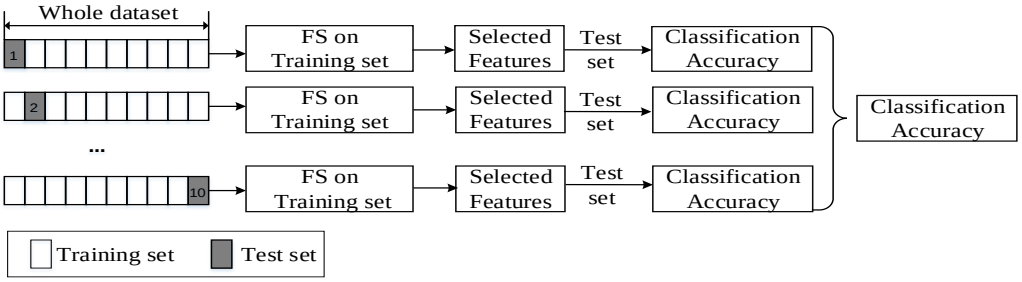


Figure 2.3: Structure of a 10-fold cross-validation without feature selection bias.

To avoid feature selection bias in experiments on datasets with small number of instances, 10-fold cross-validation (CV) is usually used to create training and test sets [217]. The experiment setting, in this case, should follow the structure shown in Figure 2.3, where the test fold is kept out of the feature selection process. The average accuracy of the 10 runs will be used to evaluate the performance of the method. The same setting should also be applied to feature construction methods. All the experiments in this thesis will follow this setting.

2.2.4 Feature Manipulation Process

Figure 2.4 presents the details of the feature manipulation process. It is a procedure that takes training data as input, loops until a predefined stopping criterion is met and outputs the best feature set. The loop contains three steps:

1. Feature set generation:

    - For feature selection, a discovery procedure is used to generate new feature subsets from the original set. This procedure starts with an empty subset, full feature set, or a random subset of features [131] and then applies a search strategy to search for the optimal

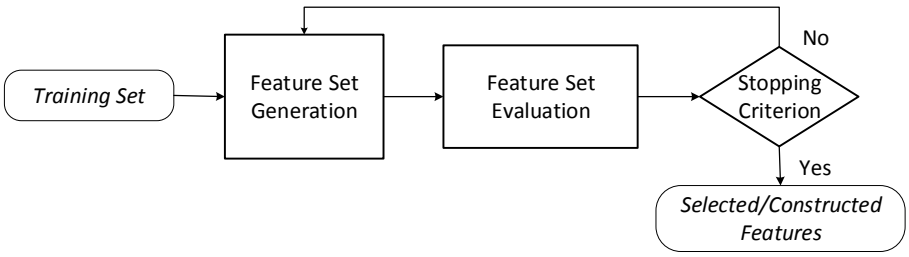


Figure 2.4: Feature manipulation process.

feature subset.

- For feature construction, before constructing new features, a discovery procedure is used to select subsets of features as in feature selection. The selected features will then be combined using some mathematical operators to create new features. The key challenge of this procedure is choosing appropriate features and operators as well as an appropriate way to combine them so that the created features have higher discriminating ability than the original ones.

Because feature selection and feature construction are combinatorial optimisation problems, using exhaustive search to find an optimal solution can be computationally impractical. Therefore, many heuristics have been proposed to search for a suboptimal solution within a reasonable time. Different search strategies will be discussed in Section 2.2.6.

2. Feature set evaluation: Generated feature sets are evaluated by an evaluation function. If better solutions are found, the best feature set is updated. Using different evaluation functions or criteria may lead to different optimal sets. Therefore, designing a good criterion to measure the goodness of a feature set is one of the main concerns in developing a feature manipulation system. Section 2.2.5 presents different approaches to creating the evaluation criteria.

3. Stopping criterion: Conditions to stop the feature manipulation loop

can be based on the generation or evaluation step. Examples of the former can be whether a predefined number of features is selected or if a given number of iterations is reached. The latter criteria can be whether an optimal solution is found. When this criterion is met, the best feature set will be returned.

2.2.5 Feature Set Evaluation

Feature set evaluation is an important component of a feature manipulation algorithm because it guides the search process towards optimal feature sets. Therefore, an optimal feature set is always relative to the evaluation function used. In general, the aim of an evaluation function is to measure the discriminating ability of the feature set in distinguishing instances of different classes. Based on the way a feature set is evaluated, feature manipulation methods are classified into three typical approaches: wrapper, filter and embedded approaches [91, 203].

2.2.5.1 Wrapper Approaches

In wrapper methods, the prediction performance of a learnt classifier is used as a criterion to evaluate the feature set. The evaluation process is considered as a black box so that any learning algorithm can be used such as KNN, DT and SVMs. This black box performs the same steps as performance evaluation stage in Figure 2.2 but applying to the training set instead. Therefore, the computational time of wrapper methods is usually more expensive than other approaches. Moreover, the best feature set evaluated by one learning algorithm, e.g. DT, may not improve the prediction performance of another learning algorithm, e.g. SVM. Therefore, this approach is claimed to be less general than filter methods [42, 259]. However, as compared to the filter approach, the wrapper approach usually achieves better classification performance.

2.2.5.2 Filter Approaches

Instead of using classification performance, filter methods evaluate the feature set based on the intrinsic characteristics of the data. Different types of measures [58] have been proposed:

1. Distance measures: They are also called separability, divergence or discriminant measures. For a binary-class problem, according to this measure, a feature is said to be better or more relevant than other features if it can induce a higher difference between the two-class conditional probability than other features. Examples are Euclidean distance and Manhattan distance.

2. Information measures: Mutual information is a popular measure showing how much information that two variables share with each other. Information gain of a feature is defined as the mutual information of itself and the class label. Therefore, a feature is more relevant if it has higher information gain. On the other hand, two features are redundant if their mutual information is high.

3. Dependence measures: The ability to predict the value of a feature from the value of another feature is quantified in these measures. They are also known as correlation measures. The higher a feature is correlated to the class label, the higher it is relevant to the target concept. When applying this measure to two features, it indicates the degree of redundancy between them. Measures of this type can be divided into distance and information measures. However, they still form a separate category due to their different viewpoint.

4. Consistency measures: these measures aim at finding the smallest subset that can provide a consistent labelling of all instances. Therefore, these measures heavily rely on the training data and are usually biased to the minimal subset.

Evaluating feature subsets based on the above measures, a filter approach has the advantages of low computational cost and high generality. However, because prediction performance of the selected features on a learning algorithm is not considered in these methods, they usually have lower classification performance than wrapper methods on a particular learning algorithm [112].

2.2.5.3 Embedded Approaches

In embedded methods, the learning process simultaneously learns a classifier and chooses a subset of features. Therefore, feature selection is intrinsic to the learning algorithms such as in decision trees or support vector machines [91]. In comparison to the wrapper approach, this approach has lower computational time. Nevertheless, the performance of the selected features highly depends on the learning algorithm.

In general, wrapper and embedded methods usually achieve better classification performance than filter methods, however with the price of longer computation time and less generalisation.

2.2.6 Search Strategies

Feature set generation is another critical component of a feature selection algorithm, especially on high-dimensional data. Generating new feature sets is actually the process of searching through the feature space. With 2^N possible feature subsets in feature selection and even a larger number of candidate solutions in feature construction, a good search strategy is very important in finding the optimal or near-optimal solutions as fast as possible. Current search strategies can be divided into following categories [145]:

1. Complete search: It guarantees to find the optimal subset according to the evaluation criterion used. An exhaustive search such as FOCUS [16] is complete. However, the computational cost of exhaustive methods is too high to use with even small problems with 30 features since the size of the search space is 2^{30} [121]. Therefore, different heuristic functions

are used to reduce the search cost without losing the chances of finding the optimal subset. Branch and bound [169] is a typical method.

2. Sequential search: The generation process iteratively adds a feature from the remaining features or removes a feature from the currently selected subset, producing sequential forward selection (SFS) [240] or sequential backward selection (SBS) [153], respectively. These hill climbing methods give up completeness and therefore risk losing optimal sets. Many variances of this simple process have been proposed but the incremental search is their basic principle.

3. Stochastic search: Starting with a randomly generated candidate solution, the generation procedure may generate the next candidate solution in a completely random manner [143, 144] or using some heuristics. Although the former approach may avoid local optima, it is too slow to converge in a large search space. This can be avoided in the latter approach which is represented by simulated annealing algorithm and evolutionary computation techniques. Different from simulated annealing algorithm and other stochastic searches, evolutionary computation techniques are population-based, enabling them to conduct a global and parallel search. Details about these techniques will be discussed in the next section.

2.2.7 Feature Clustering

Clustering or cluster analysis is one of the main tasks in exploratory data mining. It aims to group similar objects into the same group or cluster. Literature has proposed different clustering algorithms using different measures to evaluate the similarity between objects as well as different ways of grouping objects [244]. Clustering has been used for decades as an unsupervised task where instances are grouped into clusters based on the similarity between them [245].

In machine learning and data mining, the "clustering" terminology is usually meant *instance clustering* or instance grouping. Recently, clustering techniques have been proposed to group similar features, thus called *feature clustering*, to achieve feature selection [36]. Similar features are grouped into the same cluster. Then one or more representative features from each cluster were used to form final subsets. Different clustering techniques have been proposed for feature selection such as statistical clustering [129, 180] and minimum spanning trees [216].

2.3 Evolutionary Computation

Evolutionary Computation (EC) is an area of artificial intelligence that comprises of nature-inspired algorithms. All of these algorithms are population-based search. A population of candidate solutions is maintained and evolved using fitness (goodness) as a guide to search for better solutions. EC methods can be broadly divided into two main categories: evolutionary algorithms and swarm intelligence [21].

Evolutionary algorithms refer to algorithms that apply Darwinian principles to search for an optimal solution in the way that evolution searches for optimal species. These algorithms use genetic operators such as reproduction, crossover and mutation to evolve better solutions from the current ones. Typical algorithms of this class are genetic algorithms (GAs) [95] and genetic programming (GP) [124]. The main difference between GA and GP is the representation. While GA individuals are represented as chromosomes which are fixed-length strings of values, GP individuals can have variable lengths with different structures such as tree and graph.

Swarm Intelligence is another EC category that is inspired by the collective intelligence of social animals. The intelligence of these systems is based on simple interactions among individuals and between individuals and the environment [33]. Particle swarm optimisation (PSO) [66] and ant colony optimisation (ACO) [64] are typical examples of this class. While PSO

simulates birds' behaviours, ACO is inspired by the special communication system using pheromone between ants about favourable paths to food. The shortest path will be the one that has most pheromone remaining after a given time.

Although numerous EC techniques have been proposed for feature selection and feature construction in the literature [69, 256], the remainder of this section will focus on the two techniques to be used in this thesis, which are PSO and GP.

2.3.1 Particle Swarm Optimisation

Particle swarm optimisation (PSO) is an EC technique developed by Kennedy and Eberhart [66], which is inspired by social behaviours found in birds flocking. In PSO, a swarm consists of many individuals called particles communicating with each other to search for optimal solutions when iterating from one position to another according to their velocity. The position of a particle encodes a candidate solution of the problem. It is actually an N-dimension vector of numerical values, where N is the dimensionality of the problem. A similar vector is used to record the velocity of a particle, which indicates the speed and direction that it should move in each dimension.

During the search process, each particle also records the best position it has explored so far, also known as personal best or *pbest*, and communicate its *pbest* to its neighbours, which can be defined based on a certain topology such as fully connected, star or ring to get the neighbourhood best. In the fully connected topology, one particle will communicate with all the other particles, therefore, the neighbourhood best is called global best or *gbest*. *pbest* and *gbest* are used to update its velocity, which is then used to update its position. In this way, particles move in the search space towards fruitful areas to find better solutions.

2.3.1.1 Standard Particle Swarm Optimisation

PSO was first introduced in a continuous version; therefore, it is also called continuous PSO. In continuous PSO, the position is a vector of continuous values. Formulae (2.5) and (2.6) are used to update the velocity and position of each particle, respectively.

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i}^{t} * (p_{id}^{t} - x_{id}^{t}) + c_2 * r_{2i}^{t} * (p_{gd}^{t} - x_{id}^{t}) \tag{2.5}$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{2.6}$$

where v_{id}^{t} and x_{id}^{t} are velocity and position of the i^{th} particle in dimension d at time t, respectively. p_{id}^{t} and p_{gd}^{t} are *pbest* and *gbest* positions in dimension d at time t. r_{1i}^{t} and r_{2i}^{t} are uniformly distributed random values generated at time t. c_1 and c_2 are acceleration constants that determine the type of trajectory the particle travels, thus they are important to control the searching behaviour of the particle [235]. The inertia weight w is used to control the impact of the last velocity to the current velocity. The velocity values are usually limited by a predefined maximum velocity, v_{max} to the range $[-v_{max}, v_{max}]$ to maintain particles staying in the search space.

2.3.1.2 Binary Particle Swarm Optimisation

To optimise discrete problems, Kennedy and Eberhart [116] developed a binary PSO (BPSO) algorithm in which the position of each particle is encoded by a binary string. Equation (2.5) is still applied to update the velocity, in which x_{id}^{t}, p_{id}^{t} and p_{gd}^{t} are restricted to 0 or 1. However, unlike in continuous PSO, the velocity in BPSO represents the probability of an element in the position taking value 1. Therefore, a sigmoid function $s(v_{id})$ is introduced to transform v_{id} into the range of 0 and 1. After that, BPSO updates the position of the particle according to the following formulae:

$$x_{id}^{t+1} = \begin{cases} 1, & rand() < \frac{1}{1+e^{-v_{id}}} \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

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

2.3.1.3 Bare-bones Particle Swarm Optimisation

As can be seen in Equations (2.5) and (2.6) that PSO operates by sampling points in the search space. It uses *pbest* and *gbest* which are the discovered knowledge to guide the sampling. In [115], Kennedy investigated the trajectory of a single particle in a standard PSO where *pbest* and *gbest* were set as constants. All of its visiting positions after a million iterations were plotted. The obtained histogram is a tidy bell curve centred midway between these constant positions of *pbest* and *gbest*. The result suggests that trajectory of a particle is determined by the difference between its *pbest* and *gbest*. Therefore, the step size of a particle's movement should be based on these best positions. This finding resulted in a new PSO method called bare-bones PSO (BBPSO). In this method, particle's position is sampled using a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with the mean μ and standard deviation σ as shown in Equation (2.8).

$$x_{id}^{t+1} = \begin{cases} \mathcal{N}(\mu, \sigma), & rand() < 0.5 \\ p_{id}^{t}, & \text{otherwise} \end{cases} \tag{2.8}$$

where μ is the centre of *pbest* and *gbest* and σ is the absolute difference between these best positions. The $rand()$ function is used to speed up convergence by retaining the previous best position *pbest*. By using the Gaussian number generator, BBPSO avoids the need to optimise the velocity vector and the delay of position adaptation.

2.3.1.4 Main Steps in Particle Swarm Optimisation

Despite the difference in position values, continuous, binary and bare-bones PSO follow the same steps:

1. Randomly initialise position and velocity for each particle.

2. Iteratively perform the following sub-steps until the stopping criterion is met:

   (a) Evaluate the fitness of each particle using a predefined fitness function.

   (b) If the fitness of a particle is better than *pbest* then update *pbest*.

   (c) If the fitness of any *pbest* is better than *gbest* then update *gbest*.

   (d) Update velocity of each particle using Equation (2.5) *(this step is skipped in bare-bones PSO)*.

   (e) Update position using Equation (2.6) for continuous PSO, Equation (2.7) for binary PSO, and Equation (2.8) for bare-bones PSO.

3. Return the best solution which is the position of the best particle (*gbest*).

2.3.2 Genetic Programming

Genetic programming (GP) [124] is an EC method that can automatically learn a set of computer programs for a particular task. Starting with a population of candidate solutions, which are represented as individual programs, GP follows Darwinian evolution principles to evolve better solutions by using genetic operators such as crossover and mutation. Since the shape and length of the final program are normally unknown, GP individuals usually represent programs as trees with various lengths. There are also other common representations such as graph-based GP, linear-based GP, and grammar-based GP [190]. However, the tree structure provides a natural representation for feature construction. Therefore, this thesis focuses on tree-based GP.

In tree-based GP, each program is a tree consisting of two types of nodes. Internal nodes are operators or functions {e.g. $+, -, *, /, power(x, y), square\_root(x)$} with different numbers of arguments. Leaf nodes are constants or variables serving as arguments for internal nodes. The set of all possible operators or functions that can be used in an internal

node is called the *function set*. Similarly, all possible variables form the *terminal set*. The size of a GP tree is usually limited by a maximum depth which is the longest path from the root to a leaf node.

Constraints have to be predefined to make sure that any generated program is a valid, executable program. For example, function $square\_root(x)$ cannot take a negative value. Function arguments can also have different types. For example, function if may take three arguments: a condition that is of boolean type and two numerical arguments to return when the condition is true and false.

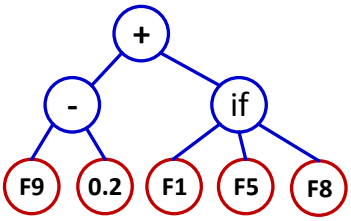


Figure 2.5: Example of single-tree GP representations.

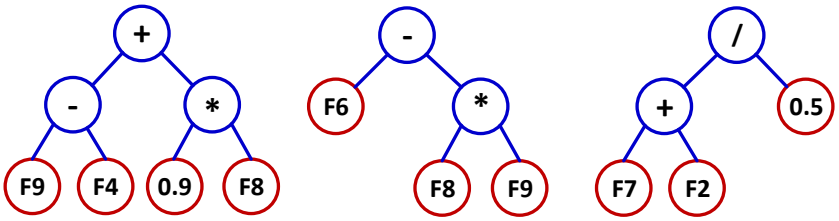


Figure 2.6: Example of multi-tree GP representations.

2.3.2.1 GP Tree-based Representation

A GP individual represents a candidate solution. It can be represented as a single tree (so called single-tree GP) or a set of trees (so called multi-tree GP). Figure 2.5 shows a simple example of the single-tree GP representation.

In this example, features F_1, F_5, F_8 and F_9 are selected to construct one new feature $CF : (F_9 - 0.2) + if(F1, F5, F8)$

Similarly, Figure 2.6 shows an example of a multi-tree GP individual that selects 6 features, namely $F_2, F_4, F_6, F_7, F_8, F_9$, to construct three new features:

- $CF_1 : (F_9 - F_4) + (0.9 * F_8)$,
- $CF_2 : (F_6 - (F_8 * F_9)$, and
- $CF_3 : (F_7 + F_2)/0.5$.

2.3.2.2 Genetic Programming Algorithm

As a population-based evolutionary algorithms, GP follows these steps:

1. Initialise GP trees by randomly selecting operators from function set for internal nodes and constants or variables from terminal set for leaf nodes (see Section 2.3.2.3).

2. Iteratively perform the following sub-steps until a stopping criterion is met:

   (a) Evaluation: Each GP tree is executed and its fitness is calculated based on a predefined fitness function.

   (b) Selection: Select one or two individual program(s) from the population with a probability based on fitness to participate in the evolution step.

   (c) Evolution: Create new individuals for the new population using the following genetic operations with specific probabilities:

      i. *Reproduction/Elitism:* Copy the selected individuals to the new population

      ii. *Crossover:* Create new offspring programs by recombining randomly chosen parts from two selected programs.

iii. *Mutation:* Create a new offspring program by randomly mutating a randomly chosen part of one selected program.

3. Return the program with the highest fitness as the best solution.

2.3.2.3 Initialisation Methods - Program Generation

Full and grow [124] are popular methods used to initialise or generate GP trees. The full method produces entirely balanced trees with all terminal nodes located at the predefined maximum depth. In contrast, terminal nodes can appear at any depth level in the grow method. Although using the same maximum depth for all trees, both methods can generate trees with very different sizes (e.g the total number of nodes). Therefore, a combination of full and grow called ramped half-and-half is widely used to diverse the initial population with individuals of various depths, lengths and shapes.

2.3.2.4 Selection Methods

Imitating the natural selection process, good individuals are more likely to be chosen to generate new individuals. Two popular selection methods are the proportional selection (or roulette wheel) and the tournament selection. In the proportional selection, individuals are randomly selected based on the probability determined by their fitness. One disadvantage of this method is that individuals with small fitness may not be selected. However, they might contain good building blocks, which may form better solutions. To solve this problem, the tournament selection randomly samples a number of individuals based on the tournament size, and then selects the best one from them. The smaller the tournament size is, the higher chance the bad individuals enter the breeding process.

2.3.2.5 Genetic Operators

GP evolves new programs by applying genetic operators such as crossover, mutation and reproduction (elitism) to the current programs. These operators

are applied with different probabilities defined as crossover rate, mutation rate and reproduction rate. Beside these basic operators, GP allows users to define new operators, which makes GP become a powerful and flexible EC technique.

Crossover is an operator applied to two selected individuals (parent trees). The most common form of crossover is subtree crossover [124]. In each parent tree, a random node is chosen as the crossover point. The two subtrees rooted at these two points will be exchanged to form two new trees. If two selected crossover points are leave nodes, crossover may not create significant new trees. To avoid this problem, Koza [124] recommended to use a probability of 90% to choose internal nodes and 10% for leaf nodes.

Mutation is used to generate a new tree by changing a chosen parent at a random node. Two popular types of mutation are the subtree mutation and the point mutation. While the former replaces the subtree rooted at the chosen node with a newly generated tree, the latter replaces the chosen node with an equivalent node from either function or terminal set. The main difference between these two methods is that the latter preserves the shape of the parent tree.

Reproduction simply copies the individual selected by the selection method (e.g. tournament) to the new population. Elitism is a special case of reproduction where only the best individuals are copied. In this way, GP ensures that the best individuals will not be lost through the evolutionary process.

2.4 Related Work

Feature manipulation is a topic with a long research history. Providing a full survey of this field is out of the scope of this thesis. This section will review typical approaches to feature selection and construction, especially on high-dimensional data, using either EC or conventional techniques. Some of the methods described in this section will be used to compared with the newly proposed methods in this thesis.

2.4.1 Feature Selection Using Conventional Techniques

This section introduces the conventional feature selection methods which can be divided into subsections of wrapper, filter and embedded approaches. While the embedded approaches follow their own ways to search and evaluate features, filter and wrapper approaches can use the same search strategies to create feature subsets. Therefore, in order to have a concise review, wrapper and filter methods will be presented together in different search strategies. Finally, feature selection via discretisation is introduced.

2.4.1.1 Sequential Search Based Feature Selection

Although exhaustive search such as in FOCUS [16] can definitely find a global optimal feature subset, its high computational cost makes it inapplicable even to datasets with tens of features. *Sequential search* is a popular heuristic, which uses a greedy technique to speed up the search process with the risk of loosening the optimality guarantee.

Two typical sequential methods are sequential forward selection (SFS) [240] and sequential backward selection (SBS) [153]. SFS starts with an empty feature subset, then gradually adds features until the classification accuracy is not improved. At each iteration, it chooses the feature that can produce the best performance when combined with the selected feature subset. Using a similar principle, SBS starts with a full set of features and gradually removes features until the classification accuracy is not improved. The backward manner enables SBS to better handle feature interaction than SFS. However, in terms of computation time, SFS is much faster than SBS. Therefore, SFS become a popular technique used in many feature selection methods. Not only used in wrapper methods with KNN as in [240], or with neural network as in [152], SFS was also used in filter methods such as SetCover [56] where *consistency* is used to evaluate the feature subset. Features are gradually added until the inconsistency measure is smaller than a predefined threshold.

Using a greedy hill-climbing search strategy to consider one feature at a time, both forward and backward selection approaches suffer from the so-called "nesting effect" because a feature which is selected or removed cannot be removed or selected at a later stage. This is the reason why sequential search based feature selection methods failed to find the optimal solution, such as forward-based feature selection methods in SetCover [56] failed to find the optimal solution for the CorrAL dataset, which has four relevant features and one feature that is 75% correlated to the class. Although this correlated feature is not relevant, it is always selected by SFS as the first feature.

To avoid this "nesting effect", "plus-l-take-away-r" [219] applies SFS l times and then SBS r times. However, it is hard to determine appropriate values for l and r. Therefore, Pudil et al. [193] proposed dynamic forward and backtracking steps in two corresponding methods: sequential backward floating selection(SBFS) and sequential forward floating selection (SFFS). These floating search methods are claimed to be better than the static sequential methods, but they are still prone to getting stuck in local optima [265].

Instead of using hill-climbing search, correlation based feature selection (CFS) [94] uses best-first search strategy so that it can back track to explore the next best unexpanded subset. *Correlation-based* measure was used to bias towards feature subset that contains features that are highly correlated to the class (relevant) and uncorrelated to each other (non-redundant). The results showed that CFS can significantly reduce the number of features while maintaining or improving the performance of KNN, NB and DT. However, although best-first search can explore the entire search space to obtain the optimal solution, CFS is still trapped in local optima [32] when facing such class-correlated features as in the CorAL dataset, because its evaluation method prefers subsets with more features that are highly correlated to the class.

Derived from SFS, linear forward selection (LFS) [90] restricts the number of features to consider at each step. Therefore, LFS is computationally

less expensive than SFS while maintaining comparable accuracy of selected features. However, LFS starts with ranking features individually, which limits LFS performance in problems where complex feature interaction exists.

2.4.1.2 Feature Ranking / Feature Weighting

Although sequential search is more efficient than exhaustive search, applying it to high-dimensional data is still computationally expensive. *Feature ranking* or *feature weighting* is probably the most scalable approach to feature selection on these datasets. In these methods, features are individually ranked based on their degrees of relevance measured by a predefined measure. Then a predefined number of top-ranked features are used to form the final subset.

Many feature ranking methods were proposed in the literature using different measures to rank features such as mutual information, gain ratio, symmetric uncertainty, distance, t-Test, ANOVA and Chi-square. Readers are referred to [28, 32, 134, 181, 203] for a more comprehensive review. Among these methods, Relief [118] and its extension, ReliefF [122], are popular methods that use *distance* measures to evaluate the degree of feature relevance. In each iteration, ReliefF randomly samples an instance from the training set, determines its two nearest instances of the same and opposite class, which are called the "near-hit" and the "near-miss", respectively. The weight of each feature is updated based on the difference between the selected instance and its near-hit and near-miss. ReliefF is an efficient method with linear complexity to the number of features and samples, noise-tolerant, and can handle a certain level of feature interaction [199, 233]. Readers are referred to [232] for an expansive review of ReliefF-based algorithms.

In general, feature ranking methods are computationally efficient. They work on the assumption that features are independently relevant to the target concept. However, this assumption may not be hold since a feature that is not useful by itself can become useful with others [91]. In addition, choosing the right number of features that should be selected may require an extensive trial and error or domain knowledge, which may be unavailable, especially

in high-dimensional problems. Furthermore, the top-ranked features may be redundant [57], which may degrade the performance of the learning algorithm.

2.4.1.3 Two-stage Approaches

To overcome this problem, a *two-stage approach* was proposed where feature ranking is followed by a heuristic search to remove redundant features from the selected top-ranked features. The last decades witnessed an increasing amount of research following this approach to feature selection on high-dimensional data [63, 202, 216, 263]

In the fast correlation-based filter method (FCBF) [262, 263], symmetrical uncertainty (SU), a normalised version of mutual information, was used to measure the relevance of a feature and redundancy between two features. A predefined threshold was used to select relevant features to enter the next stage where features that were more correlated to the selected feature than to the class label will be removed. Experiments showed that FCBF selected the least number of features in a much faster running time than the compared algorithms. However, it can only work with discrete data and a threshold for choosing relevant features had to be predefined. Similarly, FAST [216] also used SU as a measure to remove irrelevant features in the first stage. Then, the remaining features were clustered into the different minimum spanning trees, which were then partitioned so that each tree contained only redundant features. The best feature in each tree was chosen to form the final subset. The author reported that FAST outperformed the compared methods in terms of classification accuracy, number of selected features, and running time on some datasets.

Minimum redundancy and maximum relevance (mRMR) [63] was also a two-stage feature selection method. While relevance measure was based on F-test for multi-class problems or t-test for binary-class problems, redundancy measure was based on correlation coefficient for continuous data or mutual information for discrete data. The proposed two-stage methods obtained better feature subsets than using feature ranking only. Similarly, feature

dispersion was used in the first stage to rank features in [72]. The top d features are examined in the second stage to eliminate redundant features that had absolute cosine with another feature higher than a predefined threshold. The results showed that the proposed method achieved better performance than the compared methods on some datasets.

Two-stage approaches also provide a natural framework for implementing a hybrid method where filter and wrapper can be used in one or both stages. In [243], the filter method first used an unconditional mixture model to assess the discriminating potential of each feature, then information gain to rank features. Finally, Markov blanket filtering was used to choose feature subsets which were evaluated using Gaussian, logistic regression or KNN as the wrapper method. Results on one dataset showed that the proposed method improved the performance of the learning algorithms. Similarly in the BIRS method proposed in [202], features were ranked based on a learning algorithm (wrapper BIRS), or SU (filter BIRS). In the second stage, the best feature was selected and subsequent features will be added if the resulting subset significantly improves the classification performance of the learning algorithm. Results showed that BIRS obtained a similar classification performance as the compared methods with small feature subsets. Similar approaches can also be found in [76, 100].

In general, two-stage methods usually start with ranking features, then scanning forward to eliminate redundant features using some measure to detect redundancy between pairs of features. With this setting, these methods usually fail to identify multiple feature interactions and remove class-correlated features [263, 271]. To avoid this issue, INTERACT [268] was proposed to remove features from the end of the list if the inconsistency rate difference between with and without using the feature is smaller than a predefined threshold. By using backward elimination, INTERACT was able to take feature interactions into account, obtaining similar or better classification performance than FCBF [262], CFS [94], ReliefF [118], and FOCUS [16] with smaller feature subsets. However, INTERACT may suffer from noisy data.

2.4.1.4 Embedded Approaches

Some classification algorithms implicitly perform feature selection while learning a model. Thanks to this ability, embedded feature selection methods have been proposed, e.g. SVM (SVM-RFE) [92], logistic regression (Lasso) [227], neural network [23], random forests [242], and decision tree [82].

Following the backward elimination approach, SVM-RFE [92] iteratively ranked features based on the weights returned by SVM and removed the worst one until reaching a predefined number of features. Results on two gene expression datasets showed that SVM-RFE obtained significant improvement over the baseline method. However, clean data without outliers played an essential role in the performance of SVM-RFE. Different approaches have been proposed to improve the performance of SVM-based feature ranking methods [151, 225, 247], especially on high-dimensional data. However, they still require users to set the number of features that should be selected.

Another popular family of embedded feature selection methods developed based on sparse logistic regression models was first proposed in Lasso [227] and followed by many methods including [224, 264, 273]. By simultaneously minimising the residual sum of square and the sum of the absolute values of the coefficients, these methods obtained both regularisation and feature selection. Results of these methods have shown their effectiveness and efficiency, especially on high-dimensional data. However, they often have assumptions about the probability distribution of the data. Readers are referred to [88] for a comprehensive study of this approach.

2.4.1.5 Feature Selection Via Discretisation

Discretisation is a preprocessing technique which aims to find a discrete representation of the data so that it contains enough information for the learning task while eliminating minor fluctuations or noise. Thanks to this characteristic, feature selection via discretisation has been proposed to select features based on the result of the discretisation process [142, 105].

Chi2 [142] is one of the first methods proposing feature selection via discretisation. It is a bottom-up discretisation method starting from intervals with only one feature value. Then adjacent intervals with the lowest χ^2 test result will be merged recursively until χ^2 values of all pairs exceed a threshold. This threshold is determined by attempting to maintain a predefined consistency level of the data. By loosening this consistency level, Chi2 can come up with features that have only one interval, which can be removed for feature selection. Results on two synthetic datasets showed that Chi2 effectively discretised relevant features and removed all noisy features. However, it is hard to predefine the inconsistency rate since it may cause inaccuracy to the discretisation process [226]. Modified Chi2 [226] is a completely automatic discretisation method that addresses the drawbacks of Chi2.

Another approach to feature selection via discretisation is applying feature ranking based on some measures calculated during the discretisation process. Then, a number of top-ranked features will be selected. An example of this approach is PEAR [105], which ranks features based on the number of cut-points. A smaller number of cut-points indicates a better feature. The top-ranked features are selected to form the final subset. Results showed that it has similar performance as the original feature set and better than Relief. However, it is difficult to choose appropriate parameters used in the discretisation process. Similarly, in [71], features are discretised using Lloyd_Max algorithm [147]. The ratio of the original feature variance to the number of bits used to encode the discrete feature is used to rank features. Results showed that the proposed method obtains a similar or better classification performance as other discretisation methods. However, the number of features that should be selected from the ranked list is still an issue, especially for high-dimensional problems. In contrast, feature selection using EC techniques such as PSO and GA can automatically determine the number of features to select. The next section reviews different approaches to using PSO and other EC techniques for feature selection.

2.4.2 Feature Selection Using PSO

Generally, when a continuous PSO algorithm is applied to feature selection problems, a particle in the swarm is formed by a vector of N real numbers, where N is the total number of available features. In order to determine whether a feature will be selected or not, a threshold is needed to determine the selection of a feature. In binary PSO (BPSO), the representation of a particle is an N-bit binary string where "1" indicates that the corresponding feature is selected and "0" means not. Both PSO and BPSO have been proposed as wrapper, filter or hybrid (i.e. combination of wrapper and filter) feature selection methods. As a general search technique, PSO cannot be easily used for embedded feature selection. The following subsections will review wrapper, filter and hybrid approaches to PSO for feature selection.

2.4.2.1 PSO Based Wrapper Feature Selection

Many PSO based wrapper feature selection methods have been proposed in the literature using different learning algorithms such as SVM [13, 19], KNN [50, 51, 259, 258, 266] and AdaBoost [163]. For example, Azevedo et al. [19] proposed a wrapper feature selection algorithm using PSO+SVM for personal identification in a keystroke dynamic system. Experiments showed that the proposed approach produced better performance than a GA+SVM regarding the classification error, processing time and the feature reduction rate.

As a relatively new EC technique, PSO cannot avoid having some disadvantages. Its high possibility to get stuck in local optima can be a typical example. One strategy to help PSO jump out of local optima is to reset *gbest* or particles when the *gbest* fitness is identical after a predefined number of iterations [50, 51, 258]. A new *gbest* is created by using a Boolean operator to 'AND' each bit of the *pbest* of all particles [258]. The proposed method has shown to achieve higher classification accuracy with fewer features than GA and BPSO. Similarly, Chuang et al. [50] reset *gbest* to zero which is equivalent to an empty subset. Experiments on gene expression datasets

showed that this method effectively reduced the number of features and acchieved a higher classification accuracy than the method proposed in [258] in most cases. In [51], when PSO got stuck, 10% of the worst particles were forced to extreme positions (either all 0s or all 1s randomly). The reported results were better than deterministic algorithms such as SFS, PTA, and SFFS, and other stochastic algorithms including standard BPSO, simple GA and hybrid GAs on all datasets.

Enhancing the particle velocity and position updating mechanism is another way to improve PSO's search performance. Yang et al. [259] proposed to dynamically adjust inertia weight w using a logistic map or a tent map. Experiments showed that the proposed methods produced slightly higher classification accuracy than other methods, including SFS, SFFS, plus and take away, a sequential genetic algorithm (SGA) and different hybrid genetic algorithms (HGAs). Fitness was also used to dynamically set the value of inertia weight in [29]. In [161], "speed" was used to replace velocity in updating particles' positions to increase the probability of not choosing a feature. In this way, PSO was able to find much smaller feature subsets than [50] and other compared methods. However, classification performance was degraded in cases where a higher number of features might be needed to obtain a good prediction. In a BPSO wrapper feature selection method using SVM (BPSO+SVM) [13], particles were moved by applying a three-parent mask-based crossover operator involving the current position, *pbest* and *gbest*. Compared with GA+SVM, BPSO+SVM performed slightly better with smaller feature subsets. In [231], a new position updating mechanism was proposed. Features were selected according to not only their independent likelihood calculated by BPSO, but also their contribution to the subset of features already selected. Experimental results indicated that the proposed method outperformed tabu search and scatter search algorithms.

Reducing the PSO search space by eliminating redundant features was also proposed to improve PSO performance in feature selection. Lane et al. [128, 129] proposed a new mechanism to update position in BPSO. A

statistical clustering method was first used to group similar features into the same cluster. Then, during the evolutionary process, only one [128] or some [129] features with the highest probability (i.e. velocity) in each cluster were selected. Nguyen et al. [178, 180] proposed a new representation where the dimensionality of each particle was set to the number of clusters produced by statistical clustering. Each position represents the probability to choose the feature in the corresponding cluster. In [179], local search is applied to *gbest* to limit the number of features selected from each cluster. Results of these methods showed that the proposed methods can select a smaller number of features to achieve similar or better classification performance than all features and the compared methods. However, it is not easy to choose an appropriate number of features that should be selected from each cluster.

Fitness function plays an important role in guiding particles to find better feature subsets. Therefore, aggregating fitness function was proposed to simultaneously maximise the classification accuracy and minimise the number of features using different weights to balance between the two objectives [13, 250]. However, determining an optimal weight in advance is not a trivial task. Therefore, a multi-objective approach using PSO was proposed for feature selection [252] to solve this problem. In [252], two methods using the non-dominated sorting concept (NSPSOFS) and the crowding, mutation and dominance concept (CMDPSOFS) were proposed to evolve non-dominated solutions for feature selection in classification. The results showed that both algorithms achieved more and better solutions than LFS, GSBS, the standard PSO, [250], and the three well-known evolutionary multi-objective algorithms, NSGAII, SPEA2, and PAES on 12 benchmark datasets. Furthermore, comparison between continuous PSO and binary PSO for multi-objective feature selection conducted in [253] showed that the continuous representation generally achieved better performance than its binary counterpart.

A good initialisation procedure is also critical in improving PSO performance [13]. Xue et al. [251, 255] proposed three mechanisms to initialise

particles in PSO for feature selection including small (roughly 10% of features selected), large (more than 50% of features selected), and a mix of the two. Among the three methods, the mixed initialisation gave the best results with much smaller subsets and better or at least similar accuracy as the standard PSO and the aggregative fitness approach [250]. An opposition-based initialisation method was proposed in [29] to reduce the distance between the optimal solution and the initial population. Each random initial particle will be compared with its opposite particle. Only the winner will belong to the initial population.

Since PSO encodes a feature subset in a vector representation, which is similar to a chromosome in GA, different strategies to improve PSO performance have been inspired from GA. For example, in [177], a fitness-based roulette wheel was used to choose pairs of particles to perform a uniform crossover operator as in GA. The better offspring will replace its parent. Mutation is also proposed to mutate *gbest* when it does not improve for a number of iterations. The results on 8 datasets with 13 to 649 features showed that crossover and mutation operators help PSO achieve better performance than the compared PSO variants. However, a uniform crossover may not be effective for high-dimensional data. A new crossover operator was proposed in [47] using a predefined crossover weight to calculate position and velocity of the descendent particles based on their parents' positions and velocities. In [159], a new strategy was proposed to update *gbest* using replaceable and non-replaceable memories to maintain PSO diversity.

Enabling particles to learn from randomly selected competitors instead of *gbest* and *pbest* is another strategy to maintain the diversity of the swarm in a large search space. In competitive swarm optimisation (CSO) [48], the whole swarm is divided into pairs of particles in which the loser will learn from the winner particle. To reduce the high-computational cost of a CSO-based wrapper approach to FS, in [86], the fitness of all evaluated solutions were archived to be looked up later. The proposed method has shown to be efficient and effective. However, the lookup table may not maintain its efficiency when

each solution comprises tens of thousands of features.

Some wrapper methods not only used PSO for feature selection but also employed PSO to optimize parameters for the classification algorithm used in evaluation function. Lin et al. [138] proposed a wrapper feature selection approach (PSO+SVM), which simultaneously determined the parameters and picked a subset of features using continuous PSO. Experiments demonstrated that the classification accuracy of PSO+SVM outperformed that of a grid search, a newton SVM and a lagrangian SVM. A similar result was seen in [102] where the same approach was proposed for BPSO+SVM. PSO was also used in [163] to simultaneously pick the best feature subset and determine the decision thresholds for the AdaBoost classifier. Experimental results showed that the proposed method could be trained in a much shorter time and improve the performance of feature selection.

In general, most of the PSO based wrapper methods focussed on improving PSO search mechanism which is quite simple and flexible. Therefore, a large number of new strategies have been integrated into PSO based wrapper methods, resulting in a tremendous growth of this approach in the literature [256].

2.4.2.2 PSO Based Filter Feature Selection

PSO based filter approaches also gained attention from researchers. However, since these methods focussed on applying different measures to evaluate feature subsets, the number of filter methods is significantly smaller than wrapper methods. Similar to conventional methods, PSO based filter methods also used different measures to evaluate features, e.g. rough set [104, 238], fuzzy set [42], fuzzy consistency [43], mutual information, and entropy [39, 249].

Wang et al. [238] proposed a BPSO based filter approach to feature selection using rough set theory. The fitness function combined the degree of classes dependent on features (calculated using rough sets theory) and the proportion of the selected features. Results showed that the improved BPSO was computationally less expensive than a GA using rough sets in terms of

both memory and running time. However, the classification performance of the feature subsets was only tested on the LEM2, which is a rule induction algorithm that works based on rough set.

Fuzzy sets were utilised in [42] to build a BPSO based filter algorithm. Feature evaluation index [185] was used in the fitness function to minimise intraclass ambiguity and maximise interclass ambiguity. Experiments illustrated that the proposed BPSO performed better than GA did. The same author also proposed fuzzy consistency [43] as evaluation function that can also be used for continuous data. The proposed method was shown to obtain a smaller subset in a shorter time than a rough set approach. However, it is important to set appropriate thresholds for membership functions, which is not a trivial task.

Using information theory in fitness functions, Cervante et al. [39] developed two BPSO filter methods. Both algorithms used an aggregate fitness function combining the relevance level and the redundancy level with different weights. The relevance level and the redundancy level are measured by using mutual information in the first method and entropy in the second method. The results showed that the first method evolved smaller subsets while the second one produced better classification accuracy. However, the classification performance using the feature subsets evolved by both algorithms was just as good as using all the features in three out of four datasets. This confirms one of the drawbacks of filter approaches. These measures were further explored in [249] in a multi-objective approach and showed their potentials in improving the performance of DT classifiers.

A filter approach to feature selection using PSO was introduced by Guan et al. [87] for microarray data. Two informativeness metrics constructed based on ANOVA statistics were used to evaluate feature subsets. The results on two binary-class datasets were compared with six other methods. Although the proposed method always evolved the smallest subsets, it only achieved the best accuracy on one dataset.

2.4.2.3 PSO Based Hybrid Feature Selection

Recently, hybrid approaches combining filter and wrapper methods in two stages have been used for high-dimensional datasets as a way to address the limitations of both models. In [270], F-statistic was first used to rank features. The SVM accuracy from five-fold cross-validation was used to choose from a predefined set the best number of features that should enter to the second stage, where two wrapper methods were applied using either SVM or fuzzy KNN. One is the maximum relevance BPSO (MRBPSO) for class independent classification and the other one is class dependent multi-class classification (CDMC). Experiments on 8 gene expression datasets showed that both can improve the classification performance of SVM and fuzzy KNN with small numbers of features. However, the computational time was very high, especially for the class dependent method. Similarly, a quartile based preprocessing was applied in [25] to eliminate irrelevant features in the first stage. Remaining features are further selected by a binary PSO algorithm, which uses hamming distance as a proximity measure in the velocity updating formula. Results on three binary gene expression datasets showed that the proposed method outperformed the compared ones. However, its application to larger datasets with a higher number of classes may be hindered by its memory requirement.

Feature clustering was also proposed in the first stage to reduce feature redundancy before applying PSO. In [204], k-means was used to group features into k clusters. Then, signal-to-noise ratio score was used to select the best feature from each cluster to transfer into the second stage for PSO to search for the optimal subset. The results showed that with a much smaller size, the selected feature subset helped SVM, KNN and Probabilistic Neural Network achieved higher accuracy than using all features. However, the different setting values of k in k-means for different datasets might require some expert knowledge about microarray data.

Another hybrid approach using rough set theory was introduced in [1] to select reducts (i.e. feature subsets) in the first stage to be entered into PSO

for the second stage, where PSO used SVM to evaluate the particles. Feature subsets evolved by the proposed algorithm achieved higher accuracy and had smaller feature subsets than reducts chosen by the rough set approach only.

Recently, Jain et al. [106] used the feature subset returned by CFS [94] as an input of the second stage where BPSO combined with NB to further reduce the number of features. During the evolutionary process, if *gbest* does not change after a fixed number of iterations, it is reset based on the majority voting of all *pbest* for each dimension. Experiments on 11 gene expression data have shown the superior performance of the proposed method when compared with seven traditional and EC-based methods. However, feature selection bias exists since the whole dataset is used to evaluate features in the fitness function of BPSO 2.2.3.1. This problem also occurs in another recent PSO-based FS method [191] where SVM was applied on the whole dataset to rank features according to the returned absolute weight vector. The top 50 features were put into a recursive PSO scheme to further select a smaller feature subset. In each iteration, features that were not selected in *gbest* would be removed and a new PSO was called on the reduced feature set. This recursive process was continued until no more features was removed or the accuracy of *gbest* was degraded.

Instead of conducting filter and wrapper methods in two separate stages to reduce the PSO search space as in the above methods, some methods integrated filter and wrapper measures in a single stage to directly enhance the performance of PSO. For example, in a wrapper method using PSO with ID3 [146], mutual information was used to score the relevance and redundancy of a feature within a feature subset. This score was then added to the velocity of the particle to give features with a higher score a higher chance to be selected.

In a PSO based wrapper method [179], a local search using mutual information was applied to *gbest* to remove redundant features. In contrast, a PSO based filter method [35] used classification accuracy to decide if a new *pbest* should be updated or not. Results showed that the proposed

method achieved higher classification accuracy than a pure filter method. However, results from different combinations of filter and wrapper evaluations also showed that the more filter evaluations it used, the worse classification performance and the larger feature subsets it obtained [35]. Therefore, how to combine filter and wrapper approaches to synthesize their strengths while limiting their drawbacks is still challenging.

2.4.3 Feature Selection Using Other EC Methods

Not only PSO and GP, many other EC techniques have been applied to dimensionality reduction problems such as genetic algorithms (GAs), and ant colony optimisation (ACO). The rest of this section reviews some typical methods of these two techniques that have been used in feature selection.

2.4.3.1 GA-Based Feature Selection

Among EC techniques, genetic algorithm (GA) is probably the first popular EC technique that has been applied to feature selection. A large number of GA-based feature selection methods have been proposed in the last ten years [256]. In GA feature selection methods, candidate feature subsets are represented as binary strings in which "1" or "0" represents the selected or not selected feature, respectively.

Filter methods were also proposed using different measures such as consistency rate [133], fuzzy set [41], distance measure [135], ranking evaluation function [54], and rough set theory [24]. In [41], Chakraborty used fuzzy set to evaluate feature subsets. The feature subset having a minimum intraclass ambiguity and a maximum interclass ambiguity is considered as the best feature subset. Experiments on 2 datasets with 4 and 60 features showed that the proposed method was robust. However, the performance of the proposed method was inferior to the BPSO based feature selection using the same fitness function [42] proposed by the same authors in terms of classification accuracy, feature subset size and computational time. Based on

rough set theory, Banerjee et al [24] proposed a filter-based feature selection for gene expression data. To reduce the number of redundant features, several preprocessing steps were performed to enable faster convergence and reduce the computational complexity. A rough set theory was used to generate reducts that represented the minimal sets of non-redundant features capable of discerning between all objects. Evolved feature subsets of three datasets were very small and significantly improved the classification performance of KNN classifier.

Many GA-based wrapper feature selection methods were also proposed using different learning algorithms such as KNN [236], NB [54], SVMs [83, 103, 223, 271], neural networks [84, 97, 113, 182], and DT [239]. Since the learning algorithm is considered as a black-box in wrapper approaches, enhancing GA performance usually focused on improving its search mechanism, proposing new representation, or using aggregative functions.

Local search is a popular approach to balancing the global and local search in GAs [103, 113, 271]. In [103], Huang et al. proposed a local search operation called stepwise elimination in which features are ranked using an improved formula to compute the conditional mutual information between the candidate feature and the classes. The proposed fitness function used not only the SVM error rate but also the mutual information return from the local search. Experiments showed that the two approaches were well incorporated to achieve a competitive accuracy and a smaller number of features in many datasets. However, the proposed method is quite complex with a very high computational time. Zhu et al. [271] combined a GA with a local search in a Markov blanket-embedded GA for gene selection. In each generation, the best individual was input to a local search process in which a symmetrical uncertainty measure was used to calculate the correlation between selected features. In the local search, operator Add or Del was randomly applied to add a relevant feature or to remove redundant features, respectively. C4.5 and SVM were used in fitness function for synthetic datasets and microarray datasets, respectively. The chromosomes were constrained to 50 for binary-

class problems and 150 for multi-class problems. Results showed that the proposed method outperformed the compared methods in a reasonable time.

GA-based feature selection methods using a multi-objective approach to minimise both the error rate and the number of features is also popular in GAs [256]. For instance, Oliveira et al. [182] proposed a GA-based wrapper feature selection method for handwritten digit recognition. In [239], a multi-objective approach was proposed based on different classes since a subset that was irrelevant to one class might be relevant to another one.

To improve the performance of GAs on high-dimensional data, Hong et al. [97] proposed a new representation of chromosomes that are binary strings presenting the indices of the features. The chromosome length is fixed to be a predefined number of features that should be selected. Furthermore, to maintain the diversity of the population, this method used an explicit fitness sharing which degraded the fitness of individuals in densely populated regions. A neural network was used in the fitness function. Experiments on three gene expression data showed that the proposed method found better solutions in a shorter time than the standard GA. However, the computational cost was still high and some domain knowledge was required to set an appropriate number of features that should be selected for each dataset.

Combining different evolutionary algorithms was also proposed to obtain better feature subsets. A multi-stage feature selection method was proposed in [84] called SAGA which performed simulated annealing (SA), then GA and then 1-bit mutation local search to find the best subset. Generalized regression neural network is used as a classifier to evaluate an individual subset which is 100-bit genome. Experiments on synthetic, benchmark datasets with ten thousand features and a real-world dataset with 285 features. Results showed that SAGA achieved better performance in terms of classification accuracy and feature subset size than the other compared methods. However, the running time was still very high. Similarly, based on the fact that GA and PSO have the same representation in feature selection, a hybrid method was proposed in [83] where genetic operators and particle updating mechanism

are combined.

In [46] GA was proposed for feature selection via feature clustering. GA was used to evolve the cluster centre for a predefined number of feature groups. Then, features in the same cluster were ranked based on its distance to the centre. The top-ranked features in each cluster were selected to create the final subset.

Utilising traditional feature selection methods to narrow the search space is also an effective approach. Tan et al. [223] proposed a framework where features are ranked by different feature selection methods. Some top-ranked features from each method are merged into a pool where a GA-based wrapper method using SVM was applied to choose the final feature subset.

2.4.3.2 ACO-Based Feature Selection

When using ant colony optimisation (ACO) for feature selection, each feature is considered as a node in a fully connected graph, and paths between nodes represent the choices for next features. Therefore, the search for the optimal subset is an ant traversal through the graph to find a minimal path that satisfies the traversal stopping criterion.

In [114], a wrapper feature selection method based on ACO was proposed for a face recognition system. In this method, KNN classification performance and the number of selected features are used as heuristic information for ACO. Results showed that the proposed method performed consistently superior to the GA-based and other ACO-based feature selection methods. However, the computational time is still very high. The same approach was introduced in [3] for text categorisation. Experiments on the Reuters-21578 dataset showed that the proposed method was superior to GA and other statistical methods including information gain and Chi-square.

Jensen et al. [110] proposed an ACO filter method using a fuzzy-rough set dependency measure as the ant traversal stopping criterion. The proposed method was compared with other five benchmark techniques. Results showed that the proposed method and a simulated annealing based method outper-

formed the other three techniques. However, no comparison with other EC methods such as PSO or GAs was conducted. In [157], an ACO-based feature selection method using rough set theory was applied to find the core feature set which was the smallest set of features that could not be removed. Then, each ant performed forward selection to expand this core set until the original positive region was reached. Experiments showed that the proposed method is promising.

In general, ACO has shown promise in feature selection. However, with a graph based representation, ACO does not scale well when the number of features or nodes in the graph increases to thousands or tens of thousands.

2.4.3.3 GP Based Feature Selection

Although GP is well-known in feature construction, some studies [6, 130, 172, 173] have proposed GP for feature selection thanks to its intrinsic characteristic of choosing features to construct new features. In [172], a binary relevance measure was proposed to evaluate the single-tree GP individuals. Features appearing on a GP tree that had its fitness better than a predefined threshold were used to form a feature subset. For each subset size, the best subset was archived and accumulated over 50 independent runs. Then, the feature subset having the highest classification accuracy will be returned. Experiments on three datasets with tens of features showed that using the selected features improved the performance of C4.5, NB, SVM and Bayesian networks. However, the proposed method required a high computational time. In [173], Neshatian et al. proposed to use GP for context-sensitive feature scoring in classification problems. GP was used to build weak classifiers to discriminate instances from one class to other classes. Each feature was scored by its contribution to the fitness of the weak classifiers where it appeared. The true negative ratio was used as the criteria to evaluate the weak classifiers. 300 independent runs were conducted to collect 600 weak classifiers for each binary-class problem. The effectiveness of the proposed method on datasets with tens of features has shown by improvement of the four learning algorithms as in [172] using

different numbers of high-ranked features. However, it may not scale with high-dimensional data.

One of the first works proposing GP embedded approach to feature selection on high-dimensional data was from Langdon and Buxton [130]. This is a multiple-stage method. In each stage, 600 GP runs were used to collect the most-selected features to enter the next stage. The last stage ran GP as a classification algorithm. Although the obtained result on a cancer dataset was not better than the compared method, insights of applying GP to this dataset were presented.

A similar approach to feature selection was proposed for mass spectrometry data by Ahmed et al. in [6]. Features selected in all the 300 best individuals of 300 runs on each dataset was collected to feed into SVM and NB to compare the performance of selected features with original feature sets on 8 datasets with increasing within class variability. The results showed that GP selected features helped SVM and NB achieve better performance than using all features. However, the number of selected features was still quite large. Therefore, signal to noise ratio was used to rank these selected features to further decrease the dimensionality. Another work by the same authors [8] used GP to further improve the feature ranking performance of information gain and Relief-F methods. One hundred top-ranked features from each method are collected to put into GP. The frequency of their usage in the best GP individuals of 300 runs was used as the criteria to rank features. The results showed that the 20 top-ranked features by GP achieved better performance than all the original features, the top 100 or 20 features ranked by information gain and Relief-F individually. However, the computation time is quite high to collect results from all the runs for a final feature subset.

2.4.4 Feature Construction Using Conventional Techniques

Embedded methods were one of the early approaches to feature construction. Murthy [167] proposed an oblique decision tree (DT) induction system called OC1 which created a new feature for each node of the DT using linear combinations of original features. To adjust the coefficients of the linear combinations, OC1 used deterministic hill-climbing until reaching a local optimum, then applied random strategies to optimise the oblique split at each DT node starting from the best axis parallel split. Experiments on six datasets with about ten features showed that OC1 trees achieved higher accuracy than CART-LC (another oblique DT) and C4.5 with equal or smaller tree sizes. Similarly, XofN [269] was also based on DT. At each decision node, it constructed a new nominal feature in the form of $X-of-N$ representation which contained feature-value pairs. Experiments showed that XofN achieved higher accuracies and simpler DTs than the compared DT methods.

Approaching to feature construction as a separate and independent preprocessing stage, Hu et al. [101] proposed a filter method called GALA using information gain as a measure to evaluate the constructed feature. It used two logical operators, AND and OR, to construct new features from boolean ones. Results on 12 UCI datasets showed that GALA significantly improved the inductive learners on most datasets. However, GALA can generate boolean features only. Some feature construction systems such as BACON [132] and STAGGER [206] can construct numeric features by using mathematical operators such as multiplication and division. However, applications of these methods to real-world problems are rather scant due to their relatively high computational complexity [126].

Feature clustering has also long been used to construct features for text classification problems [22, 62, 212]. "Similar" features were grouped into the same cluster that its centroid was considered as a new feature that represents all the features in the group. These methods have shown promise in

dimensionality reduction for text classification problems. However, predefining an appropriate number of clusters may require some domain knowledge.

As a statistical approach, principal component analysis (PCA) [98] has been widely used as a dimensionality reduction technique. It constructs new features, or components, by linearly transforming the original features to a new uncorrelated space. PCA ranks new features based on their variances presented as eigenvalues. Top-ranked features are selected to form new features. However, as PCA is unsupervised learning, its constructed features may not well separate instances from different classes.

Linear Discriminant Analysis (LDA) [74] is a supervised version of PCA, which aims to find the projection hyperplane that minimises the interclass variance and maximises the distance between projected means of classes. Therefore, its constructed features may have better discriminating ability than PCA's components but not always [154]. However, the number of features LDA can construct is limited by the number of classes. Furthermore, both methods are limited to linear transformation of the original features.

Non-linear PCA (NLPCA) [208] was also proposed using auto-associate neural network (also known as autoencoder or replicator network). It is actually a multi-layer perception that performs an identity mapping from input to the same output values. The constructed features are taken from the middle layer that has a smaller number of nodes as a form of dimensionality reduction. However, parameter estimation must be done carefully to ensure robust approximation. Recently, deep learning [26], e.g. deep neural networks and deep belief network, has obtained very good results in feature learning. Examples can be seen in speech recognition, natural language processing, and computer vision [61]. However, large amounts of data and computational resources are crucial to their performance. Furthermore, solutions of both NLPCA and deep learning approaches are usually lack of intelligibility or human readability. In contrast, GP can automatically evolve new features by combining some of the original ones without a predefined template such as linear or non-linear. The tree representation of GP constructed features also

shows the importance of the original features as well as relationships between them [214]. The next section will introduce how GP has been used in feature construction.

2.4.5 Feature Construction Using GP

When using GP for feature construction, the terminal set comprises of random constants and features chosen from the original features. The function set comprises of any predefined mathematical operators that can be used to combine the selected features. A GP individual represents a new feature (single-tree GP) or a set of new features (multi-tree GP), resulting in single or multiple feature construction methods, respectively. GP based feature construction has been proposed as either filter or wrapper methods. To the best of our knowledge, embedded approach has not been proposed in GP based feature construction methods. Both wrapper and filter approaches will be introduced in the following subsections.

Note that GP is not the only EC technique used for feature construction, GA [15, 209] and PSO [55, 257] have also been proposed. In [209], two different GAs were used to separately select features and binary operators. The operator-selected GA was nested in the fitness function of the feature-selected GA to find an appropriate set of operators for each candidate set of features. It has shown to have comparable results as the compared feature construction methods. However, applying it to real-world problems may be impractical due to its complexity. To construct feature using the selected features by PSO, Xue et al. [257] used a local search to find the best operator for each candidate subset. The method was shown to successfully construct a new feature that generally improved the performance of KNN, DT and NB. However, operators were not evolved by PSO and the computational time was high. To address these problems, two new encoding schemes for PSO [55] were proposed to automatically choose operators. The method ran 100 times faster while achieved similar performance as [257]. However, their representations require more memory space than standard PSO, which can

be an issue for high-dimensional data.

2.4.5.1 GP Based Wrapper Methods for Feature Construction

One of the first GP based wrapper methods was proposed by Raymer et al. [198]. It aimed to improve their previous method that used a GA to evolve weights that can transform each original features into a new one. The GA was replaced by a GP to enable non-linear transformation for each feature. Each GP individual is a single-tree with n subtrees, where n is the number of original features. KNN was used to evaluate the constructed feature set. Results on a water displacement problem with four features showed that the proposed GP based feature construction method obtained better KNN accuracy than a GA-based one.

Cooperative coevolution was also proposed to combine many single-tree GP populations to construct multiple features. In [4], one GP population was used to evolve a new feature for each original feature. Another GA population was used for feature selection. KNN was used to evaluate the new feature set. Similarly, [27] used n concurrent populations of single-tree individuals to evolve n new features where n is the desired number of constructed features. The entire new feature subset was evaluated using DT. Experiments on a dataset with nine features showed that the proposed method constructed better features than the standard multi-tree GP method. However, since the number of fitness function calls was n time greater than that of GP, the computational time of cooperative coevolution approach was significantly longer.

Using multi-tree GP, Krawiec [125] proposed a wrapper method to construct a predefined number of features (n). Each GP individual comprised of n new features and n hidden features. Hidden features were also constructed features but kept out of the evolutionary process to avoid losing good constructed features during the evolutionary process. DT was used to evaluate each individual. Results on six datasets with less than ten features showed that constructed features improved the classification performance of DT on

most of them. Using hidden features also obtained close to or sometimes better results than standard approach where all features are involved in the evolutionary process. Smith and Bull [215, 214] also proposed a wrapper method for feature construction and selection simultaneously which used DT to evaluate features. For an n-dimension problem, a GP individual has n trees, each associated with a boolean variable showing if it is selected or not, and a flag to choose among DT, KNN and NB for evaluation. The method has shown to significantly improve classification performance on 2 out of 10 datasets with 5 to 60 features.

GP and GA were also combined in two stages of a single method for feature construction and selection, respectively. In [234], GA was used to select a subset of features which was then used by GP to construct new features. The reverse order where GP was used to construct features which were then selected by GA was also investigated in the proposed method [213]. A multi-tree GP was used to construct n new features, where n is the number of the original features. Experiments on 10 UCI datasets showed that GAP improved the performance of C4.5 on 8 datasets. Although C4.5 was used in its fitness function, GAP results were robust to other classifiers including KNN and NB. Another method which conducted both feature construction and selection simultaneously but in a single evolutionary process was proposed in [176]. Combined GA and GP in a single representation, each individual contains an $n - bit$ string for feature selection and one tree to construct one feature. The whole set of selected and constructed features were evaluated by KNN. Results on 10 UCI datasets showed that the proposed method either obtained similar or significantly better accuracies than the compared methods in almost all cases. The results from these work have demonstrated that combination of EC techniques is a promising approach to feature manipulation. However, on top of the high computational cost, these representations are not suitable for problems with thousands of features.

In [9], two GP based wrapper feature construction methods, namely GPWFC1 and GPWFC2 were proposed for high dimensional data with a

novel approach to constructing multiple features using single-tree GP. While GPWFC1 used classification accuracy of random forest (RF) classifier as the fitness value, GPWFC2 used entropy gain of RF and the p-value of an ANOVA test on the selected features. Results showed that GPWFC2 achieved better generalisation ability and smaller numbers of features than GPWFC1. However, the computational cost was quite high when adopting RF as the learning algorithm for fitness evaluation. This is a common drawback of GP based wrapper approaches, which can be avoided with filter methods.

2.4.5.2 GP Based Filter Methods for Feature Construction

Using information gain ratio to evaluate the constructed feature, Otero et al. [184] proposed a GP based filter feature construction method to construct a new continuous or boolean feature using four arithmetic operators (+,–,* and protected division %) and two relation operators ($\leq, \geq$). The constructed feature was then augmented to the original feature set. Results on four datasets with 4 to 21 features showed that the constructed feature can improve the performance of C4.5. However, since both C4.5 and the proposed method are based on information gain, it is unknown whether the constructed features will also be beneficial for other learning algorithms. This question has been investigated in [165, 166] where the performance of C5, which is a DT algorithm, was compared with two other DT algorithms, CART and CHAID, and multilayer perceptron. Results showed that these classifiers generally benefited from the inclusion of the constructed feature. Furthermore, the results also showed that using the evolved feature, the learnt classifiers outperformed the embedded approach which uses GP to induce a classifier.

Running single-tree GP program multiple times, Neshatian et al. [175] proposed a new approach to multiple feature construction. The number of constructed features was equal to the number of classes. Each GP run focused on one class. Constructed features were evaluated based on the impurity (using Shannon entropy) of the intervals which were formed by applying class dispersion to the transformed datasets. The results indicated that constructed

features helped DT achieved smaller error rates with much smaller sizes than using original features in most cases. However, when combining constructed features with original features, the performance of learnt DTs was worse than using constructed features only in most cases. The method was improved in [171, 174] by adding class-wise orthogonal transformed features to GP terminal sets. Results indicated that constructed features increased the classification accuracy of the learnt DTs. However, by adding more features, this strategy significantly increased the GP search space. Therefore, it may not be effective and efficient for high-dimensional data.

In general, GP based filter feature construction methods are usually limited in developing one feature at a time using single-tree GP. This may be because they usually use univariate measures, e.g. information gain, which can only evaluate one feature at a time.

2.5 Summary

This chapter reviewed the essential background of ML, classification, feature selection, feature construction, and evolutionary computation techniques, particularly PSO and GP. The related work of using conventional methods and current evolutionary computation approaches to feature selection and construction in this chapter showed that feature selection and construction on high-dimensional data still faces many challenges as mentioned in Section 1.2.

Most of the conventional feature selection methods utilised deterministic search such as sequential search which is confronted with the problem of stagnation in local optima, especially when the number of original features is large. Although population search in PSO can alleviate this problem, it is still challenging for PSO to explore the huge search space of high-dimensional data to find the optimal or near-optimal feature subset. Further improvements are needed to enhance PSO performance on these datasets.

For feature construction, the representation of GP provides a natural

and effective way to construct new features from the original ones. This has been shown from the results of the existing GP based feature construction methods. However, most of them were applied to datasets with about tens of features. Therefore, their capabilities on high-dimensional data should be further investigated.

Specifically, motivations of this thesis can be summarised as follows.

- The existing works have shown that the PSO global search has obtained better results than many conventional methods and GAs. However, they still face the problem of stagnation in local optima when applied to high-dimensional data. A good balance between global and local search in PSO can enhance the performance. Local search has been combined with PSO to overcome this drawback [188]. However, this strategy has not been investigated much in PSO based feature selection, especially for wrapper feature selection on high-dimensional data due to its high computational cost.

- Feature selection via discretisation has shown promise in the feature selection literature. However, using the uni-variate discretisation method, the existing feature selection via discretisation methods cannot deal with feature interactions. On the other hand, PSO has long been used for feature selection but never been applied to discretisation. Thanks to its real-number vector representation and the global search ability, PSO has a high potential in multi-variate discretisation. However, this approach has not been investigated.

- Existing GP based methods have shown that GP has a built-in capability to select appropriate features based on the guidance of the fitness function. However, its performance is still degraded when confronted with a huge search space. Therefore, effectively narrowing this search space can enhance GP performance for feature construction on high-dimensional data. This approach has not been investigated in GP for feature construction.

- Most of the existing GP based methods focused on constructing class-independent features based on "relevant" features selected from the whole pool of the original ones. However, not all relevant features can distinguish instances between all classes. Some particular features may have better ability than the others in distinguishing instances of a particular class. Therefore, constructing class-dependent features from the corresponding class-relevant ones may obtain better-constructed features. However, the application of this strategy to feature construction is still limited.

By proposing new algorithms using PSO and GP for feature selection and feature construction, respectively, the above-mentioned issues will be addressed in the following four chapters. Chapter 3 will develop two new local search strategies for PSO based feature selection. Chapter 4 will introduce two new PSO representations for feature selection via discretisation. Chapter 5 will develop a new GP based feature construction method using feature clustering to narrow the GP search space. Different combinations of selected and constructed features by the proposed method are also investigated. Chapter 6 will develop a class-dependent feature construction method using GP with a new filter-based fitness function.

3

PSO and Local Search Based Feature Selection

3.1 Introduction

PSO has been applied and shown promise to feature selection [25, 249]. However, applying PSO to high-dimensional data still faces the problem of stagnation in local optima due to the large search space. PSO works based on the information sharing between particles in a swarm. By communicating the best solution that each particle has explored so far (the personal best, i.e. *pbest*), each particle is able to know the best position the whole population has found (the global best, i.e. *gbest*). Using this information, each particle is driven to somewhere in around the middle of its *pbest* and *gbest* to explore better solutions. On the other hand, this behaviour in some extent prevents particles from fully exploiting the area surrounding their findings, which is *pbest*, to obtain more refined and better solutions. Local search has been combined with PSO to overcome this drawback [188]. An

evolutionary computation algorithm when combined with local search is also called a memetic algorithm, which has been shown to be effective in many feature selection algorithms [271, 272]. However, this strategy has not been investigated in PSO based feature selection for high-dimensional data due to its high computational cost.

3.1.1 Chapter Goals

The goal of this chapter is to develop a new PSO and local search based feature selection algorithm for high-dimensional classification. The proposed method is expected to select only a small subset of features and achieve similar or significantly better classification performance than using all features and other compared methods. To achieve this goal, two new local search strategies are proposed to find better solutions surrounding the new *pbest* location. One uses random strategy and the other uses a correlation based measure, resulting in two methods called PSO-RLS and PSO-CLS, respectively. For presentation convenience, PSO-LS is used when both methods are mentioned. A fast fitness evaluation strategy is also proposed to speed up the local search process. Specifically, this chapter will investigate:

- Whether the feature subsets selected by PSO-LS are smaller and achieve similar or better classification performance than the original feature sets;
- Whether PSO-LS methods can achieve better performance than other PSO based feature selection methods (without local search);
- Which of the two proposed local search strategies provides better results in terms of feature subset size and classification accuracy;
- Whether PSO-LS methods obtain better performance than the traditional methods; and
- Whether the fast fitness evaluation strategy used in the local search process helps the PSO-LS methods significantly reduce their running

time to achieve similar or better accuracy than other PSO based feature selection methods.

3.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. Section 3.2 presents the proposed PSO-LS algorithms. Section 3.3 discusses the datasets used to test the performance of the proposed methods, the parameter settings and the baseline methods for comparison. The results of the experiments are presented and discussed in Section 3.4. Section 3.5 further discusses computation time, robustness and learning capability of the proposed methods. Finally, Section 3.6 provides a summary of this chapter.

3.2 The Proposed Algorithms

The main idea of the proposed approach is to combine PSO with local search to intensify the search in the area surrounding the best position that each particle has visited, i.e. the *pbest*, with the expectation that it can explore better solutions. The overall structure of PSO-LS can be seen in Figure 3.1 in which the added step to standard PSO are highlighted: a "local search" on *pbest*. The following subsections will describe the two proposed methods using two local search procedures and the fast fitness evaluation strategy proposed to speed up the local search process.

3.2.1 PSO with Random Local Search: PSO-RLS

A local search mechanism is used as a complementary search of the PSO based global search, which aims to exploit the search space around the obtained good solutions, *pbest*, to find better ones. As can be seen in Figure 3.1, when the PSO algorithm finds a better *pbest* for a particle, e.g. *pbest* is updated, the local search is applied to refine the updated/new *pbest*. If a better solution

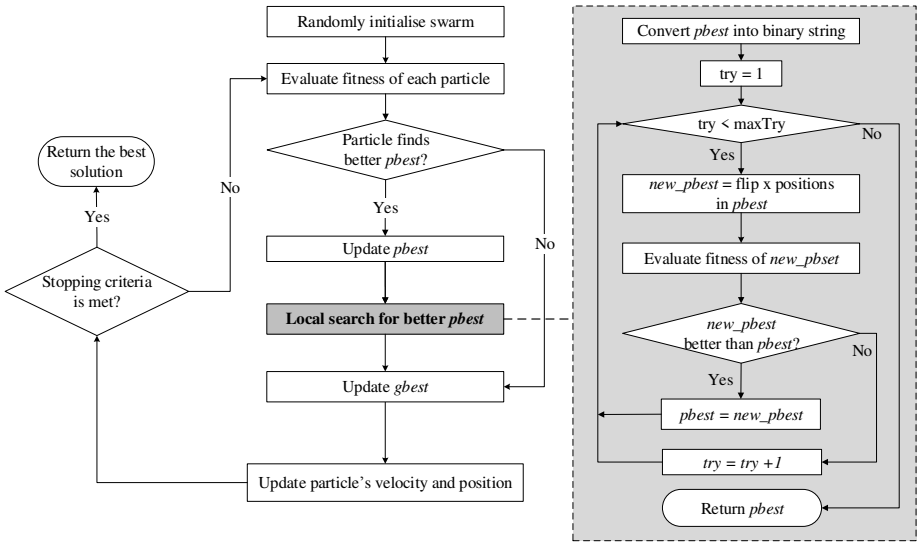


Figure 3.1: Flowchart of PSO-LS.

is found by the local search, *pbest* will be further updated as the new solution. Otherwise, *pbest* remains the same.

In PSO-LS, the position of each particle represents a solution, i.e. a feature subset. Although both continuous and binary PSO have been used for feature selection, binary PSO has potential limitations in its updating mechanism where the new position is updated based solely on the velocity while the position in the standard PSO is updated based on both the velocity and current position [68]. Therefore, the continuous PSO is used here. The representation of the particle's position is a real-number vector with the length equivalent to the total number of features in the dataset. Each element of the vector corresponds to one feature and the possible value is in the range of [0, 1], which represents the likelihood of being selected of the corresponding feature. A threshold θ is used to determine the selection of the feature. A feature is selected if its corresponding position value is larger than θ; otherwise, it is not selected. To perform local search, the *pbest*, which is a real-number vector, is converted into a binary vector to show a feature subset in which "1"s represent the corresponding feature being selected.

The aim of the local search is to explore the area surrounding the current *pbest* solutions. Therefore, it will generate and examine candidate solutions that are slightly different from the current *pbest*. To ensure that the local search only focuses on the near surroundings of the *pbest*, "similar" candidate solutions is generated by randomly flipping a small proportion of the binary vector. In other words, the local search is a flipping mechanism, which flips the values in the binary vector from "0" to "1" or from "1" to "0", i.e. a feature from being "not selected" to "selected" or from being "selected" to "not selected". Since this work targets on high-dimensional data with thousands or tens of thousands of features, only 2% of the bits/dimensions in this binary vector are flipped in each step of the local search. Algorithm 1 shows the pseudo-code of the random local search which outputs a better *pbest* based on a given *pbest*, a percentage (δ) of features to flip and a maximum number (β) of flipping tries. While δ shows how different the new *pbest* compared to the current *pbest*, β specifies how long the local search should run. Based on δ, m number of features will be randomly picked to flip. If the flipped *pbest* is better than the current *pbest*, *pbest* is updated. The flipping is repeated for β times.

3.2.1.1 Fitness Function

Although filter methods are said to be faster than wrappers, their performance is usually inferior to wrappers in terms of classification accuracy. Therefore, in this study, PSO-RLS is proposed as a wrapper method, where any learning algorithm can be used to evaluate the classification performance of the candidate solution. In this work, K-Nearest Neighbours (KNN) is used as it is simple, fast and non-parametric.

Since many datasets used in the experiment are unbalanced, a balanced classification accuracy [187] as shown in Equation (3.1) is used to guide the search.

$$balanced\_accuracy = \frac{1}{c}\sum_{i=1}^{c}\frac{TP_i}{|S_i|} \tag{3.1}$$

Algorithm 1: Random local search

Input : $pbest, \delta, \beta$
Output : *better pbest*

```
begin
    for t = 1 to β do
        m ← nbr_original_feature × δ;
        features_to_flip ← Randomly pick m features in pbest;
        pbest' ← Apply features_to_flip on pbest ;
        Evaluate pbest' using the fast fitness evaluation described in Section 3.2.3;
        if pbest' is better than pbest then
            pbest ← pbest' ;
        end
    end
    return pbest;
end
```

where c is the number of classes of the problem, TP_i is the number of correctly identified instances in class i and $|S_i|$ is the total number of instances in class i. Since there is no bias to any specific class, the weight here is set equally to $1/c$.

To evaluate a candidate solution, the fitness function transforms the training set based on the selected features. It then applies KNN with LOOCV to the transformed training set to calculate the average accuracy which will be used as the fitness of the corresponding candidate solution.

3.2.2 PSO with Correlation Based Local Search: PSO-CLS

Using a randomly flipping mechanism, the local search in PSO-RLS may have a low chance of reaching better solutions. In order to increase this chance, the local search process should be guided in order to keep relevant features and eliminate redundant ones. Therefore, a filter measure was proposed to incorporate general knowledge in feature selection to identify relevant and

redundant features. A feature is more relevant than others if it is more correlated to the class label than other features. Similarly, if features f_1 and f_2 highly correlate with each other, one of them can be said to be redundant. This means that f_1 can represent f_2 and vice versa; therefore, only one of them should be selected.

To evaluate the correlation between two variables X and Y, a normalised version of information gain (IG) [196] called symmetric uncertainty (SU) [192] is used. It is defined as follows:

$$SU(X,Y) = \left[\frac{IG(X|Y)}{H(X)+H(Y)}\right] \tag{3.2}$$

$$IG(X|Y) = H(X) - H(X|Y) \tag{3.3}$$

where $H(X)$ is the entropy of X and $H(X|Y)$ is the conditional entropy of X given Y. The value of $SU(X,Y)$ is in the range [0,1]. The higher the value of SU, the higher the dependence between X and Y.

Based on Equation (3.2), the relevance of a feature f is measured using the SU between f and the class label, which is denoted as SU_C. Similarly, SU_F is used to denote the redundancy between two features f_1 and f_2, which is measured based on the SU between f_1 and f_2.

The objective of this local search is to find a better *pbest* from each particle by removing from the current *pbest* a certain percentage of redundant features and introducing a certain percentage of relevant features. As shown in Algorithm 2, PSO-CLS has the same input and output specification as PSO-RLS. However, the number of features that will be considered to flip, m, is proportional to the current *pbest* size (i.e. the number of selected features in *pbest*) instead of the original number of features. This enables PSO-CLS to adjust its search according to the current *pbest*. Lines 4-13 remove redundant features by selecting $m/2$ features from current *pbest* to form the *ones* list, sorting *ones* based on the descending order of the relevance measure SU_C of each feature. Then, *ones* is scanned from the first feature (i.e. the most relevant one) to remove any subsequent features that are more correlated to the currently scanned feature than to the class label. In other words, any

Algorithm 2: Correlation based local search

```
   Input  : pbest, δ, β
   Output : better pbest
1  begin
2      for t = 1 to β do
3          m ← pbest_subset_size × δ;
4          ones ← Randomly pick m/2 selected features in pbest;
5          pbest′ ← pbest ;
6          Sort ones in descending order of their SU_C values calculated based on
             Equation (3.2);
7          for i = 1 to ones.length do
8              for j = i + 1 to ones.length do
9                  if (ones[i] and ones[j] are still in pbest′ and
                     SU_F(ones[i], ones[j]) > SU_C(ones[j]) ) then
10                     Remove feature ones[j] from pbest′;
11                 end
12             end
13         end
14         zeros ← Randomly pick m/2 non-selected features in pbest;
15         AvgSU ← Average SU_C of all features in ones;
16         for i = 1 to zeros.length do
17             if SU_C(zeros[i]) > AvgSU then
18                 Add feature zeros[i] to pbest′;
19             end
20         end
21         Evaluate pbest′ using the fast fitness evaluation described in Section 3.2.3;
22         if pbest′ is better than pbest then
23             pbest ← pbest′ ;
24         end
25     end
26     return pbest;
27 end
```

feature that has its SU_F with a previous feature selected in the *ones* list higher than its SU_C value is removed.

Lines 14-20 introduce more relevant features into the new *pbest* by ran-

domly choosing $m/2$ features that are not selected in *pbest* to form the *zeros* list. A feature in *zeros* will be added only if its SU_C is greater than the average SU_C values of the *ones* list to ensure that no feature that is less relevant than the current features on average is added. After being evaluated, if the new *pbest* is better than the current *pbest*, the current *pbest* will be updated.

3.2.2.1 Fitness Function

As with other EC methods, PSO search is guided by a fitness function. Therefore, the fitness score assigned to different candidate solutions should be as different as possible. However, when using solely classification accuracy as a fitness measure, this requirement may not be satisfied, especially in cases where the boundary margin between different classes is quite large, many different classifiers can achieve the same 100% classification accuracy. This scenario is likely to happen in high-dimensional data, where there may exist many redundant features, resulting in many different feature subsets which may obtain the same classification accuracy. Therefore, to improve the performance of PSO-CLS, an additional measure is added to better distinguish different candidate solutions without adding more computation effort.

A distance measure is a filter based multivariate measure that can evaluate the feature set as a whole. It can be used to maximise the distance between instances of different classes and minimise the distance between instances of the same class. Furthermore, KNN also works based on a distance measure. Therefore, a distance measure combined with KNN accuracy will not require additional computation cost. Equation (3.4) shows a new fitness function that combines the distance measure [12] and KNN accuracy using a weighting coefficient (α).

$$fitness = (\alpha \cdot balanced\_accuracy + (1 - \alpha) \cdot distance) \tag{3.4}$$

where

$$distance = \frac{1}{1 + exp^{-5(D_b - D_w)}} \tag{3.5}$$

$$D_b = \frac{1}{M} \sum_{i=1}^{M} \min_{\{j|j \neq i, class(I_i) \neq class(I_j)\}} Dis(I_i, I_j) \tag{3.6}$$

$$D_w = \frac{1}{M} \sum_{i=1}^{M} \max_{\{j|j \neq i, class(I_i) = class(I_j)\}} Dis(I_i, I_j) \tag{3.7}$$

D_b is the average distance between each instance and the nearest instance of other classes. D_w is the average distance between each instance to the farthest instance of the same class. $Dis(I_i, I_j)$ is any measure used to approximate the distance between two instances (or vectors) I_i and I_j. Here the number of matches or overlapping between two nominal vectors divided by the size of the vectors is used to approximate the distance between two instances. Since KNN also works based on this overlapping distance calculation, adding this distance component to the fitness function does *not* increase (much) the computation time to the evaluation process.

Although both D_b and D_w are in the same range of [0,1], 1 is considered as the best case for D_b and the worst case for D_w. Using the difference $D_b - D_w$, Equation (3.5) maximises D_b and minimises D_w in order to find feature subsets that keep the same-class instances close together and different-class instances far away. Since the value of $(D_b - D_w)$ ranges from -1 to 1, coefficient -5 is used in the logistic function to scale it into the full range [0,1] in which 0 is the worst *distance* and 1 is the best *distance* as shown in the right plot of Figure 3.2. Note that the logistic function with coefficient -1 shown in the left plot of Figure 3.2 does not return a value in the full range [0,1] for an input between -1 and 1. As a result, Equation (3.4) shows a maximisation fitness function.

3.2.3 Fast Fitness Evaluation in "Local Search"

The introduction of the local search brings a number of extra fitness evaluations. In wrapper approaches, each evaluation requires a process of training

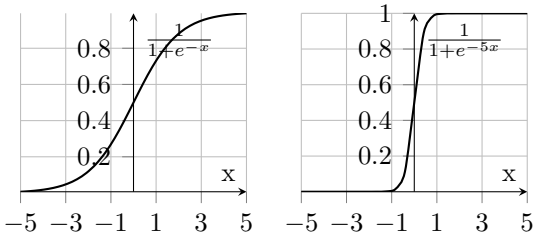


Figure 3.2: Logistic function $f(x) = \frac{1}{1+e^{-x}}$, $f(x) = \frac{1}{1+e^{-5x}}$.

and testing a classifier to get the classification performance as the fitness value. Therefore, most of the computational cost is spent on the fitness evaluations. In PSO-LS, each flipping step in the local search brings an extra calculation of the fitness value (i.e. the classification error rate), which may cause a significant increase in the computational cost.

To address this issue, a fast fitness evaluation strategy is proposed for the local search, which is designed by considering the characteristics of KNN classification algorithm and the local search. In KNN, the class label of an unseen instance is determined according to its distance to the training instances. The overall distance between the unseen instance and a training instance can be calculated based on the sum of their distances in each feature. Therefore, when adding or removing features, the overall distance value can be calculated/updated by adding or subtracting their distances in these features only. In PSO-RLS, each flipping step changes only a small percentage (2%) of the total dimensions/features while 98% of them remain the same. Instead of re-calculating the distance between an unseen instance and a training instance, the fast fitness evaluation strategy only re-calculates the 2% of the total features. To achieve this, at the beginning of each “local search" run, all the distances between the unseen instance and the training instances are calculated based on the features selected by the given *pbest* and stored in a square matrix (distance[i][j]). Since this matrix is symmetric, $\frac{M(M+1)}{2}$ distances are calculated with M instances. When a flipping step is performed, using the distances stored in the matrix can speed up the computation of

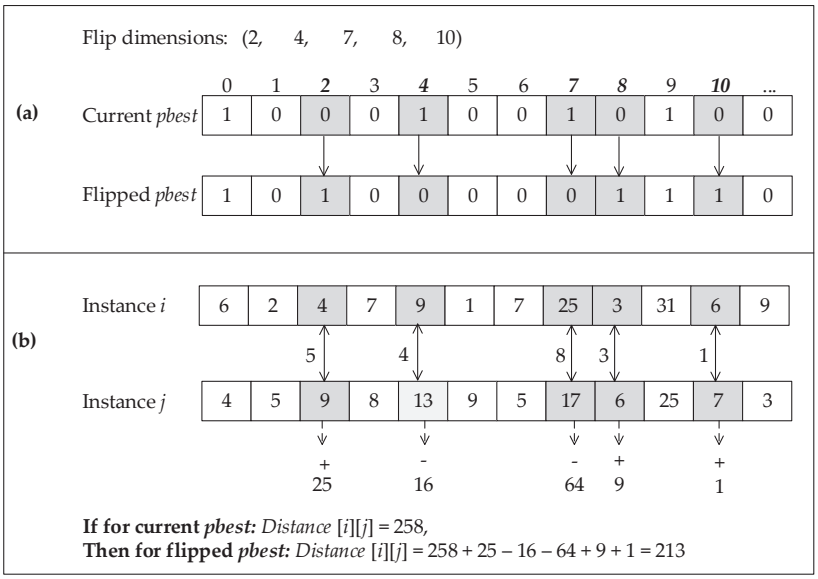


Figure 3.3: An example of re-calculating instances' distance in a local search step.

finding the nearest neighbours of a certain instance by recalculating only 2% of the dimensions. Similarly, PSO-CLS also has the same benefit when using this strategy.

3.2.4 Illustration of the "Local Search"

Figure 3.3 illustrates the process of the local search and the fast fitness evaluation process. Figure 3.3(a) shows an example of the flipping procedure, where *pbest* is converted to a binary vector. Based on the randomly chosen flipping dimensions, e.g. (2, 4, 7, 8, 10), the current *pbest* can be flipped to obtain a new *pbest* position (flipped *pbest*).

Figure 3.3(b) shows an example of the fast fitness evaluation process between two instances: I_i and I_j, where the overall distance is the sum of the squared distance between I_i and I_j in all dimensions/features. Suppose the overall distance between I_i and I_j according to the current *pbest* ($distance[i][j]$) is 258, their overall distance according to the flipped *pbest* can

be re-calculated by changing values in the five flipped dimensions. Features 4 and 7 are flipped from selected to not selected, so their distances will be subtracted from 258. Meanwhile, Features 2, 8 and 10 are flipped from not selected to selected and their distances will be added to 258. The new distance therefore is 258 - 16 - 64 + 25 + 9 +1 = 213. After calculating the overall distance with all the training instances, the KNN algorithm can quickly find the new nearest neighbour to determine the class label of the unseen instance. As a result, a large amount of time can be saved in the local search process compared with calculating all distance again.

3.3 Experiment Design

3.3.1 Datasets

In order to examine the performance of PSO-LS, a set of experiments were conducted on ten gene expression datasets downloaded from http://gems-system.org/. The details of these datasets can be seen in Table 1.1 on Page 21. Before applying feature selection, all datasets are discretised using minimum description length (MDL) method [70] in order to use symmetric uncertainty which can only be applied to categorised data.

Since the numbers of instances of these datasets are very small compared with the number of features, 10-fold cross-validation is used to create training and test sets by stratified splitting from the original dataset as shown in Figure 2.3. In this setting, 10 pairs of training and test sets are created for each dataset. In each pair, a different fold is used as the test set and the remaining folds form the training set. During the feature selection process, only the training set is used. The returned feature subset will be evaluated using the corresponding left-out test set. The classification accuracy is then averaged from the results of 10 runs.

3.3.2 Baseline Methods

The performance of PSO-RLS and PSO-CLS were tested by comparing the classification accuracy of the selected features versus the original features, the feature subsets selected by standard PSO, the PSO with resetting *gbest* (PSO-RG) [50]. We also compared PSO-CLS with three deterministic feature selection methods, which are linear forward selection (LFS) [90], greedy stepwise backward selection (GSBS), correlation-based feature selection (CFS) [94]. Details of these methods are described in Section 2.4.1.1. We choose these methods to compare because they are representative for single stage feature subset selection methods that are similar to the proposed algorithms in a way that no predefined number of selected features is required. The baseline methods are run using Weka package with default settings.

3.3.3 Experiment Configuration and Parameter Settings

Table 3.1 shows the parameter settings used in the experiments. All the four PSO based algorithms share the same settings for PSO parameter that are $c1 = c2 = 2.0$, w linearly decreases from 0.9 to 0.4 [211]. Fully connected topology is used and the maximum velocity is 0.6. The threshold $\theta = 0.6$ [252] is used to determine whether a feature is selected. The maximum number of iterations is 70. Since the 10 datasets have quite different numbers of features ranging from 2,000 to 12,000, which means that their search spaces are also very different, the population size is set to one-twentieth of the number of features, but limited to 300 due to memory limit.

In PSO-RG, if *gbest* is not improved for three iterations, it is reset to all 0, which is the same as in [50] for comparison purposes. In PSO-RLS and PSO-CLS, 100 times of flipping will be performed to find a better *pbest* in each local search. In PSO-RLS, a fix flipping size of 2% of the original dimension is used to maintain the same small portion of changes in the new pbest. 2% is chosen because it is small enough to maintain a local search

Table 3.1: Parameters settings

Parameter	**Parameter value**
Acceleration constants ($c1 = c2$)	2.0
Inertia weight (w)	$0.9 - 0.5 * \frac{current\ iteration}{max\ iteration}$
Communication topology	Fully connected
Velocity range	[-0.6..0.6]
Threshold for selected feature (θ)	0.6
Maximum iteration	70
Population size	#features/20 (restricted to 300)
gbest resetting	*PSO-RG: gbest* is not improved for 3 iterations
Local search tries	100
Local search flipping size	– *PSO-RLS:* 2% of #features – *PSO-CLS:* 25% of current *pbest*'s size
Local search frequency	– *PSO-RLS:* every iteration – *PSO-CLS:* the first ten odd iterations
Stopping criterion	– *PSO-RLS:* reaching maximum iteration – *PSO-CLS:* reaching maximum iteration or having the same *gbest* for 10 iterations

and not too small to lower the convergence rate. In PSO-CLS, the flipping size is dynamically determined as shown in Algorithm 2. The number of features are considered in the flipping process is proportional to the current *pbest* size. 25% of the current *pbest* size is chosen to encourage removing more redundant features and adding more relevant ones in one local search try. While PSO-RLS runs the local search in every iteration, PSO-CLS only runs in the first ten odd iterations.

The fitness function used in the Standard PSO and PSO-RG was the same as in PSO-RLS which is based on the average balanced accuracy of KNN (K=1, i.e. 1NN) with leave-one-out cross-validation on the training data. The Standard PSO and PSO-RG also follow the same stopping criteria as PSO-RLS, which is when reaching the maximum iteration.

Since PSO is a stochastic optimisation technique, 30 independent runs

with different seeds are executed on each training set. As a result, PSO is run 300 times (30 runs × 10 folds) on each dataset. Wilcoxon rank sum test, a non-parametric statistical significance test, is performed to compare the classification performance of different algorithms, where the significance level is set to 0.05.

3.4 Results and Discussions

Table 3.2 shows the experimental results of the four PSO algorithms: PSO, PSO-RG, PSO-RLS, and PSO-CLS. "Full" means the original feature set is used for classification. "Ave-Size" shows the average number of features selected by each method over the 30 runs. "Best", "Mean" and "StdDev" show the best, the average, and the standard deviation of the classification accuracy achieved by each method in the 30 independent runs. The last two columns show the results of the significance tests comparing PSO-RLS, or PSO-CLS with the other corresponding algorithms. For example, the "+" ("–") in the "T_{RLS}" column means the corresponding method achieves significantly better (worse) accuracy than PSO-RLS. "=" means they are similar. Similarly, "T_{CLS}" is the results of the Wilcoxon tests comparing the classification accuracy achieved by other methods versus PSO-CLS. In general, the more "–"s, the better the proposed methods are.

3.4.1 Results of PSO-RLS

PSO-RLS versus Full. As can be seen from Table 3.2, PSO-RLS selected about 25% to 30% of the original features on all the 10 datasets. Using the selected features, KNN significantly improved its classification performance on all datasets except Prostate where both have a similar accuracy. More than 10% increase in average accuracy was found on five datasets with the highest improvement of nearly 14% on Leukemia1. The results showed that PSO-RLS can eliminate a significant number of irrelevant and/or redundant features to improve the classification performance of KNN in almost all cases.

Table 3.2: Experimental results

Dataset	**Method**	**Ave-Size**	**Best**	**Mean±StdDev**	T_{RLS}	T_{CLS}
	Full	2308.0	87.08		–	–
	PSO	915.0	99.17	96.56 ± 1.55	=	–
SRBCT	PSO-RG	606.2	99.17	95.60 ± 1.66	=	–
	PSO-RLS	545.1	99.17	96.08 ± 1.73		–
	PSO-CLS	59.7	100.00	99.97 ± 0.15		
	Full	5469.0	83.00		–	–
	PSO	2279.9	95.83	93.61 ± 2.19	=	+
DLBCL	PSO-RG	305.5	93.33	85.92 ± 4.06	–	–
	PSO-RLS	1417.4	97.33	93.72 ± 1.79		+
	PSO-CLS	47.4	96.67	90.86 ± 3.19		
	Full	5726.0	36.67		–	–
	PSO	2564.2	65.00	51.22 ± 5.23	+	=
9Tumor	PSO-RG	1894.3	60.00	50.22 ± 4.54	+	=
	PSO-RLS	1352.0	58.33	48.39 ± 4.88		–
	PSO-CLS	46.7	60.00	51.39 ± 4.22		
	Full	5327.0	79.72		–	–
	PSO	2143.8	95.56	93.88 ± 1.47	=	–
Leukemia1	PSO-RG	786.1	94.31	89.63 ± 2.96	–	–
	PSO-RLS	1534.9	95.56	93.45 ± 1.71		–
	PSO-CLS	31.9	95.42	94.84 ± 1.16		
	Full	5920.0	72.08		–	–
	PSO	2481.6	80.00	75.89 ± 1.68	=	=
Brain1	PSO-RG	519.7	77.08	72.00 ± 3.13	–	–
	PSO-RLS	1549.0	77.50	75.00 ± 1.80		–
	PSO-CLS	1081.5	82.50	76.78 ± 2.09		
	Full	11225.0	89.44		–	–
	PSO	4577.7	93.89	92.07 ± 1.40	=	–
Leukemia2	PSO-RG	1116.6	95.00	90.72 ± 2.59	=	–
	PSO-RLS	3426.5	93.89	91.72 ± 1.46		–
	PSO-CLS	53.7	98.33	95.56 ± 1.68		
	Full	10367.0	62.50		–	–
	PSO	4249.9	81.25	75.83 ± 2.99	=	+
Brain2	PSO-RG	654.9	85.00	73.74 ± 4.95	=	=
	PSO-RLS	3099.0	82.50	75.35 ± 3.16		+
	PSO-CLS	2647.7	78.75	73.47 ± 2.82		
	Full	10509.0	85.33		=	–
	PSO	4603.1	88.17	85.04 ± 1.59	=	–
Prostate	PSO-RG	873.2	89.33	84.97 ± 2.55	=	–
	PSO-RLS	2690.3	89.17	85.79 ± 1.49		–
	PSO-CLS	2670.3	91.17	86.98 ± 1.76		
	Full	12533.0	71.42		–	–
	PSO	5588.9	87.67	84.26 ± 1.35	=	–
11Tumor	PSO-RG	2108.4	86.82	83.84 ± 2.25	=	–
	PSO-RLS	3163.9	87.77	84.19 ± 1.47		–
	PSO-CLS	266.8	90.72	87.51 ± 1.73		
	Full	12600.0	78.05		–	–
	PSO	5353.3	84.73	83.18 ± 0.77	=	–
Lung	PSO-RG	887.3	84.72	82.13 ± 1.78	–	–
	PSO-RLS	3453.9	86.87	83.50 ± 1.16		–
	PSO-CLS	311.6	96.43	90.78 ± 2.61		

PSO-RLS versus PSO. According to Table 3.2, PSO-RLS selected about 10% to 20% fewer features than standard PSO on all datasets. With this reduction, PSO-RLS maintained PSO classification performance on nine out of the ten datasets. Results show that the proposed random local search effectively removed redundant features to further reduce the feature subset size selected by PSO without degrading its classification performance.

PSO-RLS versus PSO-RG. Compared with PSO-RLS, PSO-RG selected a smaller number of features than PSO-RLS on eight out of the ten datasets. However, according to the statistical test results shown in the T_{RLS} column, its classification performance is significantly worse than PSO-RLS on four datasets, similar on five and better on one where PSO-RG selected 542 more features on average than PSO-RLS. This indicated that by resetting *gbest* to an empty set, PSO-RG facilitates particles search towards smaller feature subsets; however, without guaranteeing to maintain the classification performance.

In general, over the 30 comparisons with Full and the two PSO based feature selection methods using three learning algorithms, PSO-RLS won 13, drew 15 and lost 2. This indicated that a random search mechanism could achieve significantly better classification performance than the compared methods.

3.4.2 Results of PSO-CLS

PSO-CLS versus Full. As can be seen from Table 3.2, PSO-CLS achieved significantly better classification performance than using all features on all datasets. The highest improvement is on 11Tumor with more than 16% on average and 19% in the best run. In seven out of the ten datasets, the feature subsets selected by PSO-CLS are reduced one to two orders of magnitude with the best ratio of 1/209 in Leukemia2. On SRBCT, PSO-CLS selected about 60 features among 2,308 features to achieve 100% accuracy in almost all 300 runs.

PSO-CLS versus PSO. According to Table 3.2, PSO-CLS generated

feature subsets with significantly higher accuracy than PSO on six datasets and selected at least an order of magnitude fewer features than PSO on seven datasets. The highest dimensionality reduction is in Leukemia2 where PSO-CLS selected 85 times fewer features than PSO and still improved more than 3% on the PSO performance. On 9Tumor, PSO-CLS obtained a similar classification performance to PSO with only 46 features which are 2,505 fewer features than PSO. Only on DLBCL, PSO achieved a higher accuracy but with a much larger number of features. Therefore, the proposed local search heuristic and the combined wrapper and filter fitness measure in PSO-CLS has achieved the goal of obtaining smaller feature subsets while maintaining or improving the classification performance.

PSO-CLS versus PSO-RG. Although PSO-RG selected a much smaller number of features than PSO, it still selected many more features than PSO-CLS on seven datasets with an order of magnitude more features on four datasets. According to the significance test results in the last column of Table 3.2, PSO-CLS outperformed PSO-RG on eight datasets and had a similar accuracy on the other two. For example on Leukemia1 and Leukemia2, PSO-CLS selected more than 20 times fewer features than PSO-RG while obtained 5% higher accuracy than PSO-RG on both datasets. On Brain1 and Prostate, PSO-RLS selected larger feature subsets than PSO-RG; however, to achieve 4% and 2% higher accuracy, respectively. This indicated that PSO-CLS can select an appropriate number of important features to improve or maintain the discriminating ability of the feature subset. The results showed that using general knowledge in feature selection to guide the search for smaller feature subsets is better than solely resetting *gbest* to an empty set.

In general, selecting a very smaller number of features, PSO-CLS won 24, drew 4 and lost 2 out of the 30 comparisons with the compared methods. It achieved the best average accuracy on eight out of the ten datasets.

3.4.3 PSO-CLS versus PSO-RLS

As can be seen in Table 3.2, feature subsets selected by PSO-CLS are always smaller than those of PSO-RLS with an order of magnitude fewer features on six datasets. The smaller feature subsets by PSO-CLS achieved a significantly better classification accuracy than PSO-RLS on eight datasets with the highest improvement of 7% in average and 10% in the best case on Lung. On 9Tumor and Leukemia2, while PSO-RLS selected 1,352 and 3,426 features, PSO-CLS selected only 46 and 53 features to further improve 3% and 4% accuracy, respectively. The results show that by using a filter measure to guide the flipping process and a new fitness function, PSO-CLS can effectively choose relevant features and remove redundant ones to obtain much smaller feature subsets with better discriminating power.

3.4.4 PSO-CLS versus Traditional Methods

Table 3.3 shows the compared results of PSO-CLS with LFS and CFS. Besides the size shown in the third column, the best and the mean of the training and test accuracies are shown under each corresponding column. The "S_{Tr}" and "S_{Te}" display the Wilcoxon significance test results (with a 5% significance level) of the corresponding method over PSO-CLS in terms of training and test accuracy, respectively. The results of GSBS are not displayed because GSBS could not finish its run in 12 hours for any of the 10 datasets because backward selection is too slow for high-dimensional data. CFS method is also quite expensive, it fails to finish its run in 12 hours for the two largest datasets that are Lung and 11Tumor with more than 12,000 features.

As can be seen from Table 3.3, PSO-CLS outperformed LFS on almost all datasets in terms of training and test accuracies. At least 10% higher test accuracy was found on nine datasets. It can be noted that on Brain2, while LFS had better training accuracy than PSO-CLS, its test accuracy was 20% lower than PSO-CLS. Although LFS selected the smallest feature subsets among all compared methods on all datasets, it failed to maintain

Table 3.3: Comparison with traditional methods

Dataset	Method	Size	Training			Test		
			Best	Mean	S_{Tr}	Best	Mean	S_{Te}
SRBCT	Full	2,308.0		83.35	–		87.08	–
	LFS	6.1		98.19	–		88.75	–
	CFS	80.9		100.00	=		100.00	=
	PSO-CLS	59.7	100.00	100.00		100.00	99.97	
DLBCL	Full	5,469.0		81.71	–		83.00	–
	LFS	4.0		98.24	–		74.00	–
	CFS	58.0		99.22	–		91.67	=
	PSO-CLS	47.4	100.00	100.00		96.67	90.86	
9Tumor	Full	5,726.0		33.44	–		36.67	–
	LFS	12.6		82.39	–		41.67	–
	CFS	38.0		90.71	–		53.33	+
	PSO-CLS	46.7	97.78	97.78		60.00	51.39	
Leukemia1	Full	5,327.0		79.77	–		79.72	–
	LFS	4.8		99.17	–		81.39	–
	CFS	56.0		100.00	=		93.19	–
	PSO-CLS	31.9	100.00	100.00		95.42	94.84	
Brain1	Full	5,920.0		65.07	–		72.08	–
	LFS	9.9		89.13	–		59.17	–
	CFS	115.4		99.93	–		79.58	+
	PSO-CLS	1081.5	100.00	99.96		82.50	76.78	
Leukemia2	Full	11,225.0		88.82	–		89.44	–
	LFS	4.3		99.08	–		90.00	–
	CFS	79.0		100.00	=		98.89	+
	PSO-CLS	53.7	100.00	100.00		98.33	95.56	
Brain2	Full	10,367.0		63.52	–		62.50	–
	LFS	5.6		98.80	+		53.33	–
	CFS	63.4		100.00	+		71.25	–
	PSO-CLS	2647.7	99.20	98.55		78.75	73.47	
Prostate	Full	10,509.0		82.08	–		85.33	–
	LFS	4.9		82.44	–		73.17	–
	CFS	51.6		98.12	–		90.17	+
	PSO-CLS	2670.3	98.92	98.64		91.17	86.98	
Lung	Full	12,600.0		71.59	–		78.05	–
	LFS	12.2		95.12	–		80.55	–
	CFS	NA		NA			NA	
	PSO-CLS	311.6	99.11	99.02		96.43	90.78	
11Tumor	Full	12,533.0		71.01	–		71.42	–
	LFS	14.3		79.96	–		61.71	–
	CFS	NA		NA			NA	
	PSO-CLS	266.8	100.00	100.00		90.72	87.51	

the baseline accuracy of using Full on five datasets with the most significant drop of 13% on Brain1. In contrast, PSO-CLS achieved better performance than Full in all cases.

Compared with CFS, PSO-CLS obtained a similar or better training accuracy on seven out of eight datasets with smaller feature subsets on four. In terms of test accuracy, PSO-CLS feature subsets obtained a higher accuracy than CFS on two datasets, namely Leukemia1 and Brain2. On SRBCT and DLBCL, PSO-CLS selected a smaller number of features than CFS to achieve a similar accuracy as CFS. On the other four datasets, CFS had better results than PSO-CLS on average but the best accuracy of PSO-CLS is almost always better than CFS.

In general, over the 28 comparisons based on test accuracy, PSO-CLS won 22, drew 2 and lost 4. The results showed that PSO-CLS had better performance than the compared methods on high-dimensional data.

3.5 Further Analysis

3.5.1 Computation Time

Since the local search introduces extra fitness evaluations that require longer computational time, the fast fitness evaluation mechanism was proposed to speed up the process. To see the effect of this fast mechanism on the computational cost of the local search, Table 3.4 summaries the average CPU time used by each method in the 30 independent runs on the ten datasets, where the numbers are expressed in minutes. The last two columns show the ratio between PSO-RLS and PSO-CLS time versus PSO time, respectively.

As can be seen from Table 3.4, among the four methods, PSO-RG required the shortest running time in all cases, where the main reason is that by resetting *gbest*, PSO-RG attracts all particles flying towards the small feature subset space. Therefore, the number of features selected by particles in PSO-RG is usually smaller than in other methods. This confirms the significant influence of the size of feature subsets on the computational time in wrapper

Table 3.4: Average computation time (minutes)

	PSO	PSO-RG	PSO-RLS	PSO-CLS	PSO-RLS vs PSO	PSO-CLS vs PSO
SRBCT	7.19	5.89	43.8	6.51	6.1	0.9
DLBCL	33.73	18.8	156.57	56.82	4.6	1.7
9Tumor	22.92	18.69	85.37	22.45	3.7	1.0
Leukemia1	28.31	20.29	115.14	29.41	4.1	1.0
Brain1	55.67	32.29	262.94	234.65	4.7	4.2
Leukemia2	117.95	75.42	323.01	134.73	2.7	1.1
Brain2	73.31	42.3	214.09	264.71	2.9	3.6
Prostate	112.87	58.04	520.62	548.35	4.6	4.9
Lung	333.52	151.18	3532.11	1830.07	10.6	5.5
11Tumor	393.54	257.24	1772.29	373.76	4.5	0.9

feature selection approaches.

Compared with PSO, PSO-RLS running time is two to ten times longer as shown in the sixth column of Table 3.4. Note that both algorithms were run with the same number of particles and iterations, so they have the same number of fitness evaluations in the PSO search process. However, since PSO-RLS involves a local search process, they involve a much larger number of evaluations than PSO. Every time a particle reaches a new *pbest*, the local search repeats 100 times of flipping to search for a better *pbest*. In other words, it will call 100 fitness evaluations. For example, in DLBCL, the population size is $5469/20 = 273$ and hence the number of evaluations in PSO is $273 * 70 = 19,110$. If in one PSO iteration, only one-third of the population, which is about 90 particles in this example, can reach new *pbest*, PSO-RLS will have extra 9000 (100*90) evaluations in each iteration. In total, PSO-RLS may have extra 630,000 (9000*70) fitness evaluation, which is about 33 times more than PSO. However, by reaching solutions with smaller subsets and the use of the fast fitness evaluation in local search, the computational time of PSO-RLS on this dataset is just 4.6 times longer than PSO. The results show that this new strategy successfully reduces the running time of 1NN classifier.

Using the same strategy in speeding up fitness evaluation, PSO-CLS even had shorter running time than PSO-RLS on 8 datasets. Note that compared with PSO-RLS, PSO-CLS has extra steps for choosing the flipping features based on their symmetric uncertainty values. Its fitness function also involves an additional filter measure. However, by eliminating redundant features to obtain smaller feature subsets and by running local search only in the first odd iterations, PSO-CLS has significantly reduced its running time while achieved much better discriminating subsets than PSO-RLS. The results confirmed that applying a small number of informed local search is better than applying an extensive number of random local search. With this strategy, PSO-CLS required a similar running time as PSO on 5 datasets and 5.5 times longer than PSO in the largest dataset which is Lung.

In general, the results indicate that the proposed local search heuristic is not only an effective but also efficient to achieve much smaller feature subsets with significantly better classification performance.

3.5.2 Robustness

In the experiments, each algorithm was run 300 independent times producing 300 different solutions for each dataset. An investigation to see if the features selected by the proposed method in all runs are not selected by random chance was done by comparing the Z-score [111, 271] of the top 100 features selected by each method. Z-score of a feature indicates the significance of the selection frequency of that feature. Z-score of feature i is defined as follows:

$$Z_i = \frac{f_i - \mu}{\sigma} \tag{3.8}$$

where f_i is the number of times feature i appeared in S solutions, μ and σ are the mean and standard deviation of f_i. Let A be the average feature subset size of S solutions, N is the total number of features, then the probability of feature i being selected is denoted as, $P(f_i) = A/N$. Using this probability, the mean and the standard deviation of f_i are calculated using $\mu = P(f_i) \cdot S$ and $\sigma = \sqrt{P(f_i) \cdot (1 - P(f_i)) \cdot S}$.

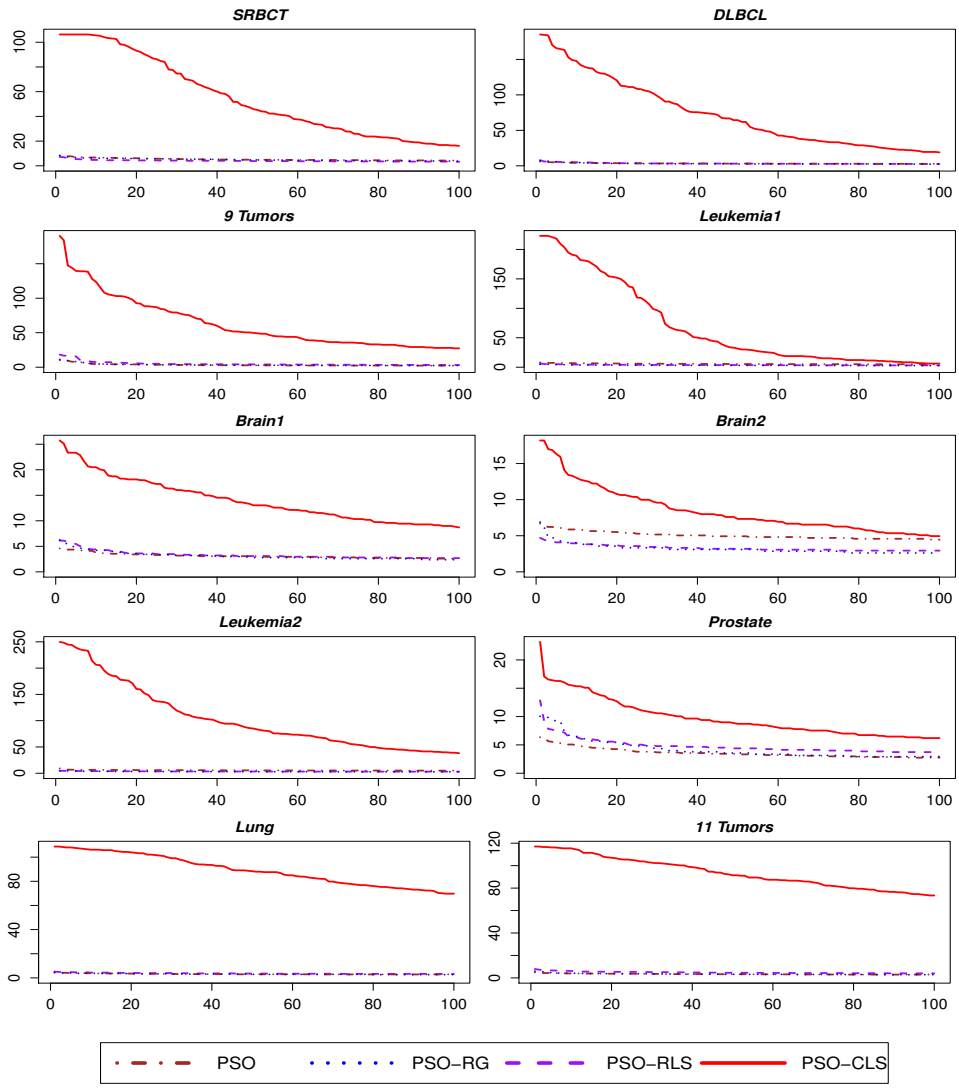


Figure 3.4: Z-score of the top 100 selected features on each dataset.

As can be seen from Equation (3.8), the higher the value of Z_i, the less likely that feature i is selected randomly. Therefore, an algorithm that selects more features with higher Z-score is said to be more robust.

The procedure to compare the robustness of the four methods starts with sorting all features in descending order based on their selection frequency in each method. Z-score of the top 100 selected features by each method on

Table 3.5: Experimental results with feature selection bias

Dataset	**Method**	**Ave-Size**	**Best**	**Mean±StdDev**	T_{RLS}	T_{CLS}
	PSO	930.0	100.00	100.00 ± 0.00	=	=
	PSO-RG	465.4	100.00	99.89 ± 0.37	=	=
SRBCT	PSO-RLS	536.9	100.00	100.00 ± 0.00		=
	PSO-CLS	65.6	100.00	100.00 ± 0.00		
	PSO	2306.2	94.83	94.83 ± 0.00	=	–
	PSO-RG	295.5	98.28	95.14 ± 1.25	=	–
DLBCL	PSO-RLS	1377.6	95.69	94.86 ± 0.28		–
	PSO-CLS	50.5	100.00	100.00 ± 0.00		
	PSO	2594.2	88.89	80.93 ± 3.23	–	–
	PSO-RG	1947.1	92.86	82.22 ± 3.95	–	–
9Tumor	PSO-RLS	1304.4	95.83	86.85 ± 5.08		–
	PSO-CLS	51.7	100.00	96.40 ± 2.80		
	PSO	2159.5	100.00	99.79 ± 0.38	=	–
	PSO-RG	434.9	100.00	99.50 ± 0.63	–	–
Leukemia1	PSO-RLS	1496.3	100.00	99.85 ± 0.33		–
	PSO-CLS	33.1	100.00	100.00 ± 0.00		
	PSO	2501.5	88.00	87.72 ± 0.20	=	–
	PSO-RG	584.1	95.33	89.01 ± 2.28	=	–
Brain1	PSO-RLS	1488.7	91.33	87.93 ± 1.25		–
	PSO-CLS	2100.2	96.67	91.46 ± 2.40		
	PSO	4585.2	100.00	98.97 ± 0.41	=	–
	PSO-RG	511.0	100.00	99.35 ± 0.71	+	–
Leukemia2	PSO-RLS	3335.4	100.00	99.01 ± 0.45		–
	PSO-CLS	56.7	100.00	100.00 ± 0.00		
	PSO	4173.0	94.64	94.64 ± 0.00	=	=
	PSO-RG	605.9	96.43	94.52 ± 1.23	=	=
Brain2	PSO-RLS	3132.9	94.64	94.64 ± 0.00		=
	PSO-CLS	4830.6	96.43	94.70 ± 0.33		
	PSO	4590.3	91.19	90.89 ± 0.47	=	–
	PSO-RG	761.3	93.19	90.79 ± 1.29	=	–
Prostate	PSO-RLS	2676.7	91.19	91.02 ± 0.38		–
	PSO-CLS	2560.2	96.12	92.51 ± 1.25		
	PSO	5448.7	91.23	88.94 ± 1.72	=	–
	PSO-RG	767.7	91.80	89.10 ± 1.70	=	–
11Tumor	PSO-RLS	3363.3	91.37	88.75 ± 1.62		–
	PSO-CLS	347.1	99.86	99.38 ± 0.49		
	PSO	5702.4	98.08	96.81 ± 0.76	–	–
	PSO-RG	2517.3	99.24	97.04 ± 1.16	=	–
Lung	PSO-RLS	3074.7	99.24	97.39 ± 0.90		–
	PSO-CLS	297.1	100.00	99.54 ± 0.38		

each dataset were plotted in Figure 3.4. As can be seen from these plots, the features selected by PSO-CLS had much larger Z-score than other methods. This indicated that feature subsets selected by PSO-CLS in different runs had much more overlap than other methods. In other words, PSO-CLS consistently chose similar features across different runs with different random seeds and training datasets. Furthermore, PSO-CLS's Z-scores started at a very high value and gradually decreased, forming a gentle slope in almost all cases, which indicated that more relevant features were selected by PSO-CLS. The results showed that PSO-CLS was more robust than the other methods.

3.5.3 Feature Selection Bias

As mentioned in Section 2.2.3.1, feature selection bias usually happens in studies where datasets have a small number of instances as the datasets that are used in this thesis. In many studies [2, 5, 13, 20, 50, 106, 158, 161, 162, 205, 210, 261], the whole dataset is used during the feature selection process as shown in Figure 2.2. There is no separate unseen data to test the generality of the selected features. According to Ambroise and McLachlan [17], there is a feature selection bias issue in these studies, so one cannot claim that the selected features can be used for future unseen data.

This problem is investigated here by conducting another set of experiments of all the compared PSO methods on the same datasets using the same parameter settings except that the whole dataset is used by PSO to evolve the feature subsets. In other words, this set of experiments follows the system shown in Figure 2.2 (with feature selection bias), while the experiments in Section 3.4 follows the system shown in Figure 2.3 (without feature selection bias). Table 3.5 shows the experimental results with feature selection bias of the four PSO algorithms: PSO, PSO-RG, PSO-RLS, and PSO-CLS. The same format and meaning as Table 3.2 are used in this table.

A comparison between Table 3.2 that does not have feature selection bias and Table 3.5 that has feature selection bias showed that all the four algorithms obtained a much higher accuracy in the latter than the former.

The highest difference can be seen in 9Tumor with 30% to 45% increase in the results having feature selection bias. With feature selection bias, PSO-CLS achieved the best accuracy among the four algorithms on all the ten datasets. It obtained 100% accuracy in all the 300 runs on four datasets, which can be 10% higher than the results in Table 3.2.

The performance gap between the four methods decreased in some datasets and increased in some others in the experiments having feature selection bias. In SRBCT and Brain, the performance difference became smaller or even neglectable, while their results in Table 3.2 showed a different pattern. On the other hand, while PSO-CLS had a similar accuracy as PSO and PSO-RG on 9Tumor in the previous experiment, it obtained a significantly better results than the other two methods in this experiment. In general, the comparison showed that in many cases the patterns found from the results with feature selection bias cannot be generalised on those without feature selection bias. This suggested that a feature selection method that was tested with feature selection bias might not be a good feature selection approach in practice.

Although feature selection bias should be avoided in most cases, there still can be situations where feature selection bias does not lead to misleading observations or conclusions. For example, in some engineering problems, the goal of feature selection is to find factors or features that significantly influence the model performance. There is no unseen data to be tested on and the whole set of data (with feature selection) can be safely used to find such important features. Feature selection bias is also not a problem in some biological analysis tasks in which feature selection is used to find key genes or proteins without the need of testing them on unseen data.

3.5.4 Discussion on Closely Related Work

A new local search approach to using PSO for FS called HPSO-LS [164] was proposed recently in 2016. In HPSO-LS, a local search procedure is used to add relevant features and remove redundant features to maintain 65% of relevant features and 35% of redundant features in each particle position

(i.e. feature subset). The number of selected features by each particle is initially fixed based on a predefined parameter and kept unchanged by the local search.

Although both HPSO-LS and PSO-CLS use general knowledge about features to add and remove features, they work in different ways. Unlike the local search proposed in this chapter where the new solution found by the local search is only updated back to the particle if it achieves better performance, the proposed local search in [164] is always applied on each particle right after the particle updating step and before fitness evaluation. Although this strategy does not involve more evaluation time, it is likely to cancel the good results shared among particles through communication, which is an essential characteristic of PSO. Furthermore, restricting the number of features selected by particles to a predefined one also limits PSO from showing its fundamental strength in automatically choosing an appropriate subset size. The performance of HPSO-LS will highly depend on the predefined number of selected features.

HPSO-LS was tested on 13 datasets having 9 to 7129 features which are not used in this chapter. The average results of HPSO-LS over 20 independent runs have shown to outperform standard PSO, GA, ACO, an improved PSO method using new initialisation method called PSO(4-2) [255] and other traditional FS methods such as information gain, F-score, and term variance. However, the results are confusing since the paper did not describe how PSO(4-2) can produce a feature subset with a predefined size while this feature is not designed in the original method [255]. The same question arises for other baseline methods.

3.6 Chapter Summary

The overall aim of this chapter was to investigate a new PSO and local search based feature selection approach to high-dimensional classification. The goal has been achieved by developing two PSO methods namely PSO-RLS and

PSO-CLS. While PSO-RLS employs a random flipping mechanism to produce a new candidate solution based on the newly found *pbest*, PSO-CLS choose some redundant features to eliminate and some relevant ones to add to *pbest*. An efficient local search was proposed to speed up the evaluation time in the local search of both methods.

Ten high-dimensional datasets of varying difficulties were used to compare the proposed methods with standard PSO, PSO with *gbest* resetting mechanism (PSO-RG). The experimental results showed that PSO-RLS had better performance than the original feature sets, standard PSO and PSO-RG in terms of feature subset size and/or classification accuracy. Compared with PSO-RLS, the use of symmetric uncertainty in PSO-CLS to choose features for flipping helps local search effectively discover relevant and redundant features. As a result, PSO-CLS obtained much smaller feature subsets with higher discriminating ability than PSO-RLS. PSO-CLS's performance is confirmed by the analysis on the Z-score of the top 100 selected features. The results show that PSO-CLS selected more relevant features than the other compared methods. Analyses of the running time revealed that although PSO-RLS and PSO-CLS had a much larger number of fitness evaluations than PSO and PSO-RG, their computation time is not as much longer as they should thanks mainly to the fast fitness evaluation procedure. Comparisons between PSO-CLS and two traditional feature selection methods, namely LFS and CFS, in both scenarios with and without feature selection bias have shown that PSO-CLS outperformed LFS in almost all cases and had better performance than CFS in terms of feature subset size and/or classification accuracy in many cases.

The results and analyses have shown that the proposed local search has effectively and efficiently enhanced PSO performance by balancing its exploration and exploitation. PSO-CLS results also suggested that local search using general knowledge in feature selection can significantly improve the performance of PSO on high-dimensional data.

In this chapter, symmetric uncertainty was used to evaluate feature rele-

vance and redundancy. This measure can only be applied to categorical data. Therefore, all datasets are discretised before applying feature selection method using MDL method as mentioned in Section 3.3. Since MDL discretises feature individually, if feature interactions exist in a dataset, the discretised data may lose the interaction information. Therefore, the performance of the feature selection in the later stage may be affected by this process. This problem will be addressed in the next chapter.

4

PSO Based Feature Selection via Discretisation

4.1 Introduction

In data preprocessing, discretisation is an important and popularly used technique to transform continuous data into discrete values. It aims at maintaining essential information while eliminating the minor fluctuations that may be noisy in the original data so that learning algorithms can easily learn the target concept. Therefore, discretisation can improve the performance of many machine learning algorithms [148]. Feature selection methods have also shown to obtain better results on discrete data [31].

During the discretisation process, the information used to obtain discrete features that can satisfy the above criteria also reflects their relevance to the target concept. Because of this, *feature selection via discretisation* has been proposed to select features based on the results of the discretisation process [142, 105]. However, for the sake of efficiency, these methods often

use univariate discretisation where features are discretised individually. This scheme works based on the assumption that each feature independently influences the task (i.e. there is no interaction between features), which may not hold in most real-world problems. Therefore, the performance of the feature selection stage may be degraded since information showing feature interactions may be lost during the discretisation process. With the ability of simultaneously optimise multiple variables, PSO may achieve better performance with feature selection via multiple (features) discretisation. However, this approach has never been investigated in the literature.

4.1.1 Chapter Goals

The goal of this chapter is to develop a new PSO based approach to feature selection via discretisation for high-dimensional continuous data. As a first work in using PSO for discretisation-based feature selection, binary discretisation was used for simplicity. In other words, the proposed methods evolve a single cut-point for each feature. Two new methods called EPSO and PPSO, are proposed based on two new representations to achieve this goal. While EPSO used PSO to directly *evolve* a cut-point for each feature, PPSO allows PSO to automatically choose *potentially* good cut-points for discretisation and feature selection. EPSO and PPSO are compared with using the original feature set, and the two-stage approach of discretisation and feature selection on high-dimensional data. Our specific research objectives include the following:

1. How to perform multivariate discretisation and feature selection in a single stage to improve the discriminating power of the feature set;

2. Whether EPSO obtains better performance in terms of classification accuracy and feature subset size than using all features and the two-stage approach;

3. Whether PPSO outperforms using all features and the two-stage approach in terms of classification accuracy and feature subset size;

4. Which of the two proposed methods, EPSO and PPSO, has better performance;

5. Whether the proposed approach achieves better classification performance than traditional methods in both cases with and without *feature selection bias*;

6. Whether the proposed approach is more efficient than the two-stage approach and scales better to high-dimensional data; and

7. Whether the features generated by results of the proposed approach are robust and generalised well to other classification algorithms than the wrapped learning method.

4.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. Section 4.2 presents the overall system as well as details of EPSO and PPSO. Section 4.3 describes the information related to the experiments including the datasets, the baseline methods used to compare with the proposed methods, the parameter settings, the termination criteria and the configuration of the experiments. Results of both methods are presented and discussed in Section 4.4. Further discussion about generalisation, robustness and learning capability of the proposed methods is given in Section 4.5. Finally, Section 4.6 provides a summary of this chapter.

4.2 The Proposed Algorithms

In order to achieve feature selection via discretisation using PSO, PSO representation must be able to encode a discretisation solution. As the first work on PSO based feature selection via multi-variate discretisation, the proposed method used binary discretisation where each continuous feature value is discretised into either 0 or 1 using a single cut-point. Therefore, a

discretisation solution, in this case, is a set of N cut-points corresponding to the N original features. PSO's task is to evolve a set of good cut-points so that the corresponding discrete data can improve their discriminating ability. The evolved cut-points are then used as criteria to select features.

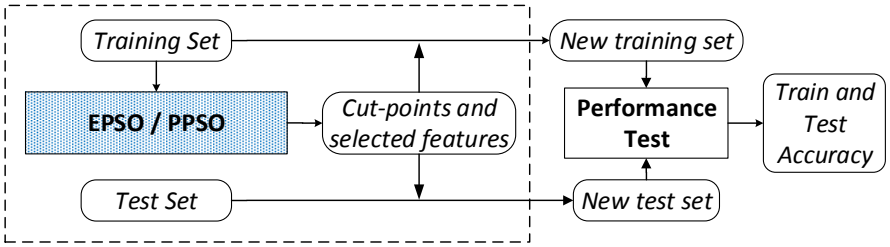
This chapter proposed to use a derived type of PSO called "bare-bones" PSO (BBPSO) [115] to simultaneously discretise and select features because of the following reasons. In PSO for feature selection, the PSO representation is usually an N-dimension vector corresponding to N features. Each value is of the range [0,1] and represents the likelihood of being selected of the corresponding feature. If it is greater than the predefined threshold, the corresponding feature is selected and vice versa, regardless of how much smaller or greater it is when compared to the threshold. Therefore, two different evolved vectors may result in the same feature subset. On the other hand, in PSO for discretisation, the evolved value represents a cut-point whose slight change may lead to a different discrete feature. As a result, finding a good cut-point requires a fine-tuned search mechanism which can be found in BBPSO.

In BBPSO, new positions are sampled using a Gaussian random generator with the centre being the mean position of the individual best position (*pbest*) and its neighbours' best position (*gbest*). The standard deviation is the distance between them. Therefore, when the difference between these two bests is large, the variance enables particles to explore new regions in the space. When they are closer, the new position is limited to a smaller region around this mean.

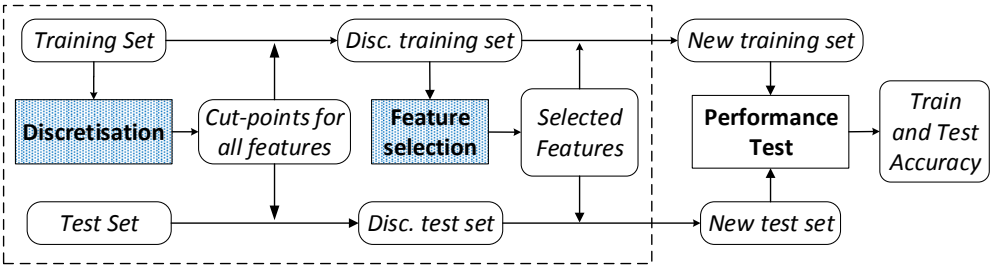
Based on BBPSO, EPSO and PPSO are proposed. While EPSO aims to directly *evolve* a cut-point for each feature, PPSO focuses on choosing one from many *potentially* good cut-points.

4.2.1 Overall Structure

The overview of the proposed approach to multivariate discretisation and feature selection in one stage is shown in Figure 4.1(a). The two-stage



(a) EPSO and PPSO (Discretisation and feature selection in a single stage).



(b) PSO-FS (Discretisation and feature selection in two stages).

Figure 4.1: System Overview.

approach (named PSO-FS) is shown in Figure 4.1(b) in which features are first discretised and then selected. Except for the details inside the dotted box, both systems have the same structure. Firstly, the dataset is split into a training and a test set. Based on the training set, the system evolves a set of cut-points for a set of selected features that are used to transform the training and test sets. These transformed datasets (i.e. the new training set and new test set) are input to the classification algorithms for performance evaluation.

4.2.2 EPSO and PPSO basic steps

Both EPSO and PPSO follow the basic steps shown in Figure 4.2. After initialisation, particles are iteratively evaluated and updated until the stopping criterion is met. In order to evaluate a particle, the training data is first

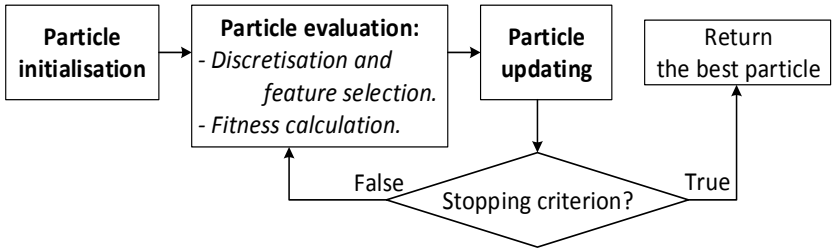


Figure 4.2: EPSO and PPSO basic steps.

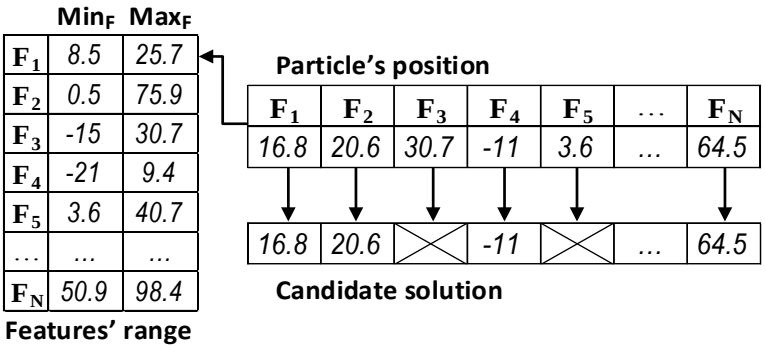


Figure 4.3: Particle representation of EPSO.

discretised and features are selected based on the evolved cut-points. The transformed data is then put into a learning algorithm to calculate the fitness. Based on this fitness, *pbest* and *gbest* are updated and used to update a particle's position as described in Equation (2.8).

Discretisation and feature selection steps in both methods work based on the same principle. To achieve *discretisation*, a feature value is converted/discretised into 0 if it is smaller than the evolved cut-point; otherwise, it will be 1. If all values of a feature are converted into the same discrete value, it is considered as an irrelevant or useless feature because it cannot distinguish instances of different classes. *Feature selection* is done by eliminating these useless features. The remaining discrete features are evaluated based on the improvement in classification performance of the whole discretised data. Using a single step to evaluate both discrete and selected features enables EPSO and PPSO to take into account *possible interactions* between features.

4.2.3 EPSO: PSO for *Evolving* a Cut-Point

4.2.3.1 Particle Representation

The main idea of EPSO is using BBPSO to directly *evolve* a cut-point which can be any real value falling in the range of the corresponding feature values $[Min_F - Max_F]$ in the training set, where Min_F and Max_F are the smallest and largest values of feature F, respectively. The position of each particle represents a candidate solution which is a vector of real-values with N dimensions corresponding to the dimensionality of the problem. Figure 4.3 shows an example of a particle's position and its corresponding candidate solution. In this example, the first dimension of the particle, which represents the cut-point for the first feature (F_1), needs to have a value within the range [8.5, 25.7]. If an updated cut-point of a feature F is out of this range, it will be set to the nearest boundary.

4.2.3.2 Particle Initialisation

Because the search space of multivariate discretisation on high-dimensional data is huge, each initial particle is restricted to a random feature subset of size 50 for binary-class problems and size 150 for multi-class problems as suggested in [271]. This means that for those features that are not selected in the initial candidate solutions, their cut-points will be set to the maximum values of the corresponding features. For the other selected features, their cut-points are initialised using the best entropy-based cut-point that satisfies minimum description length principle (MDLP) [70]. In principle, they can be initialised based on any value within the range of the corresponding feature. However, completely random initial cut-points may lead to slow convergence. Furthermore, information gain in partitioning feature F using the best cut-point T (calculated based on Equation (4.1)) is an indicator of its relevance to the class label. Therefore, features with larger information gain will have a higher probability to be selected in the initialisation procedure.

$$Gain(T, F; S) = E(S) - \frac{|S_1|}{|S|}E(S_1) - \frac{|S_2|}{|S|}E(S_2) \tag{4.1}$$

where S is the set of feature F values, $E(S)$ is the entropy of S. S_1 and S_2 are subsets of S after partitioning F based on cut-point T. Further details are explained in Section 2.1.4 on Page 29.

4.2.3.3 Particle Evaluation

Based on the cut-points evolved by the particle, the training data is transformed into a new training set with discrete values with a smaller number of features thanks to eliminating features whose cut-points are equal to the minimum or maximum values. For example, in Figure 4.3, F_3 cut-point is equal to its maximum value and F_5 cut-point is equal to its minimum value, both features will be discarded.

The discretisation and feature selection solution in each particle is then evaluated based on the classification accuracy of the transformed training set. By evaluating the whole discrete data, the proposed method is able to evaluate cut-points of all the selected features as well as implicitly consider possible feature interactions at the same time. The fitness function uses the balanced classification accuracy [187] shown in Equation (4.2), which is the same as in Chapter 3.

$$balanced\_accuracy = \frac{1}{c}\sum_{i=1}^{c}\frac{TP_i}{|S_i|} \tag{4.2}$$

where c is the number of classes of the problem, TP_i is the number of correctly identified instances in class i and $|S_i|$ is the sample size of class i. All classes are treated equally with the weight of $1/c$. Algorithm 3 shows the pseudo code of EPSO.

Algorithm 3: EPSO pseudo code

Input : Original training set T
Output : Selected features and their cut-points

```
1  begin
2      Best_cut_points ← the best entropy cut-points that satisfy Equation (2.3)
         of each feature;
3      Initialise particles using Best_cut_points;
4      while Stopping criteria are not met do
5          for each particle i do
6              T'_i ← Discretise T based on the evolved cut-points of particle i;
7              T'_i ← Remove one-value features from T'_i;
8              φ_i ← Evaluate the accuracy of T'_i using Equation (4.2);
9              ξ_i ← pbest's fitness;
10             if φ_i is better than ξ_i then
11                 Update pbest ;
12             end
13         end
14         Update gbest;
15         for each particle i do
16             for each dimension d do
17                 Update position x_id using Equation (2.8) ;
18                 if x_id is out of the valid range of feature d then
19                     x_id ← nearest boundary of feature d
20                 end
21             end
22         end
23     end
24     Return the selected features and their cut-points from gbest's position;
25 end
```

4.2.4 PPSO: PSO for Choosing a *Potentially* Good Cut-Point

In the first proposed method (EPSO), BBPSO is free to evolve a cut-point within the range of the corresponding feature. This may result in a huge

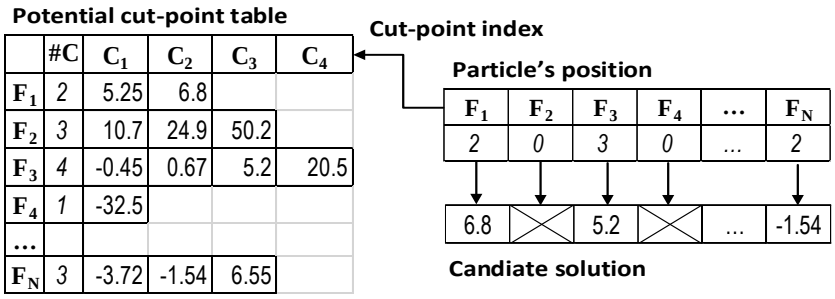


Figure 4.4: Particle representation of PPSO.

search space especially in multivariate discretisation on high-dimensional data. Therefore, PPSO uses BBPSO to choose a cut-point from *potential* cut-points of each feature to narrow the search space into potentially promising areas.

4.2.4.1 Particle representation

Potential cut-points are entropy-based cut-points that have their information gain satisfying the MDLP criterion shown in Equation (2.3) on Page 31. Each feature may have a different number of potential cut-points which are calculated and stored in a potential cut-point table. Figure 4.4 shows an example of this table and a particle's position as well as the corresponding candidate solution. Each particle position is an integer vector denoting the chosen cut-point indexes. The size of the vector, therefore, is equal to the number of the original features and the evolved value needs to be between 1 and the number of potential cut-points of the corresponding feature. For example, in Figure 4.4, the first feature (F_1) has two potential cut-points with index 1 and 2. Therefore, the first dimension of the particle needs to fall in the range [1,2]. If it is 2 as in the example, then the cut-point 6.8 is chosen to discretise F_1.

During the updating process, if the updated value of a dimension is outside the cut-point index range, then it is set to 0, which indicates that the corresponding feature does not have a good cut-point and therefore should be ignored, not selected.

4.2.4.2 Particle Initialisation

Each particle position is initialised as a random feature subset with a restricted number of selected features. Similar to EPSO, features with higher information gain will have higher chance to be selected. Then, for those features that are selected, their corresponding positions in the particle will be set to the indexes of their best MDLP cut-points. For the remaining not selected features, their positions will be set to 0.

In order to make PPSO general to all problems, one restricted size of initial feature subset is applied to all datasets. Then during the evolutionary process, when BBPSO seems to get stuck in local optima, all particles will be reset with a larger size if the current *gbest* fitness is better than the last *gbest* fitness at least 1%. The aim of this *scaling mechanism* is to start the search from small feature subsets while leaving open the chances for larger and better feature subsets. At the beginning, PPSO initialises particles with the subset of 150 features which is suggested in [271] for multi-class datasets. However, based on the experiments, this value is also a good initial size for binary class problems since PSO is able to choose an appropriate feature subset size during the evolutionary process. This initial size increases by 50 features every time the scaling criterion is met.

4.2.4.3 Particle Evaluation

Based on the chosen cut-point index for each feature, the cut-point value is retrieved from the potential cut-point table. It is then used to discretise the corresponding feature. If the evolved cut-point index of a feature is 0, that feature is considered as not selected. For example, Features F_2 and F_4 are not selected in the example shown in Figure 4.4.

In EPSO, classification accuracy is used as the fitness measure to evaluate each particle. This may be difficult for PSO to distinguish between particles that obtain the same accuracy, which may happen in problems where the margin for separating instances between different classes is quite large, enabling

many different models to obtain the same accuracy. Furthermore, while the wrapper methods can produce solutions with high accuracy, filter methods are usually faster and more general. Combining the strength of these two approaches in the evaluation function is expected to produce better solutions. In addition, combining the two measures can also better distinguish the subtle difference between feature subsets, providing a smoother fitness landscape to facilitate the search process. However, simply combining these measures may be impractical in terms of computation time. Therefore, a smart way is needed to combine them without requiring more running time. Among the commonly used filter measures, distance is a multivariate measure that can evaluate the discriminating ability of a feature set and it is used as the base measure of KNN. Incorporating this measure with the KNN wrapper method will not increase the computation time much because the distance measure is calculated only once but used twice. Therefore, in the fitness function of PPSO, balanced KNN classification accuracy is combined with a distance measure using a weighting coefficient (γ) as described in Section 3.2.2. Its description is repeated here for reading convenience.

$$fitness = (\gamma \cdot balanced\_accuracy + (1 - \gamma) \cdot distance) \tag{4.3}$$

where

$$distance = \frac{1}{1 + exp^{-5(D_b - D_w)}} \tag{4.4}$$

$$D_b = \frac{1}{M} \sum_{i=1}^{M} \min_{\{j|j \neq i, class(I_i) \neq class(I_j)\}} Dis(I_i, I_j) \tag{4.5}$$

$$D_w = \frac{1}{M} \sum_{i=1}^{M} \max_{\{j|j \neq i, class(I_i) = class(I_j)\}} Dis(I_i, I_j) \tag{4.6}$$

D_b is the average distance between each instance and the nearest instance of other classes. D_w is the average distance between each instance to the farthest instance of the same class. $Dis(I_i, I_j)$ is the number of matches or overlapping between two nominal vectors divided by the size of the vectors, which is as in Chapter 3. Since KNN also works based on this overlapping distance

calculation, adding this distance component to the fitness function does *not* substantially (except for simple additions and multiplications) increase the computation time to the evaluation process.

Although both D_b and D_w are in the same range of [0,1], 1 is considered as the best case for D_b and the worst case for D_w. As a result, Equation (4.4) maximises D_b and minimises D_w in order to find feature subsets that keep the same class instances close together and different class instances far away. The logistic function in Equation (4.4) is used to transform the difference between D_b and D_w from the range [-1,1] into the range [0,1]. Therefore, distance has 0 as the worst and 1 as the best case. As a result, Equation (4.3) shows a maximisation fitness function. Algorithm 4 presents the pseudo-code of PPSO.

4.3 Experiment Design

4.3.1 Datasets

The performances of EPSO and PPSO are tested on ten gene expression datasets that are available on http://www.gems-system.org. Table 1.1 on Page 21 describes details about these datasets.

4.3.2 Baseline Methods

The effectiveness of EPSO and PPSO on feature selection via discretisation is tested by comparing the classification performance of KNN on the transformed dataset by EPSO and PPSO with the original dataset using the full set of features. Furthermore, they are also compared with the two-stage method (PSO-FS) to see if combining both feature discretisation and feature selection in a single stage achieves better results than applying them in two stages. In PSO-FS, data is first discretised by MDL [70], then PSO runs on the discrete data to find the best feature subset. In PSO-FS, particles are initialised with the same restricted numbers of selected features as in EPSO.

Algorithm 4: PPSO pseudo code

Input : Original training set T
Output : Selected features and their cut-points

```
1  begin
2      Potential_cut_points ← All the entropy cut-points that satisfy Equation
         (2.3) of each feature;
3      repeat
4          Initialise particles using the index of the best cut-point in
             Potential_cut_points ;
5          while Stopping criteria are not met do
6              for each particle i do
7                  T'_i ← Discretise T based on the evolved cut-points of particle i;
8                  T'_i ← Remove one-value features from T'_i φ_i ← Calculate fitness
                     of particle i based on T'_i using Equation (4.3);
9                  ξ_i ← pbest's fitness;
10                 if φ_i is better than ξ_i then
11                     Update pbest ;
12                 end
13             end
14             Update gbest;
15             if The scaling criterion is met then
16                 Increase the initial size;
17                 break;
18             end
19             for each particle i do
20                 for each dimension d do
21                     Update x_id (position of particle i at dimension d) using
                         Equation (2.8) ;
22                     Adjust x_id to the valid range of feature d;
23                 end
24             end
25         end
26     until The scale criterion is not met;
27     Return the selected features and their cut-points from gbest's position;
28 end
```

Table 4.1: Parameter settings

Parameters	**Settings**
Population Size	#features/20 (restriction to a maximum of 300)
Communication topology	Fully connected
Maximum iterations	70
Stopping criterion	- *EPSO*: When *gbest* not improved for 10 iterations. - *PPSO*: When *gbest* not improved for 10 iterations and the current *gbest* fitness is not better than the previous *gbest* fitness at least 1%.
Scaling criterion (PPSO only)	When *gbest* not improved for 10 iterations and the current *gbest* fitness is better than the previous *gbest* fitness at least 1%.

PPSO is also compared with several traditional two-stage methods which combine MDL [70] for discretisation with LFS [90], with consistency method [58] and correlation-based feature selection method (CFS) [94] for feature selection. It is also compared with the Modified Chi2 (or MChi2) [226], which is a representative method for feature selection via discretisation. Weka package [93] was used to run LFS, CFS and Consistency, and the KEEL package [14] to run MChi2.

4.3.3 Parameter Settings and Termination Criteria

Parameter settings for the three compared methods, EPSO, PPSO and PSO-FS, are described in Table 4.1. The size of the search space is proportional to the dimensionality of the problem, which varies from one dataset to another. The number of features in the ten datasets ranges from about 2,000 to 12,000, leading to very different sizes of search space. Therefore, the population size is set to the number of features divided by 20 with a restriction to 300 due to the limited computer memory. The maximum number of iterations was set

to 70.

In EPSO, an early stop was applied when *gbest* is not improved after 10 iterations. On the other hand, when this happens in PPSO, the scaling mechanism (as described in Section 4.2.4.2 on Page 123) would be triggered if the current *gbest* fitness is better than the previous *gbest* fitness for at least 1%; otherwise, PPSO also stops.

4.3.4 Experiment Configuration

Since these datasets have small sample sizes, 10-fold cross-validation (CV) is used to generate training and test sets. In each cross-validation, one fold is used to form the test set and the remaining nine folds are used to form the training set. The test set is used for performance evaluation of the discretisation and feature selection solution produced by each method based on the training set. During the evolutionary process, an inner loop of 10-fold CV on the training set is used for fitness evaluation. Therefore, each method comprises of two loops of CV as recommended in [120] to avoid *feature selection bias*.

To eliminate the statistical variations, each method was run 30 times with different random seeds for each dataset. Because each dataset was split into training and test set using 10-fold CV, a total of 300 runs were executed for each method on each dataset. Experiments were run on PC with Intel Core i7-4770 CPU @ 3.4GHz and 8GB memory. The results of 30 runs from each method were compared using the statistical Wilcoxon significance test with a 5% significance level.

4.4 Results and Discussions

Table 4.2 shows the results of PSO-FS, EPSO, and PPSO. The average feature subset size returned by each method over the 30 runs is shown in column "Size". The best, mean and standard deviation of KNN accuracies using "Full" (i.e. all continuous features), or using the transformed data by each of the

compared methods are shown in columns four and five, respectively. The test accuracies calculated based on Equation (4.2) are reported.

Column T_E and T_P display the statistical Wilcoxon significance test results of the method in the corresponding row over EPSO and PPSO, respectively. In T_E, "+" or "–" means the result is significantly better or worse than EPSO. "=" means they have similar performance. In other words, the more "–", the better EPSO. The same meaning of these symbols is used for T_P.

4.4.1 Results of EPSO

4.4.1.1 EPSO versus Full

According to Table 4.2, the average number of features selected by EPSO was less than 1% of the original number of features in Prostate and DLBCL, 1% to 3% in other seven datasets and 6% in SRBCT. Using the discretised and selected features by EPSO, KNN achieved significantly better performance than using Full on seven out of the ten datasets. An increase of more than 7% in average accuracy was achieved on five out of the ten datasets and the highest improvement was 22% on 9Tumor.

EPSO obtained a similar accuracy as using original features on Brain1 and Leukemia2 and 2% lower accuracy on Prostate with much smaller feature subsets (on average 54.9 features from 10,509 features). However, the best accuracies of EPSO on these three datasets were still 7%, 5% and 5% higher than using all features, respectively.

The results indicated that EPSO could simultaneously discretise and select relevant features so that the discriminating power of the feature set was either significantly improved or maintained with a much smaller number of features.

4.4.1.2 EPSO versus PSO-FS

As can be observed in Table 4.2 that EPSO selected a much smaller number of features than PSO-FS. With the transformed features, EPSO outperformed PSO-FS on six datasets with the highest improvement of about 12% in

Table 4.2: Experimental results

Dataset	**Method**	**Size**	**Best**	**Mean ± Std**	T_E	T_P
SRBCT	Full	2308.0	87.08		–	–
	PSO-FS	150.0	97.50	91.31 ± 2.71	–	–
	EPSO	137.3	100.00	96.89 ± 1.64		+
	PPSO	108.5	100.00	95.78 ± 1.96		
DLBCL	Full	5469.0	83.00		–	–
	PSO-FS	101.8	96.67	80.03 ± 6.13	–	–
	EPSO	42.8	94.17	85.18 ± 5.46		=
	PPSO	44.0	94.17	86.22 ± 3.58		
9Tumor	Full	5726.0	36.67		–	–
	PSO-FS	955.0	55.00	45.95 ± 4.93	–	–
	EPSO	138.5	65.00	58.22 ± 3.12		=
	PPSO	118.1	65.00	59.28 ± 2.08		
Leukemia1	Full	5327.0	79.72		–	–
	PSO-FS	150.0	92.22	81.60 ± 4.72	–	–
	EPSO	135.9	95.56	93.37 ± 1.83		–
	PPSO	80.4	95.42	94.37 ± 1.36		
Brain1	Full	5920.0	72.08		=	–
	PSO-FS	317.3	78.75	71.00 ± 3.06	–	–
	EPSO	150.7	79.17	72.79 ± 3.48		=
	PPSO	73.4	82.08	74.40 ± 3.67		
Leukemia2	Full	11225.0	89.44		=	–
	PSO-FS	150.0	93.89	86.11 ± 3.97	–	–
	EPSO	139.9	94.44	89.93 ± 2.79		–
	PPSO	86.7	100.00	96.74 ± 1.64		
Brain2	Full	10367.0	62.50		–	–
	PSO-FS	417.9	82.08	69.11 ± 5.89	=	=
	EPSO	152.8	83.75	70.76 ± 5.30		=
	PPSO	66.7	74.58	68.75 ± 4.24		
Prostate	Full	10509.0	85.33		+	–
	PSO-FS	777.4	90.33	85.20 ± 2.35	+	–
	EPSO	54.9	90.33	83.74 ± 3.55		–
	PPSO	65.6	95.17	91.82 ± 1.77		
11Tumor	Full	12533.0	71.42		–	–
	PSO-FS	1638.8	86.07	82.62 ± 1.70	+	+
	EPSO	149.9	83.68	79.29 ± 2.11		+
	PPSO	167.0	83.20	76.83 ± 2.91		
Lung	Full	12600.0	78.05		–	–
	PSO-FS	686.2	85.73	81.72 ± 2.08	=	+
	EPSO	150.8	85.58	80.60 ± 2.42		=
	PPSO	203.0	84.11	79.38 ± 3.26		

average accuracy on 9Tumor and Leukemia1. While PSO-FS degraded the performance of KNN over using Full on DLBCL, EPSO still attained a better performance than using the original feature set on this dataset with only half the number of features selected by PSO-FS.

The biggest difference between these two methods can be seen on the 9Tumor dataset. While PSO-FS selected 955 features to achieve an improvement of 9% in classification accuracy, EPSO improved 22% accuracy with only 139 features. A similar pattern can be seen from the first six datasets.

On Brain2, Prostate and Lung, both methods obtained a similar accuracy. However, EPSO feature subset was smaller than a half of the size of feature subset selected by PSO-FS. On Prostate and Lung, EPSO selected only 54 and 150 features on average compared to 777 and 686 features selected by PSO-FS. In 11Tumor, PSO-DFS attained 3% lower accuracy than PSO-FS with only 149 features while PSO-FS selected more than 1,600 features.

In general, out of the 20 comparisons between EPSO versus Full and PSO-FS, EPSO won 13 cases, drew 4 and lost 3. The results showed that EPSO effectively performed feature selection via discretisation to create a more compact and better discriminating representation for data than the original dataset. The EPSO results also suggested that the proposed approach of combining data discretisation and feature selection in one stage performed better than separating the two steps into different stages.

4.4.2 Results of PPSO

4.4.2.1 PPSO versus Full

As can be seen from Table 4.2 that PPSO selected less than 1% of the total number of features on four datasets, less than 2% on five datasets and 4.6% on SRBCT. In general, PPSO selected the smallest subsets among the compared methods on six datasets. With very compact solutions, PPSO achieved significantly better classification performance than using Full on all ten datasets. It increased more than 5% accuracy on seven datasets with the

highest improvement of 23% on 9Tumor.

The results indicated that PPSO could produce a much more powerful and compact representation for high-dimensional datasets.

4.4.2.2 PPSO versus PSO-FS

In terms of the dimensionality reduction, feature subsets returned by PPSO were much smaller than those selected by PSO-FS, which is the two-stage approach. In terms of the classification accuracy, the PPSO transformed data outperformed PSO-FS on seven datasets. While PSO-FS degraded the classification performance on Brain1 and Prostate, PPSO still obtained better accuracies than using "Full" on these datasets with about 11% and 8% of feature subset size selected by PSO-FS on these datasets respectively. A similar pattern was observed on DLBCL and Leukemia2.

Similar to EPSO, the results on the 9Tumor dataset revealed the largest difference between two approaches. While the transformed dataset by PSO-FS improved 9% classification accuracy with 955 features, the transformed dataset by PPSO achieved 23% improvement using eight times fewer features than PSO-FS (118 features). The first six datasets witnessed a similar pattern.

On Brain2, PPSO obtained a similar accuracy to PSO-FS with just a quarter of features selected by PSO-FS. On Lung and 11Tumor, PPSO obtained worse results than PSO-FS with about one third and one-tenth number of features selected by PSO-FS, respectively.

In general, solutions evolved by PPSO obtained either a significantly better or similar classification accuracy to PSO-FS on eight out of the ten datasets.

4.4.3 PPSO versus EPSO

As can be seen from Table 4.2 that PPSO selected a smaller number of features than EPSO on six datasets and obtained better or similar classification accuracy to EPSO on eight datasets. On Leukemia2, while EPSO obtained a

similar classification accuracy to "Full" with 140 features, PPSO increased 7% accuracy with only 87 features.

The improvement of PPSO over EPSO may result from several proposed strategies in PPSO. First of all, PPSO only finds an appropriate cut-point for each feature from the potentially good cut-points while EPSO chooses any possible cut-point within the range of the feature values. Therefore, PPSO has a much smaller search space than EPSO. This enables PPSO to better cover the search space when using the same parameter settings as EPSO. Furthermore, the evolutionary process of PPSO is not only guided by the accuracy of the transformed dataset but also the distance measure. This measure helps PPSO to differentiate feature subsets with the same training accuracy. Finally, the scaling mechanism enables PPSO to move the evolutionary process to larger feature subsets that may provide better solutions.

4.4.4 Computation Time

Figure 4.5 compares the running time to complete a single run of PSO-FS, EPSO, and PPSO. The reported time (in minutes) is the average of the 30 runs. Note that the three compared methods used the same wrapper approach with KNN for fitness evaluation. We also note that to evaluate each particle, PSO-FS only needs to transform data based on the selected features, while EPSO and PPSO need to discretise and select features at the same time. This means that the EPSO and PPSO evaluation procedures need an additional overhead compared to PSO-FS. Therefore, EPSO and PPSO were expected to run longer than PSO-FS. However, an opposite trend is reported in Figure 4.5.

Compared with PSO-FS, EPSO had a shorter running time on eight datasets although both have the same stopping criterion, which is either the maximum of 70 iterations is reached or *gbest* is not improved for 10 iterations. In addition, the initial particles in both methods have the same number of selected features. Similarly, PPSO's running time was also shorter than

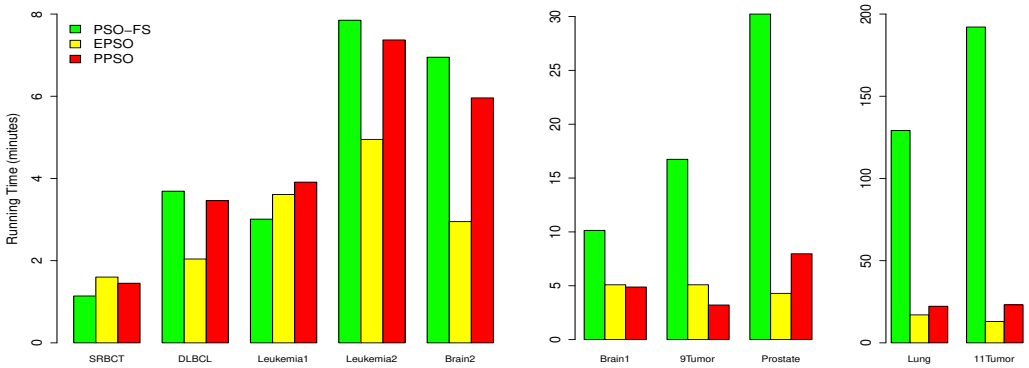


Figure 4.5: Running Time of PPSO.

PSO-FS on eight datasets with about 8 times shorter on the 11Tumor dataset. As can be seen from Figure 4.5 that the larger the dataset, the greater the ratio of PSO-FS's running time to PPSO's running time.

Compared with EPSO, PPSO took slightly longer time on most datasets due to the scaling mechanism, which introduces large feature subsets. Note that although the fitness function of PPSO combines both wrapper and filter measures, the time spent on fitness evaluation was not longer than EPSO. This is because both KNN and the filter measure are based on the distance which only needs to be calculated once and used for both measures.

The main contribution of this improvement in EPSO and PPSO computation time can be explained by the average number of iterations run by each method on each dataset which is presented in Table 4.3. The results showed that PSO-FS and EPSO evolved solutions using similar numbers of iterations. On the other hand, due to the scaling mechanism, the total number of iterations PPSO used were 2 to 7 times larger than PSO-FS and EPSO on all datasets. However, despite of this longer evolutionary process, PPSO is still more efficient. This indicates that the difference in computation time was mainly because the numbers of selected features in PSO-FS were much larger than those of EPSO and PPSO (as shown in Table 4.2), which in turn leads to a much longer running time required by KNN in the fitness evaluation process.

Table 4.3: The average number of iterations until the stopping criterion is met

Dataset	PSO-FS	EPSO	PPSO
SRBCT	10.87	13.13	73.60
DLBCL	11.63	15.43	70.00
9Tumor	26.80	34.23	70.00
Leukemia1	10.87	13.27	70.33
Brain1	12.63	12.70	70.00
Leukemia2	11.00	13.40	70.33
Brain2	13.77	12.73	70.70
Prostate	16.97	15.87	70.00
11Tumor	25.23	11.27	85.27
Lung	14.70	11.00	56.87

4.4.5 PPSO versus Traditional Methods

Table 4.4 shows the compared results of PPSO with "Full", MDL+LFS, MDL+CON, MDL+CFS and MChi2. Besides the size and time, which are shown in the third and the last columns, the best and the mean of the training and test accuracies are shown under each corresponding column. The "S_{Tr}" and "S_{Te}" display the Wilcoxon significance test results of the corresponding method over PPSO in terms of training and test, respectively. The result of MChi2 on the Lung dataset is unavailable due to the out of memory error. Since the CFS method is quite expensive, it failed to finish its run in 12 hours for the two largest datasets, which are Lung and 11Tumor with more than 12,000 features.

As can be seen from Table 4.4 that PPSO outperformed MDL+LFS and MDL+CON on all the ten datasets in terms of training accuracy and on nine datasets in terms of test accuracy. Compared with MDL+LFS, PPSO achieved 6% to 19% higher test accuracy on all datasets except for the Lung dataset where both obtain similar accuracy. However, the best test accuracy achieved by PPSO was 4% higher than MDL+LFS on this dataset. Similarly, PPSO achieved 5% to 31% higher test accuracy than MDL+CON on all datasets except for DLBCL where MDL+CON achieved the highest

Table 4.4: Comparison with traditional methods

Dataset	Method	Size	Training			Test			Time(s)
			Best	Mean	S_{Tr}	Best	Mean	S_{Te}	
SRBCT	Full	2,308.0		83.35	–		87.08	–	
	MDL+LFS	6.1		98.19	–		88.75	–	28.78
	MDL+CON	4.3		97.71	–		85.83	–	3.24
	MDL+CFS	80.9		100.00	=		100.00	+	318.04
	MChi2	85.9		100.00	=		100.00	+	83.07
	PPSO	108.5	100.00	100.00		100.00	95.78		87.14
DLBCL	Full	5,469.0		81.71	–		83.00	–	
	MDL+LFS	4.0		98.24	–		74.00	–	73.78
	MDL+CON	3.4		98.14	–		92.50	+	6.29
	MDL+CFS	58.0		99.22	–		91.67	+	1310.49
	MChi2	10.2		87.96	–		75.50	–	369.45
	PPSO	44.0	100.00	100.00		94.17	86.22		207.40
9Tumor	Full	5,726.0		33.44	–		36.67	–	
	MDL+LFS	12.6		82.39	–		41.67	–	64.20
	MDL+CON	7.6		71.61	–		28.33	–	12.65
	MDL+CFS	38.0		90.71	–		53.33	–	656.21
	MChi2	58.5		77.28	–		48.33	–	260.70
	PPSO	118.1	92.87	92.31		65.00	59.28		192.68
Leukemia1	Full	5,327.0		79.77	–		79.72	–	
	MDL+LFS	4.8		99.17	–		81.39	–	70.48
	MDL+CON	3.0		98.67	–		89.17	–	5.69
	MDL+CFS	56.0		100.00	=		93.19	–	1358.35
	MChi2	46.4		98.80	–		92.08	–	276.35
	PPSO	80.4	100.00	100.00		95.42	94.37		236.56
Brain1	Full	5,920.0		65.07	–		72.08	–	
	MDL+LFS	9.9		89.13	–		59.17	–	104.41
	MDL+CON	6.2		79.99	–		55.42	–	14.51
	MDL+CFS	115.4		99.93	–		79.58	+	2859.76
	MChi2	290.0		80.63	–		74.58	=	438.82
	PPSO	73.4	100.00	99.99		82.08	74.40		292.84
Leukemia2	Full	11,225.0		88.82	–		89.44	–	
	MDL+LFS	4.3		99.08	–		90.00	–	231.15
	MDL+CON	3.0		99.44	–		85.56	–	17.54
	MDL+CFS	79.0		100.00	=		98.89	+	5407.43
	MChi2	166.1		99.56	–		93.33	–	904.96
	PPSO	86.7	100.00	100.00		100.00	96.74		441.95
Brain2	Full	10,367.0		63.52	–		62.50	–	
	MDL+LFS	5.6		98.80	–		53.33	–	179.83
	MDL+CON	4.7		89.92	–		61.67	–	19.33
	MDL+CFS	63.4		100.00	=		71.25	+	3304.74
	MChi2	160.7		84.00	–		70.00	=	562.42
	PPSO	66.7	100.00	99.99		74.58	68.75		357.34
Prostate	Full	10,509.0		82.08	–		85.33	–	
	MDL+LFS	4.9		82.44	–		73.17	–	227.68
	MDL+CON	4.7		98.46	–		70.50	–	23.61
	MDL+CFS	51.6		98.12	–		90.17	–	2694.27
	MChi2	33.6		95.44	–		86.17	–	1246.46
	PPSO	65.6	100.00	99.84		95.17	91.82		478.17
11Tumor	Full	12,533.0		71.01	–		71.42	–	
	MDL+LFS	14.3		79.96	–		61.71	–	555.99
	MDL+CON	9.4		72.58	–		53.83	–	85.41
	MDL+CFS	NA		NA			NA		NA
	MChi2	2098.0		93.32	–		84.54	+	3125.14
	PPSO	167.0	99.47	99.14		83.20	76.83		1387.01
Lung	Full	12,600.0		71.59	–		78.05	–	
	MDL+LFS	12.2		95.12	–		80.55	=	685.78
	MDL+CON	6.6		89.54	–		74.64	–	63.37
	MDL+CFS	NA		NA			NA		NA
	MChi2	NA		NA			NA		NA
	PPSO	203.0	97.67	97.09		84.11	79.38		1335.21

average test result. Although these two methods obtained the smallest feature subsets, their test accuracies were the lowest among the compared methods. Compared with the Full accuracy, MDL+LFS even had 13% lower on Brain1 and MDL+CON had 18% lower on 11Tumor. In contrast, PPSO achieved better accuracy than the full feature sets on all datasets.

Compared with MDL+CFS, PPSO obtained better or similar training accuracies on the eight datasets. For testing, PPSO achieved better results than MDL+CFS on three datasets, namely 9Tumor, Leukemia1 and Prostate. For the other five datasets, MDL+CFS had better results than PPSO on average but the best accuracy of PPSO was almost always better than MDL+CFS. As can be seen in the last column of Table 4.4 that the computation time of MDL+CFS quickly increased along with the number of features. Therefore, scalability is a major concern when applying this method to high-dimensional data.

Similarly, in terms of training accuracy, PPSO also outperformed MChi2 on all datasets. In terms of test accuracy, PPSO performed either similar to or significantly better than MChi2 on seven datasets and selecting smaller feature subsets on four datasets. We also noted that although MChi2 is a filter method, its running time was longer than that of PPSO and MDL+LFS on almost all datasets. While the difference in running time between MChi2 and PPSO was not too big on small datasets, it became much larger on datasets with more than ten thousand features. This indicates that the proposed algorithm is scalable to high-dimensional datasets.

4.5 Further Analysis

In addition to classification performance and feature subset size, generalisation and robustness are also important criteria in evaluating the performance of a feature selection method. While generalisability relates to the ability of the learned model to have a similar performance on both training and test data, robustness relates to the ability to reproduce the results regardless of

Table 4.5: Average training accuracy.

Dataset	Method	Train-Acc(Std)	T	Dataset	Method	Train-Acc(Std)	T
SRBCT	PSO-FS	100.00 (0.00)	=	Leukemia2	PSO-FS	100.00 (0.00)	=
	EPSO	100.00 (0.00)	=		EPSO	100.00 (0.00)	=
	PPSO	100.00 (0.00)			PPSO	100.00 (0.00)	
DLBCL	PSO-FS	100.00 (0.00)	=	Brain2	PSO-FS	99.73 (0.12)	–
	EPSO	100.00 (0.00)	=		EPSO	98.38 (0.27)	–
	PPSO	100.00 (0.00)			PPSO	99.99 (0.05)	
9Tumor	PSO-FS	97.49 (0.23)	+	Prostate	PSO-FS	98.89 (0.10)	–
	EPSO	95.03 (0.22)	+		EPSO	98.56 (0.14)	–
	PPSO	92.31 (0.33)			PPSO	99.84 (0.05)	
Leukemia1	PSO-FS	100.00 (0.00)	=	11Tumor	PSO-FS	99.80 (0.08)	+
	EPSO	100.00 (0.00)	=		EPSO	96.21 (0.19)	–
	PPSO	100.00 (0.00)			PPSO	99.14 (0.20)	
Brain1	PSO-FS	100.00 (0.00)	=	Lung	PSO-FS	97.77 (0.05)	+
	EPSO	99.33 (0.29)	–		EPSO	97.10 (0.14)	=
	PPSO	99.99 (0.06)			PPSO	97.09 (0.30)	

the small variance of the experiments. In this section, the performance of the one-stage (EPSO and PPSO) is compared with the two-stage (PSO-FS) approaches in terms of these two aspects.

4.5.1 Generalisation

The generalisability of the one-stage (EPSO and PPSO) and two-stage (PSO-FS) approaches are compared based on the training accuracy of the returned solutions. Table 4.5 shows the mean and standard deviation of the training accuracy obtained by the three methods in 30 runs of each dataset.

As can be seen from Table 4.5, PPSO achieved better training accuracy than PSO-FS on two datasets and similar on five. However, as can be seen from Table 4.2, PPSO achieved significantly better test accuracy on seven datasets than PSO-FS and similar on one. This indicated that PPSO had better generalisability than PSO-FS.

Compared with EPSO, PPSO obtained the same training accuracy (100%)

on SRBCT with a smaller number of features and 3% higher accuracy on 11Tumor with a bigger feature subset. This indicated that the scaling mechanism helped PPSO effectively select appropriate feature subsets that can maintain or increase the training accuracy. However, these solutions seemed to overfit the training data when they got 1% and 3% lower test accuracy than those of EPSO, respectively as shown in Table 4.2. Another possible reason is that PPSO used the fix potential cut-points calculated from the training data that may not be representative of the test data. A better mechanism to calculate the potential cut-points may help PSO improve the performance of PPSO.

We also noted a big difference between the test and training accuracies (shown in Table 4.2 and 4.5), especially on 9Tumor for all the three methods. This indicated the existence of overfitting, which may be because the skew distribution of features in these datasets creates different distributions on training and test sets. Therefore, the learned model may not be generalised well to the test data. This problem is worse on datasets with smaller numbers of samples, such as Brain2 and 9Tumor with 50 and 60 instances. On top of this, the large number of classes and the class imbalance nature are additional challenges of these datasets. With 9 classes, 9Tumor had worse results than Brain2 which has 4 classes. As can be seen from Table 4.4, MDL+LFS and MChi2 performances were also affected by this phenomenon. The gaps between training and test accuracy obtained by MDL+LFS on 9Tumor and Brain2 are about 40%, which is even larger than PPSO with about 30%.

4.5.2 Robustness

Since each method was run 30 times on each 10-fold CV, it produced 300 solutions for each dataset. If the dataset has redundant features, different solutions may include very different features if the method is sensitive to the experimental conditions, leading to a low selection frequency of each feature. As described in Section 3.5.2 on Page 104, Z-score [271, 111] is a measure indicating the significance of the selection frequency of a feature, which shows

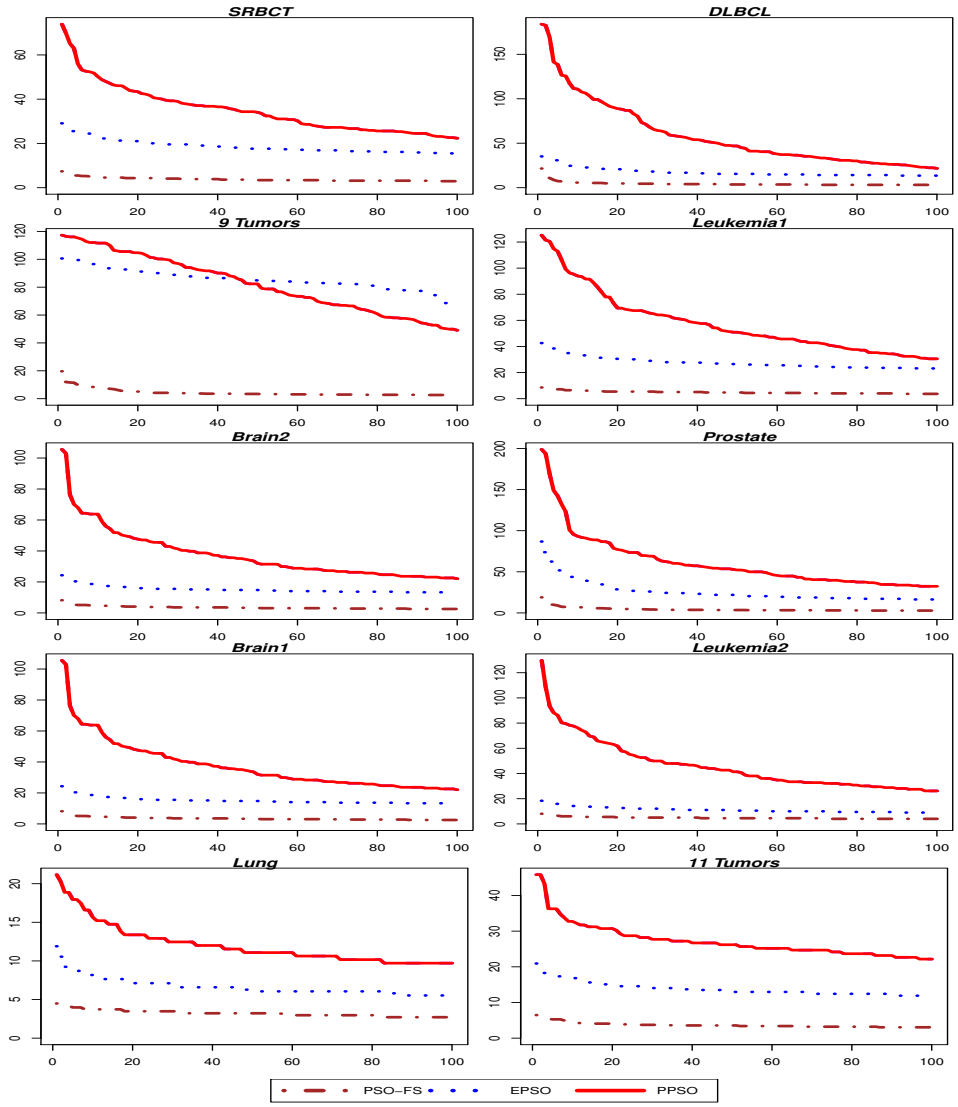


Figure 4.6: Z-score of the top 100 selected features on each dataset.

the robustness of a method. Figure 4.6 shows the Z-score calculated based on Equation 3.8 of 100 most selected features of each method.

As can be observed from the chart, the Z-scores of the features selected by PPSO are greater than those of EPSO, which are in turn greater than PSO-FS. This indicates that PPSO is more robust than the other methods.

In general, the results show that PPSO had better generalisability and

robustness than PSO-FS. Solutions produced by PPSO were more compact and had higher discriminating power than PSO-FS. This demonstrated the hypothesis that important information including feature interaction may be lost during the discretisation process in the first stage of PSO-FS. Since the proposed one-stage approach evaluates the selected features with their cut-points simultaneously, it takes this important information into account.

4.5.3 Comparison with PSO-LS

To see if the multi-variate discretisation and feature selection proposed in this chapter obtained better results than the previously proposed methods in Chapter 3, this section will compare the results of PPSO with PSO-RLS and PSO-CLS. For reading convenience, the results of these methods were shown again in Table 4.6. The last column T shows the Wilcoxon significance test results with a 5% significance level comparing the corresponding method against PPSO.

As can be seen from Table 4.6, PPSO selected a much smaller number of features than PSO-RLS on all datasets and PSO-CLS on six datasets. Its feature subsets performed similar to or significantly better than PSO-RLS on 6 datasets and PSO-CLS on 4 datasets. Using the PPSO generated features on 9Tumor, KNN obtained 8% and 11% higher accuracy than using PSO-RLS and PSO-CLS features, respectively. On Prostate, PPSO selected about 65 features that obtained 5% higher accuracy than 2600 features selected by the other two methods. Note that PSO-RLS and PSO-CLS discretised data before applying feature selection using the MDL method [70], which can discretise feature into multiple discrete values. On the other hand, PPSO is a binary discretisation which can only divide features into two discrete values only. This difference may mainly account for the inferior performance of PPSO on the other datasets, where more discrete values may be needed to better represent the feature space.

Table 4.6: Comparison between PPSO and PSO-LS methods

Dataset	**Method**	**Size**	**Best**	**Mean ± Std**	T
SRBCT	PSO-RLS	545.07	99.17	96.08 (1.73)	=
	PSO-CLS	59.69	100.00	99.97 (0.15)	+
	PPSO	108.47	100.00	95.78 (1.96)	
DLBCL	PSO-RLS	1417.44	97.33	93.72 (1.79)	+
	PSO-CLS	47.43	96.67	90.86 (3.19)	+
	PPSO	44.01	94.17	86.22 (3.58)	
9Tumor	PSO-RLS	1352.04	58.33	48.39 (4.88)	–
	PSO-CLS	46.68	60.00	51.39 (4.22)	–
	PPSO	118.12	65.00	59.28 (2.08)	
Leuk1	PSO-RLS	1534.91	95.56	93.45 (1.71)	–
	PSO-CLS	31.92	95.42	94.84 (1.16)	=
	PPSO	80.39	95.42	94.37 (1.36)	
Brain1	PSO-RLS	1549.01	77.50	75.00 (1.80)	=
	PSO-CLS	1081.48	82.50	76.78 (2.09)	+
	PPSO	73.37	82.08	74.40 (3.67)	
Leuk2	PSO-RLS	3426.47	93.89	91.72 (1.46)	–
	PSO-CLS	53.68	98.33	95.56 (1.68)	–
	PPSO	86.65	100.00	96.74 (1.64)	
Brain2	PSO-RLS	3099.04	82.50	75.35 (3.16)	+
	PSO-CLS	2647.70	78.75	73.47 (2.82)	+
	PPSO	66.74	74.58	68.75 (4.24)	
Prostate	PSO-RLS	2690.31	89.17	85.79 (1.49)	–
	PSO-CLS	2670.34	91.17	86.98 (1.76)	–
	PPSO	65.55	95.17	91.82 (1.77)	
11Tumor	PSO-RLS	3163.91	87.77	84.19 (1.47)	+
	PSO-CLS	266.84	90.72	87.51 (1.73)	+
	PPSO	166.96	83.20	76.83 (2.91)	
Lung	PSO-RLS	3453.94	86.87	83.50 (1.16)	+
	PSO-CLS	311.55	96.43	90.78 (2.61)	+
	PPSO	202.95	84.11	79.38 (3.26)	

4.6 Chapter Summary

The goal of this chapter was to propose an integrating approach to multivariate discretisation and feature selection in a single stage using BBPSO. The goal has been achieved by proposing two new PSO based methods, EPSO and PPSO, with two new PSO representations for discretising and selecting features simultaneously. While EPSO aims to directly *evolve* a cut-point for each feature, PPSO aims to choose one from all the *potentially* good cut-points. EPSO and PPSO were compared with using the full set of original features, and the two-stage approach (PSO-FS).

Experimental results on ten datasets having thousands to tens of thousands of features with various numbers of classes showed that the proposed methods were able to discretise and select features simultaneously. A much smaller number of relevant features were selected with better discriminating ability than the two-stage approach. The comparison results suggested that it is more effective to combine discretisation and feature selection in a single stage. Compared with EPSO, PPSO obtained either equivalent or better results with smaller numbers of features. Further analysis also showed that PPSO was more general and more robust than the compared PSO methods.

PPSO was also compared with four traditional methods as representatives of the two-stage and one-stage approaches, MDL+LFS, MDL+CON, MDL+CFS and MChi2. The results of two experiments with and without *feature selection bias* showed that PPSO had significantly better performance than MDL+LFS, MDL+CON and MChi2 and similar performance to MDL+CFS in most cases. The results of PPSO also showed that it is more scalable than MDL+CFS and MChi2 in dealing with high-dimensional problems. Comparing results on both KNN and NB indicated that solutions obtained by PPSO could be generalised to other classification algorithms than the one used during the training process.

In Chapters 3 and 4, feature selection has shown to be effective in improving classification performance of learning algorithms on high-dimensional data.

However, its performance may still be limited if the original features do not provide enough discriminative information for learning algorithms to learn a good classifier. There may exist certain combinations of features that may provide better discriminating ability [170]. In this case, feature construction can be used to construct new high-level features that better represent the problem. We will investigate the techniques of feature construction in the next two chapters.

5

Cluster-Based Single Feature Construction Using GP

5.1 Introduction

The previous chapters have shown that feature selection is an important step to improve classification performance of learning algorithms, especially for high-dimensional data due to the curse of dimensionality. Apart from feature selection, feature construction can also be used for dimensionality reduction by constructing a much smaller number of new high-level features from the original ones [9].

Genetic programming (GP) has shown promise in feature construction [9, 174]. By applying the Darwinian principle *"Survival of the fittest"*, GP can choose good features and operators from the given feature set and function set, respectively, to create better discriminating features. Although GP has shown its capability in feature selection and construction, its performance on high-dimensional data is still limited due to the large search space. Compared

with feature selection on the same problem, feature construction has much larger search space, because feature construction needs to select not only good features but also appropriate operators and ways to combine them. Therefore, it is critical to narrow the search space to improve GP performance on high-dimensional data. Results of GP based methods for feature selection [7] and classification [168] have shown that helping GP select appropriate features is critical in enhancing its performance. However, applying this approach to GP for feature construction is still limited, especially for high-dimensional problems. Furthermore, feature clustering, which was introduced in Section 2.2.7 on Page 41, has been proposed and shown promise in feature selection [81, 89, 109, 127, 200]. Different from the common data mining task of clustering that groups similar *instances* into clusters [36], feature clustering groups similar *features* into one cluster. Based on the resulting clusters, one or several features from each group can be chosen as representatives to form the final feature subset. This approach effectively reduces the search space for feature selection. However, feature clustering has not been investigated in feature construction for classification.

5.1.1 Chapter Goals

The goal of this chapter is to develop a clustering and GP based feature construction method (CGPFC) for classification in high-dimensional data. The scope of the proposed algorithm is to construct a single feature for binary classification problems. The new algorithm is expected to select appropriate features to construct a single new feature that can improve the classification performance of the common classification algorithms on these problems. To achieve this goal, a new feature clustering technique is proposed to narrow the GP search space. Specifically, this chapter will investigate:

1. How to use feature clustering to automatically and effectively reduce the GP search space;

2. Whether CGPFC can construct features with better discriminating

ability (better classification accuracy) than the original full feature set and the one constructed by standard GP;

3. Whether CGPFC can select a smaller number of features than standard GP to construct a new feature;

4. Whether the sets of features selected by CGPFC have better discriminating ability than those of standard GP; and

5. Which combination set of selected and/or constructed features from a single GP tree helps learning algorithms obtain the best performance.

5.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. Section 5.2 describes the proposed algorithm CGPFC. Section 5.3 discusses the datasets, parameter settings and experiment configuration used to test the performance of CGPFC. The results of CGPFC are shown and discussed in Section 5.4. Further analysis is followed in Section 5.5. Finally, a summary of this chapter is given in Section 5.6.

5.2 The Proposed Algorithms

High-dimensional data may contain a significant number of irrelevant and redundant features that should be eliminated to reduce the GP search space. Therefore, this new method aims at using feature clustering to group similar features so that only representative features from these groups are fed into GP for feature construction.

5.2.1 Overall Structure

Figure 5.1 shows the overall system of the proposed method called CGPFC. Firstly, all features are grouped into clusters using a proposed redundancy

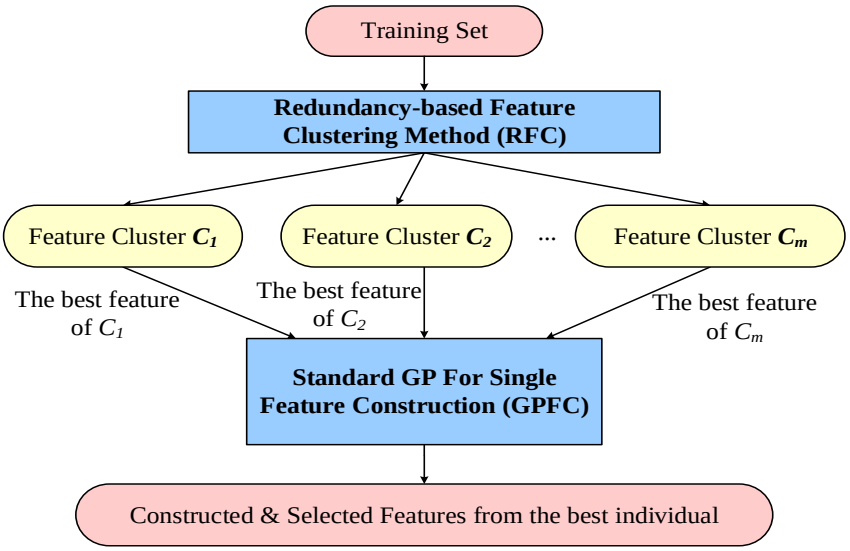


Figure 5.1: The CGPFC overall system.

based feature clustering algorithm (RFC), which will be described in the following section. Each cluster includes similar or redundant features. Then, the best feature of each cluster is put into the terminal set for GP (GPFC) to construct a new feature. Therefore, CGPFC can be seen as a combination of RFC and GPFC. Note that while RFC uses a filter measure to group features, GPFC follows the embedded approach. More details are explained in the following sections.

5.2.2 Redundancy Based Feature Clustering: RFC

As shown in Figure 5.1, the first component of the proposed method is the redundancy based feature clustering method or RFC. The aim of RFC is to group similar or redundant features into the same cluster. Among many clustering techniques, K-means is a well-known, simple and effective algorithm. It has also been used to cluster features for feature selection method [99]. However, it is essential to predefine a suitable number of clusters, which is not easy, especially in high-dimensional data with thousands to tens of

thousands of features. An inappropriate value may lead to clusters with uncorrelated or non-redundant features. In addition, the number of clusters in feature clustering is not as meaningful as the number of clusters in instance clustering, which represents the number of different types of objects/instances. Therefore, instead of grouping features based on a predefined number of clusters, the proposed feature clustering method will group features based on the redundancy levels between them (so-called RFC). Different from the number of clusters, the level of redundancy or correlation between two features is a value in the range of 0 and 1, representing no and full correlation between them, respectively.

RFC uses a simple approach to ensure that all features in the same cluster are redundant features with their correlation level higher than a predefined threshold. If two features X and Y have their CC(X,Y) larger than this threshold, they will be grouped into the same cluster. In this way, the number of clusters will be automatically determined. If a dataset has a large number of redundant features, the number of clusters will be much smaller than the number of features, and vice versa. Furthermore, this strategy ensures that the generated clusters include only features having their correlation levels higher than the predefined redundancy threshold.

Algorithm 5 shows the pseudo code of RFC with a redundancy threshold θ. First of all, features are analysed to remove irrelevant ones (lines 4-9). In this study, a feature is considered irrelevant if it does not give any information about the class label. Since the class label is a discrete variable, Symmetrical Uncertainty (SU), which is a normalised version of information gain, is a suitable measure for feature relevancy. Therefore, in this step, all features whose SU with the class label are equal to zero will be removed.

SU between a feature X and the class C is calculated based on Equation (5.1) which gives a value between 0 and 1 representing no to full correlation, respectively. As SU is an entropy-based measure, it can only be applied to category or nominal data. Therefore, data is discretised before calculating SU using the popular discretisation method, the minimum description length

Algorithm 5: The pseudo code of RFC

Input : Training data, redundancy_threshold θ
Output : Clusters of features

begin
  $F \leftarrow \emptyset$;
  for *each feature* f_i *in Training data* **do**
    $su \leftarrow SU(f_i, class)$ (based on Equation (5.1)) ;
    if $su > 0$ **then**
      $F \leftarrow F \cup \{f_i\}$;
    end
  end
  Sort F based on descending order of su;
  $clusters \leftarrow \emptyset$;
  while $(F \neq \emptyset)$ **do**
    $f_i \leftarrow$ next feature in F;
    $F \leftarrow F \setminus \{f_i\}$;
    $new\_cluster \leftarrow \{f_i\}$;
    while $(F \neq \emptyset)$ **do**
      $f_j \leftarrow$ next feature in F;
      $cc \leftarrow CC(f_i, f_j)$ (based on Equation (5.3)) ;
      if $(cc > \theta)$ **then**
        $new\_cluster \leftarrow new\_cluster \cup \{f_j\}$;
        $F \leftarrow F \setminus \{f_j\}$;
      end
    end
    $clusters \leftarrow clusters \cup new\_cluster$;
  end
  Return $clusters$;
end

(MDL) [70], whose principle is explained in Section 2.1.4.

$$SU(X, C) = 2\left[\frac{IG(X|C)}{H(X) + H(C)}\right] \tag{5.1}$$

where

$$IG(X|C) = H(X) - H(X|C) \tag{5.2}$$

and $H(X)$ is the entropy of X and $H(X|C)$ is the conditional entropy of X given C.

All the remaining features are then sorted based on their SU values so that the most relevant feature will be the first one in the list. The outer while loop in Line 11 picks the next feature f in the list to form the first cluster, while the inner while loop scans the remaining features in the list to add all features that are correlated with f. In this step, Correlation Coefficient (CC) is used to measure the redundancy level between features because it can be directly applied to numerical data. Although CC can only measure the linear relationship between variables, it has been shown to be effective in many feature selection methods [99, 246]. The CC measure gives a value between -1 and 1 whose absolute value represents the correlation level between two features. Given that M is the number of instances in the dataset, CC between features X and Y is calculated based on Equation (5.3), which gives a value between 0 and 1.

$$CC(X,Y) = \left| \frac{\sum_{i=1}^{M} X_i Y_i - n\bar{X}\bar{Y}}{\sqrt{\sum_{i=1}^{M} X_i^2 - n\bar{X}^2}\sqrt{\sum_{i=1}^{M} Y_i^2 - n\bar{Y}^2}} \right| \tag{5.3}$$

If the CC value of two features is larger than a threshold, they are said to be redundant and grouped in the same cluster. When a feature is added to a cluster, it is removed from the list. Therefore, all features are grouped into exclusive or non-overlapped clusters and the number of clusters is automatically determined based on the given redundancy threshold.

Finally, at the end of the outer while loop when all features are clustered, RFC returns all the created clusters whose best features will be used to form the terminal set which is fed into GP for feature construction as shown in Figure 5.1.

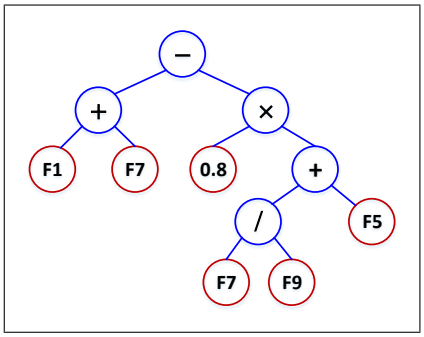


Figure 5.2: GP representation of a constructed feature.

5.2.3 Representation

CGPFC aims at constructing a single feature using a tree-based representation. Each GP individual has one tree (so-called single-tree GP) which represents a constructed high-level feature because it can generate a new value from the original feature values.

The main focus of this chapter is single feature construction. However, multiple constructed features can also be generated from a single GP tree as proposed by Ahmed et al. in [10]. Furthermore, since GP has a built-in feature selection mechanism, the selected features in the evolved tree are found to be relevant [6]. Therefore, the performance of five different combinations of selected and constructed features from a single GP tree is also investigated. Figure 5.2 shows a simple GP tree, which combines Feature 1 (F_1), Feature 5 (F_5), Feature 7 (F_7), Feature 9 (F_9), and a constant value 0.8 using operators $+$, $-$, $\times$ and $/$ to construct a new feature. Five different feature sets can be generated from this example tree as follows:

1. Set 1: The single constructed feature only ("CF"), which is $F'_0 = (F_1 + F_7) - (0.8 \times (F_7/F_9 + F_5))$;

2. Set 2: Terminal feature set that is used to construct the new feature ("Ter"), which is $\{F_1, F_5, F_7, F_9\}$;

3. Set 3: The combination of Sets 1 and 2 ("CFTer"), which is $\{F_0^{'}, F_1, F_5, F_7, F_9\}$;

4. Set 4: Multiple constructed features from all possible subtrees of the GP tree ("mCF"), which is $\{F'_0, F'_1, F'_2, F'_3, F'_4\}$, where $F'_1 = F_1 + F_7$, $F'_2 = 0.8 \times (F_7/F_9 + F_5)$; $F'_3 = F_7/F_9 + F_5$; and $F'_4 = F_7/F_9$;

5. Set 5: The combination of Sets 2 and 4 ("mCFTer"), which is $\{F_1, F_5, F_7, F_9, F'_0, F'_1, F'_2\}$.

Among the five feature sets, Set 1 has been used in many existing methods [166, 174]. Set 2 and Set 4 have been proposed in [10]. The other subsets have not been proposed. In addition, no investigation has been done to compare the performance of these sets.

5.2.4 Fitness Function

When applied to high-dimensional data, GP usually needs a large population size to cover the huge search space with thousands of features. This may lead to an expensive evaluation in terms of computational cost. Therefore, a filter approach is preferred to speed up this process [10]. However, while filters are said to be faster than wrappers, their performance is usually not as good as wrappers. In CGPFC, an embedded approach that compromises these two approaches is proposed. Since a GP tree can be used as a classifier, GP can be employed as an embedded method for feature construction. A GP tree or the corresponding constructed feature works as a simple classifier that can classify an instance x by executing Rule 5.4. The classification accuracy of GP on training data will be used as a fitness measure to guide the search.

$$\begin{aligned} IF \;\; constructedF <= 0 \;\; THEN \;\; x \in class_0; \\ ELSE \;\; x \in class_1. \end{aligned} \tag{5.4}$$

Since many of the high-dimensional datasets are unbalanced data, the balanced accuracy [30, 187] as shown in Equation (5.5) is used to evaluate

the fitness of each GP individual. Equation (5.5) is essentially the same as Equation (3.1) and Equation (4.2). TP, TN, FN, FP are the numbers of true positive, true negative, false negative and false positive respectively. The same weight 1/2 is used for each class to treat the two classes as equally important.

$$fitness = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{5.5}$$

Algorithm 6: The pseudo code of CGPFC

Input : Training_data, redundancy_threshold θ
Output : Constructed feature and selected features

```
1  begin
2      clusters ← RFC(Training_data, θ);
3      Initialise population using the best feature in each cluster of clusters;
4      while Maximum generation is not reached or the best solution is not found do
5          // The best solution is the GP tree that obtains 100% accuracy
6          for i = 1 to Population Size do
7              transf_train ← Calculate constructed feature of individual i on
                Training data (transf_train has only one new feature) ;
8              Evaluate transf_train by applying Rule (5.4);
9              fitness ← accuracy based on Equation (5.5);
10         end
11         Select parent individuals using tournament;
12         Create offspring individuals by applying crossover or mutation to the
            selected parents;
13         Place new individuals into the population of the next generation;
14     end
15     Return the constructed feature and selected features of the best individual;
16 end
```

5.2.5 Cluster-Based Feature Construction Method: CGPFC

Algorithm 6 describes the pseudo code of CGPFC. Given a training set and a redundancy threshold, the algorithm will return the combination of one constructed feature and the selected features that are used to construct the new feature.

First of all, the feature clustering procedure RFC is called to create a set of clusters from the training data. The most relevant feature (based on SU measure) in each cluster is employed to initialise GP individuals. Lines 5-9 are used to evaluate GP individuals. The loop of evaluation-selection-evolution (lines 4-13) is executed until the stopping criterion is met. Then, the constructed and selected features of the best individual are returned.

5.3 Experiment Design

5.3.1 Datasets

Eight binary-class gene expression datasets, namely Colon, DLBCL, Leukemia, CNS, Prostate, Ovarian, Alizadeh, and Yeoh) with thousands of features are used to examine the performance of the proposed method on high-dimensional data. Details about these datasets are shown in Table 1.1 on Page 21.

Since gene expression data usually contains substantial noise generated during the data collection in laboratories, discretisation is applied to reduce noise as suggested in [63]. Each feature is first standardised to have zero mean and a standard deviation of one. Then its values are discretised into -1, 0 and 1 representing the three states that are the under-expression, the baseline and the over-expression of a gene. Values that fall in the interval $[\mu - \sigma/2, \mu + \sigma/2]$, where μ and σ are mean and standard deviation of the feature values, are transformed to state 0. Values that are in the left or in the right of this interval will be transformed to state -1 or 1, respectively.

Table 5.1: Parameter settings

Function set	$+$, $-$, $\times$, %, $sqrt$, min, max, if
Terminal set	Features and random constant values
Population size	– *GPFC:* #features $\cdot$ α
	– *CGPFC:* #clusters $\cdot$ β
Maximum generations	50
Initial population	Ramped Half-and Half
Maximum tree depth	17
Selection method	Tournament Method
Tournament size	7
Crossover rate	0.8
Mutation rate	0.2
Elitism	1
Redundancy threshold	0.9

5.3.2 Baseline Methods

The performance of the proposed method was tested by comparing the discriminating ability of the constructed feature and selected features by CGPFC versus the original features and those constructed and selected by the standard GP (GPFC). The comparison is done based on the classification accuracy of three common learning algorithms including K-nearest neighbour (KNN) with $K = 1$, Naive Bayes (NB) and Decision Tree (DT).

Since the principal component analysis algorithm (PCA) is a traditional well-known algorithm in feature transformation, it is also compare with CGPFC using the most important component returned by PCA.

5.3.3 Parameter Settings

Table 5.1 shows the parameter settings for both GPFC and CGPFC. The function set includes four basic arithmetic operators and three functions that are used to construct a numeric feature from the selected features and random constants.

Since the numbers of features in these datasets are quite different, ranging

from about two thousand to fifteen thousand, the search spaces of these problems are very different. Therefore, the population size is set proportional to the size of the terminal set. Since GPFC uses the full feature set to form the terminal set, the population size is set to #features$\cdot$ α. Due to the limitation of computer memory, α is set to 3 for datasets with the number of features less than 5,000, and 2 for the others. In CGPFC, the terminal set takes only one representative feature from each cluster. Therefore, the population size is set to #clusters $\cdot$ β. β is set to 20 in this experiment. The mutation rate is set to 0.2, but after generation 10 it gradually increases with a step of 0.02 in every generation to avoid stagnation in local optima. The crossover rate is also updated accordingly to ensure the sum of these two rates always equal to 1. The redundancy threshold is empirically set to 0.9. The stopping criterion is either GP reaches the maximum generation or the best solution is found. The best solution is the GP tree that obtains 100% classification accuracy (i.e. $fitness$ is 1).

5.3.4 Experiment Configuration

Due to the small number of instances in each dataset, a stratified 10-fold CV on the whole dataset is used to create training and test data. In the fitness function, the single constructed feature or GP tree is used as a classifier classifying all instances in the training set based on Rule 5.4 on Page 153 to evaluate the discriminating ability of itself. Note that there is no inner cross-validation loop in the fitness function of an embedded method.

Since GP is a stochastic algorithm, CGPFC is run on each dataset 50 times independently with different random seeds. Therefore, a totally 500 runs (50 runs combined with 10-fold CV) are executed on each dataset. Experiments were runs on PC with Intel Core i7-4770 CPU @ 3.4GHz, running Ubuntu 4.6 and Java 1.7 with a total memory of 8GB. The results of 50 runs from each method were compared using Wilcoxon statistical significance test [241], which is a non-parametric rank sum test. A significance level of 0.05 was used.

5.4 Results and Discussions

5.4.1 Performance of the Constructed Feature

Table 5.2 shows the average test accuracy obtained by KNN, NB and DT using the constructed feature by CGPFC over the 50 runs. The results are compared with "Full"(i.e. using the original feature set) and the constructed feature by the standard GP (GPFC). Column "#F" shows the average size of each feature set. The number of instances in each dataset is also displayed in parentheses under its name, followed by the average computation time in second that CGPFC spent on each run to construct a single feature. The following columns display the best (B), mean and standard deviation of the accuracy (M±Std) obtained by KNN, NB and DT on the corresponding feature set. The highest average accuracy of each learning algorithm for each dataset is bold. Columns S_1, S_2, and S_3 display the Wilcoxon significance test results of the corresponding method over CGPFC with a significance level of 0.05. "+" or "–" means that the result is significantly better or worse than CGPFC and "=" means that their results are similar. In other words, the more "–", the better the proposed method.

5.4.1.1 Computation Time

The average running time in the first column of Table 5.2 showed that by using an embedded approach, the running time of the proposed method was quite fast. The smallest dataset (Alizadeh) took less than one second and the largest datasets (Ovarian) required a little bit more than 4 minutes to finish one run. We noted that there was a high correlation between the running time and the dataset size including both the number of instances and the number of features.

Table 5.2: Results of the constructed feature

Dataset	**Method**	**#F**	**B-KNN**	**M±Std-KNN**	S_1	**B-NB**	**M±Std-NB**	S_2	**B-DT**	**M±Std-DT**	S_3
Colon	Full	2000	74.29		=	72.62		–	74.29		=
(62)	PCA	1	43.81		–	64.76		–	64.76		–
2.04(s)	GPFC	1	82.14	72.42 ±4.89	–	80.47	70.37 ±4.62	–	82.14	73.01 ±4.51	–
	CGPFC	1	86.91	**75.43 ±3.60**		85.48	**75.53 ±3.65**		86.91	**75.39 ±3.68**	
DLBCL	Full	5469	84.46		–	81.96		–	80.89		–
(77)	PCA	1	68.75		–	76.79		–	75.36		–
15.65(s)	GPFC	1	96.07	86.46 ±3.94	–	93.57	86.01 ±4.35	–	94.64	86.39 ±3.95	–
	CGPFC	1	97.50	**88.49 ±4.01**		97.50	**88.68 ±3.61**		97.50	**88.40 ±4.09**	
Leukemia	Full	7129	88.57		–	91.96		+	91.61		+
(72)	PCA	1	61.67		–	60.17		–	62.67		–
17.14(s)	GPFC	1	94.46	88.92 ±3.00	=	94.46	87.23 ±4.30	–	95.89	88.80 ±3.17	=
	CGPFC	1	94.64	**89.82 ±3.20**		94.64	89.73 ±3.36		94.64	89.86 ±3.19	
CNS	Full	7129	56.67		–	58.33		–	50.00		–
(60)	PCA	1	55.00		–	66.67		+	65.00		+
7.17(s)	GPFC	1	70.00	57.29 ±5.82	–	70.00	58.04 ±5.65	–	70.00	57.46 ±5.88	–
	CGPFC	1	70.00	**61.54 ±4.04**		70.00	61.33 ±4.45		70.00	61.50 ±4.03	
Prostate	Full	10509	81.55		–	60.55		–	86.18		=
(102)	PCA	1	53.64		–	61.64		–	51.91		–
128.21(s)	GPFC	1	90.18	83.47 ±3.38	–	90.18	83.20 ±3.75	–	90.18	83.82 ±3.01	–
	CGPFC	1	92.27	**85.88 ±2.72**		92.27	**85.95 ±2.74**		92.27	**85.93 ±2.74**	
Ovarian	Full	15154	91.28		–	90.05		–	98.41		–
(253)	PCA	1	56.54		–	64.01		–	64.03		–
255.40(s)	GPFC	1	99.62	97.84 ±1.10	–	99.62	97.25 ±1.45	–	99.62	97.84 ±1.06	–
	CGPFC	1	99.62	**98.62 ±0.56**		99.62	**98.52 ±0.63**		99.62	**98.63 ±0.57**	
Alizadeh	Full	1095	77.00		=	92.50		+	78.50		=
(42)	PCA	1	48.00		–	33.00		–	48.00		–
0.97(s)	GPFC	1	90.50	**78.59 ±6.13**	=	88.50	77.43 ±6.27	=	90.50	78.19 ±6.31	=
	CGPFC	1	95.00	78.58 ±5.85		95.00	78.48 ±5.91		95.00	**78.58 ±5.85**	
Yeoh	Full	2526	89.97		–	93.57		–	97.57		=
(248)	PCA	1	73.03		–	81.83		–	82.67		–
16.12(s)	GPFC	1	99.17	97.11 ±0.95	–	97.57	95.20 ±2.45	–	99.17	97.13 ±0.90	–
	CGPFC	1	99.18	**97.59 ±0.84**		98.78	**97.17 ±0.98**		99.18	**97.59 ±0.84**	

5.4.1.2 CGPFC versus Full

As can be seen from Table 5.2 that using only a single feature constructed by CGPFC, KNN obtained significantly better results than Full on six datasets with the highest improvement in the average accuracy of 8% on Yeoh, and 14% in the best result on the CNS dataset. On the remaining two datasets namely Colon and Alizadeh, it had a similar performance on average. However, in the best case, it achieved 12% and 18% higher accuracy, respectively.

Similar to KNN, the constructed feature by CGPFC also helped NB achieve 3% to 25% higher accuracy than Full on six datasets. Using only one constructed feature on Prostate, the best accuracy NB achieved is 32% higher than Full. On Leukemia, although NB obtained 2% lower than Full on average, its best accuracy was still 3% higher. On Alizadeh, the accuracy of CGPFC constructed feature was significantly lower than Full. We also noted that the accuracy of NB on the Full feature set of this dataset was much higher than KNN and DT. This phenomenon indicated that the features in this dataset were quite independent to meet the NB assumption, and they might include a significant amount of noise data which could be averaged out in NB by estimating conditional probabilities; however, these noise strongly affected the performance of KNN and DT.

DT also benefited from the constructed feature, shown as significant improvement in its performance on three datasets and obtaining a similar result on four. The highest improvement was on CNS with 11% increase on average and 20% in the best case. Although the average accuracy was slightly worse on Leukemia, the best accuracy DT obtained on this dataset was still 3% higher than Full.

In general, over the 24 comparisons between CGPFC and Full on the 8 datasets and 3 learning algorithms, the constructed feature by CGPFC won 15, drew 6 and lost 3. The results indicated that the CGPFC constructed feature had much higher discriminating ability than the original feature set with thousands of features.

5.4.1.3 CGPFC versus PCA

The result of PCA in Table 5.2 showed that using the first component of PCA, the three learning algorithms obtained significantly worse accuracy than using the feature constructed by CGPFC or using Full on almost all datasets. PCA's results on Colon and Alizadeh were even worse than using random guess. Only 2 out of the 24 comparisons, the first principle component obtained better performance than CGPFC constructed feature, which was on the CNS dataset. The results showed that using the CGPFC constructed feature, all the three learning algorithms could effectively and efficiently improve their performance. However, given that PCA is an unsupervised method that can only linearly transform the features and it is not designed to construct one strong component, these results are not surprising. Therefore, this is certainly not a fair comparison. We showed PCA results here so that readers could have a rough idea about the performance of one PCA component on these datasets.

5.4.1.4 CGPFC versus GPFC

Compared with GPFC, CGPFC helped KNN further improve its results to achieve the best results on almost all datasets. The result was significantly better than GPFC on six datasets and similar on the other two. Similarly, using the CGPFC constructed feature, NB obtained significantly better results than using the GPFC constructed feature on seven datasets with the highest improvement of 5% on Colon. Applying the CGPFC constructed feature on DT also gave similar results as KNN with significant improvement on six datasets and equivalent on the remaining two.

In summary, the CGPFC constructed feature won 19, drew 5, and lost 0 out of the 24 pairs of comparisons with GPFC. The results showed that by reducing the irrelevant and redundant features in the GP terminal set, the constructed feature had a better discriminating ability than the one constructed from the full feature set.

5.4.2 Performance of the Selected Features

In GP based methods, there is a built-in feature selection process which selects informative features from the original set to construct the new feature. This subsection will investigate whether CGPFC selects a smaller number of features with better discriminating ability than GPFC and using all features. Table 5.3 shows the average test results of GPFC and CGPFC selected features in the 50 runs. Besides the highest average accuracy, the smallest average size among the feature sets is also in bold.

5.4.2.1 CGPFC versus Full

As can be seen from Table 5.3, the number of features selected by CGPFC was very small compared with the original feature set size. On five out of the eight datasets, CGPFC selected less than 10 features from thousands to construct a new feature.

Using such a small number of features, KNN had significantly better accuracy than using Full on six out of the eight datasets. Using about 4 features selected by CGPFC from the 15,154 original features in Ovarian, KNN increased 7% on its average accuracy and obtained 100% in the best case. On Leukemia and CNS, although KNN attained a similar accuracy as Full on average, it still obtained 6% and 14% higher accuracy in the best case, respectively.

Similarly, NB also significantly improved its accuracy on six datasets and worse on two. The set of about 18 features selected by CGPFC among 10,509 features on Prostate helped NB increase 30% on average accuracy and 35% in the best case.

Using features selected by CGPFC, DT significantly increased its accuracy on four datasets and similar on one. On the remaining three datasets, namely Leukemia, Prostate and Alizadeh, the selected features slightly degraded DT accuracies in about 2% to 4%. However, the best accuracies attained by the CGPFC selected features on these datasets were still higher than Full.

Table 5.3: Results of the selected features

Dataset	**Method**	**#F**	**B-KNN**	**M±Std-KNN**	S_1	**B-NB**	**M±Std-NB**	S_2	**B-DT**	**M±Std-DT**	S_3
Colon (62)	Full	2000.0	74.29		–	72.62		–	74.29		–
	GPFC	22.0	87.14	76.46 ±5.01	–	85.48	75.97 ±3.95	=	84.05	73.48 ±4.91	–
	CGPFC	22.7	83.81	**78.53 ±2.82**		87.38	**77.46 ±3.74**		84.05	**76.33 ±5.20**	
DLBCL (77)	Full	5469.0	84.46		–	81.96		–	80.89		–
	GPFC	14.8	95.00	86.70 ±3.76	–	96.25	88.65 ±3.61	=	98.75	85.48 ±5.16	=
	CGPFC	7.8	95.00	**89.27 ±3.36**		94.82	**90.06 ±2.98**		93.57	**86.09 ±3.52**	
Leukemia (72)	Full	7129.0	88.57		=	91.96		+	91.61		+
	GPFC	11.2	97.32	**89.57 ±3.51**	=	96.07	91.88 ±2.75	=	98.75	90.14 ±4.37	+
	CGPFC	5.4	94.82	89.56 ±2.81		97.50	90.99 ±2.76		93.21	87.69 ±3.01	
CNS (60)	Full	7129.0	56.67		=	58.33		–	50.00		–
	GPFC	30.5	70.00	**57.67 ±5.47**	=	70.00	60.38 ±3.99	=	73.33	57.21 ±6.04	=
	CGPFC	12.8	70.00	57.17 ±5.38		68.33	**61.25 ±3.62**		70.00	**58.38 ±4.58**	
Prostate (102)	Full	10509.0	81.55		–	60.55		–	86.18		+
	GPFC	21.6	90.36	83.48 ±3.60	–	91.27	87.16 ±2.10	–	90.18	82.70 ±3.33	–
	CGPFC	18.1	91.18	**85.80 ±2.53**		95.09	**90.62 ±2.32**		88.46	84.66 ±2.60	
Ovarian (253)	Full	15154.0	91.28		–	90.05		–	98.41		=
	GPFC	9.4	100.00	98.13 ±0.88	–	98.82	97.78 ±0.64	–	100.00	97.78 ±1.06	–
	CGPFC	4.4	100.00	**98.81 ±0.63**		99.62	**98.40 ±0.56**		100.00	**98.52 ±0.73**	
Alizadeh (42)	Full	1095.0	77.00		–	92.50		+	78.50		+
	GPFC	10.7	92.50	78.41 ±7.27	=	93.00	82.29 ±6.29	=	88.00	75.95 ±5.97	=
	CGPFC	5.5	95.00	**80.03 ±5.92**		93.00	81.85 ±5.52		86.50	75.56 ±4.61	
Yeoh (248)	Full	2526.0	89.97		–	93.57		–	97.57		–
	GPFC	14.0	94.75	91.33 ±1.95	–	98.77	**97.22 ±0.85**	=	98.77	98.62 ±0.46	=
	CGPFC	7.0	97.97	**96.21 ±1.02**		98.38	97.12 ±0.76		98.77	**98.70 ±0.15**	

In general, the CGPFC selected features had significantly better classification performance than Full on 16 cases, similar on 3 and worse on 5 over the 24 comparisons. The results indicated that CGPFC had the ability to select an extremely small number of relevant features from thousands of features to improve the performance of KNN, NB and DT.

5.4.2.2 CGPFC versus GPFC

It can be seen from Table 5.3 that although GPFC selected a very smaller number from thousands of features, the feature sets selected by CGPFC were even smaller on almost all datasets. On most datasets, the feature set selected by CGPFC was about half size or less compared to those selected by GPFC.

With a much smaller size, the CGPFC feature sets obtained either significantly better or similar classification performance as GPFC using KNN, NB and DT on almost all the 24 comparisons. Only on Leukemia, DT obtained 4% lower accuracy on average when using about 5 features selected by CGPFC compared with 11 features by GPFC. The results indicated that by using clustering to narrow the search space, GP could select a much smaller number of features with higher discriminating ability.

5.4.3 Comparisons Between Different Created Feature Sets

This section will investigate which combinations of the constructed and selected features has the best performance in improving classification accuracy of KNN, NB and DT.

Five different feature sets, namely CF, CFTer, Ter, mCF and mCFTer as described in Section 5.2.5, were created from the best GP tree at the end of each run. Training and test sets were transformed according to these feature sets and put into the three learning algorithms for performance evaluation. The average training and test accuracy of these learning algorithms on each dataset over 50 runs are reported and compared in Tables 5.4 and 5.5, respectively. In these tables, the "+" or "-" means the result of the corresponding feature set is significantly better or worse than "Full", and "=" means they are similar in the Wilcoxon tests. Note that the KNN accuracy shown in Table 5.4 are the average of 10-fold CV within the training set.

As can be seen from the third column of Table 5.5, sizes of the five created feature sets were one to two orders of magnitude smaller than the original

Table 5.4: Results on training set of the five combinations

Dataset	Subset	#F	B-KNN	A±Std-KNN	B-NB	A±Std-NB	B-DT	A±Std-DT
Colon (62) 2.04(s)	Full	2000.0	72.94		84.96		97.13	
	CF	1.0	100.00	99.96 ±0.07 +	100.00	99.83 ±0.28 +	100.00	100.00 ±0.00 +
	CFTer	23.7	89.07	87.51 ±0.96 +	96.06	94.34 ±0.92 +	100.00	100.00 ±0.00 +
	Ter	22.7	87.09	85.26 ±1.16 +	90.51	88.89 ±0.91 +	96.42	94.68 ±0.79 –
	MulCF	28.0	95.17	92.96 ±1.35 +	97.85	96.42 ±0.74 +	100.00	100.00 ±0.00 +
	MulCFTer	50.7	92.66	90.29 ±1.22 +	97.49	95.53 ±0.87 +	100.00	100.00 ±0.00 +
DLBCL (77) 15.65(s)	Full	5469.0	83.55		90.91		98.85	
	CF	1.0	100.00	99.95 ±0.09 +	100.00	99.79 ±0.24 +	100.00	100.00 ±0.00 +
	CFTer	8.8	98.70	97.40 ±0.65 +	100.00	99.57 ±0.24 +	100.00	100.00 ±0.00 +
	Ter	7.8	97.69	95.76 ±0.85 +	98.85	97.80 ±0.53 +	98.56	97.38 ±0.58 –
	MulCF	6.6	99.28	97.95 ±0.83 +	100.00	98.92 ±0.52 +	100.00	100.00 ±0.00 +
	MulCFTer	14.4	98.56	97.11 ±0.70 +	99.57	98.99 ±0.46 +	100.00	100.00 ±0.00 +
Leukemia (72) 17.14(s)	Full	7129.0	90.73		98.15		99.38	
	CF	1.0	100.00	99.93 ±0.12 +	100.00	99.87 ±0.18 +	100.00	100.00 ±0.00 +
	CFTer	6.4	98.61	97.28 ±0.61 +	99.85	99.33 ±0.28 +	100.00	100.00 ±0.00 +
	Ter	5.4	97.84	96.34 ±0.83 +	98.77	98.00 ±0.49 =	98.46	97.41 ±0.58 –
	MulCF	4.2	99.23	97.67 ±0.72 +	99.85	98.63 ±0.56 +	100.00	100.00 ±0.00 +
	MulCFTer	9.6	98.31	97.08 ±0.56 +	99.69	99.30 ±0.30 +	100.00	100.00 ±0.00 +
CNS (60) 7.17(s)	Full	7129.0	58.70		73.89		98.70	
	CF	1.0	100.00	99.97 ±0.08 +	100.00	99.86 ±0.47 +	100.00	100.00 ±0.00 +
	CFTer	13.8	90.00	86.96 ±1.77 +	100.00	99.43 ±0.41 +	100.00	100.00 ±0.00 +
	Ter	12.8	81.67	78.77 ±1.48 +	95.56	93.36 ±1.01 +	97.04	96.14 ±0.50 –
	MulCF	15.0	97.96	95.99 ±1.05 +	99.44	98.58 ±0.71 +	100.00	100.00 ±0.00 +
	MulCFTer	27.8	94.63	92.61 ±1.34 +	99.81	98.59 ±0.62 +	100.00	100.00 ±0.00 +
Prostate (102) 128.21(s)	Full	10509.0	79.30		66.67		98.59	
	CF	1.0	100.00	99.96 ±0.06 +	100.00	99.86 ±0.23 +	100.00	100.00 ±0.02 +
	CFTer	19.1	95.21	93.66 ±0.91 +	98.80	97.68 ±0.49 +	100.00	100.00 ±0.02 +
	Ter	18.1	94.01	92.07 ±0.96 +	96.63	95.49 ±0.64 +	97.39	96.77 ±0.40 –
	MulCF	17.3	96.95	95.54 ±0.74 +	98.58	97.87 ±0.43 +	100.00	100.00 ±0.02 +
	MulCFTer	35.3	95.86	94.52 ±0.76 +	98.58	97.84 ±0.48 +	100.00	100.00 ±0.02 +
Ovarian (253) 255.40(s)	Full	15154.0	90.95		91.79		99.91	
	CF	1.0	100.00	99.99 ±0.02 +	100.00	99.89 ±0.11 +	100.00	100.00 ±0.00 +
	CFTer	5.4	99.87	99.62 ±0.14 +	99.87	99.43 ±0.15 +	100.00	100.00 ±0.00 +
	Ter	4.4	99.69	99.39 ±0.23 +	99.25	98.87 ±0.18 +	99.82	99.57 ±0.12 –
	MulCF	3.7	99.87	99.53 ±0.18 +	99.78	99.25 ±0.25 +	100.00	100.00 ±0.00 +
	MulCFTer	8.2	99.82	99.49 ±0.18 +	99.74	99.42 ±0.17 +	100.00	100.00 ±0.00 +
Aalizadeh (42) 0.97(s)	Full	1095.0	75.94		100.00		97.88	
	CF	1.0	100.00	99.87 ±0.19 +	100.00	99.85 ±0.45 –	100.00	100.00 ±0.00 +
	CFTer	6.5	98.94	95.12 ±1.40 +	100.00	99.82 ±0.21 –	100.00	100.00 ±0.00 +
	Ter	5.5	95.78	92.80 ±1.58 +	99.47	97.96 ±0.82 –	97.08	95.47 ±0.78 –
	MulCF	4.5	98.16	96.19 ±1.16 +	99.73	98.74 ±0.55 –	100.00	100.00 ±0.00 +
	MulCFTer	10.0	98.42	94.89 ±1.23 +	100.00	99.34 ±0.39 –	100.00	100.00 ±0.00 +
Yeoh (248) 16.12(s)	Full	2526.0	90.01		100.00		99.28	
	CF	1.0	100.00	99.78 ±0.12 +	99.91	99.38 ±0.58 –	100.00	99.83 ±0.08 +
	CFTer	8.0	99.01	98.20 ±0.45 +	99.87	99.67 ±0.11 –	100.00	99.83 ±0.07 +
	Ter	7.0	98.65	97.78 ±0.54 +	99.24	98.77 ±0.28 –	99.10	99.01 ±0.07 –
	MulCF	11.1	99.64	99.20 ±0.26 +	99.69	99.36 ±0.27 –	100.00	99.83 ±0.08 +
	MulCFTer	18.2	99.51	98.77 ±0.44 +	99.73	99.47 ±0.17 –	100.00	99.83 ±0.07 +

Table 5.5: Results on test set of the five combinationsr

Dataset	Subset	#F	B-KNN	A±Std-KNN	B-NB	A±Std-NB	B-DT	A±Std-DT
	Full	2000.0	74.29		72.62		74.29	
	CF	1.0	86.90	75.43 ±3.60 =	85.48	75.53 ±3.65 +	86.90	75.39 ±3.68 +
Colon	CFTer	23.7	84.05	78.76 ±2.88 +	87.14	77.88 ±3.79 +	86.90	75.39 ±3.68 +
(62)	Ter	22.7	83.81	78.53 ±2.82 +	87.38	77.46 ±3.74 +	84.05	76.33 ±5.20 +
	MulCF	28.0	85.24	76.97 ±4.14 +	85.48	77.02 ±3.82 +	86.90	75.17 ±3.57 =
	MulCFTer	50.7	88.81	78.68 ±4.40 +	85.24	77.44 ±3.65 +	86.90	75.17 ±3.57 =
	Full	5469.0	84.46		81.96		80.89	
	CF	1.0	97.50	88.49 ±4.01 +	97.50	88.68 ±3.61 +	97.50	88.40 ±4.09 +
DLBCL	CFTer	8.8	95.89	89.46 ±3.16 +	97.50	90.19 ±2.91 +	97.50	88.40 ±4.09 +
(77)	Ter	7.8	95.00	89.27 ±3.36 +	94.82	90.06 ±2.98 +	93.57	86.09 ±3.52 +
	MulCF	6.6	96.25	88.77 ±3.29 +	96.25	89.24 ±3.36 +	97.50	88.46 ±4.21 +
	MulCFTer	14.4	94.64	89.25 ±3.33 +	96.25	90.35 ±3.06 +	97.50	88.46 ±4.21 +
	Full	7129.0	88.57		91.96		91.61	
	CF	1.0	94.64	89.82 ±3.20 +	94.64	89.73 ±3.36 –	94.64	89.86 ±3.19 –
Leukemia	CFTer	6.4	94.64	90.02 ±2.59 +	94.82	91.17 ±2.65 –	94.64	89.86 ±3.19 –
(72)	Ter	5.4	94.82	89.56 ±2.81 =	97.50	90.99 ±2.76 –	93.21	87.69 ±3.01 –
	MulCF	4.2	94.64	88.40 ±3.37 =	94.64	88.32 ±3.59 –	94.64	89.79 ±3.25 –
	MulCFTer	9.6	94.82	89.87 ±2.71 +	94.64	90.44 ±2.85 –	94.64	89.79 ±3.25 –
	Full	7129.0	56.67		58.33		50.00	
	CF	1.0	70.00	61.54 ±4.04 +	70.00	61.33 ±4.45 +	70.00	61.50 ±4.03 +
CNS	CFTer	13.8	70.00	58.96 ±4.95 +	70.00	61.71 ±4.14 +	70.00	61.50 ±4.03 +
(60)	Ter	12.8	70.00	57.17 ±5.38 =	68.33	61.25 ±3.62 +	70.00	58.38 ±4.58 +
	MulCF	15.0	68.33	60.50 ±4.26 +	71.67	60.96 ±4.48 +	70.00	61.25 ±4.04 +
	MulCFTer	27.8	70.00	59.42 ±5.00 +	71.67	61.33 ±4.34 +	70.00	61.25 ±4.04 +
	Full	10509.0	81.55		60.55		86.18	
	CF	1.0	92.27	85.88 ±2.72 +	92.27	85.95 ±2.74 +	92.27	85.93 ±2.74 =
Prostate	CFTer	19.1	91.18	86.67 ±2.18 +	93.18	89.24 ±2.25 +	92.27	85.93 ±2.74 =
(102)	Ter	18.1	91.18	85.80 ±2.53 +	95.09	90.62 ±2.32 +	88.45	84.66 ±2.60 –
	MulCF	17.3	91.36	85.52 ±2.29 +	92.27	86.72 ±2.68 +	92.27	85.88 ±2.79 =
	MulCFTer	35.3	93.09	86.96 ±2.47 +	93.18	87.88 ±2.40 +	92.27	85.88 ±2.79 =
	Full	15154.0	91.28		90.05		98.42	
	CF	1.0	99.62	98.62 ±0.56 +	99.62	98.52 ±0.63 +	99.62	98.63 ±0.57 +
Ovarian	CFTer	5.4	99.62	98.94 ±0.60 +	99.60	98.50 ±0.55 +	99.62	98.63 ±0.57 +
(253)	Ter	4.4	100.00	98.81 ±0.63 +	99.62	98.40 ±0.56 +	100.00	98.52 ±0.73 =
	MulCF	3.7	100.00	98.50 ±0.72 +	99.22	97.78 ±0.82 +	99.62	98.64 ±0.57 +
	MulCFTer	8.2	100.00	98.86 ±0.64 +	99.62	98.40 ±0.73 +	99.62	98.64 ±0.57 +
	Full	1095.0	77.00		92.50		78.50	
	CF	1.0	95.00	78.58 ±5.85 =	95.00	78.48 ±5.91 –	95.00	78.58 ±5.85 =
Alizadeh	CFTer	6.5	95.00	80.79 ±5.95 +	95.00	80.41 ±5.85 –	95.00	78.58 ±5.85 =
(42)	Ter	5.5	95.00	80.03 ±5.92 +	93.00	81.85 ±5.52 –	86.50	75.56 ±4.61 –
	MulCF	4.5	95.00	78.78 ±5.99 =	92.50	79.70 ±5.78 –	95.00	78.51 ±5.83 =
	MulCFTer	10.0	95.00	80.30 ±6.44 +	95.00	80.70 ±5.52 –	95.00	78.51 ±5.83 =
	Full	2526.0	89.97		93.57		97.57	
	CF	1.0	99.18	97.59 ±0.84 +	98.78	97.17 ±0.98 +	99.18	97.59 ±0.84 =
Yeoh	CFTer	8.0	97.97	96.47 ±0.98 +	99.17	97.97 ±0.64 +	99.18	97.64 ±0.82 =
(248)	Ter	7.0	97.97	96.21 ±1.02 +	98.38	97.12 ±0.76 +	98.77	98.70 ±0.15 +
	MulCF	11.1	98.77	97.51 ±0.88 +	98.77	97.49 ±0.90 +	99.18	97.60 ±0.81 =
	MulCFTer	18.2	99.17	97.13 ±1.07 +	98.78	97.85 ±0.74 +	99.18	97.65 ±0.80 =

feature set. With this reduction, they could significantly speed up the learning process as well as simplify the learnt classifiers of any learning algorithm.

With such small sizes, all the feature sets generated by CGPFC helped KNN improve its performance on all datasets with extremely small amounts of features compared to the full feature set. In each dataset, the average accuracies slightly changed when using different feature sets. The difference could be up to 4% as on CNS. Among the five generated feature sets, CFTer achieved the highest average accuracy on five datasets.

Similarly, all the five feature sets also helped NB significantly improve its performance on six out of the eight datasets. On Leukemia and Alizadeh, all the five created feature sets had a worse performance than Full on average. However, they still had higher accuracy in the best case. For DT, similar classification performance was witnessed in all the feature sets except for the selected feature set (Ter) with slightly lower accuracies. Only on Colon and Yeoh, this feature set had a slightly better performance than the others. The same accuracies were obtained by CF and CFTer on almost all datasets, which showed that the learnt decision trees in both cases were the same. A closer look at the learnt decision trees showed that they used only the single constructed feature. This indicated that the constructed feature had much better discriminating abilities than the selected ones, resulting in the decision trees with the size of several nodes only.

Friedman test with Tukey as the post hoc test was used to confirm which feature set to be significantly better than others. R package is used to run this test. The results showed that there was no significant difference between different created feature sets for DT. The reason may be related to the fact that since the single constructed feature had much better discriminating power than the other features in the set, DT always used it to build classifiers. However, only one feature might not be enough to represent the whole complex feature space of the original problem.

For KNN and NB, on the other hand, a significant difference between these feature sets was found with *p-value 0.0002* for KNN and *0.0249* for NB.

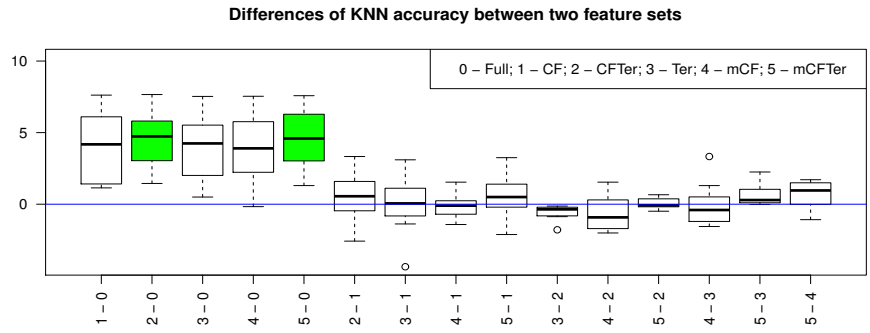


Figure 5.3: Friedman with Tukey post-hoc test for KNN on eight datasets.

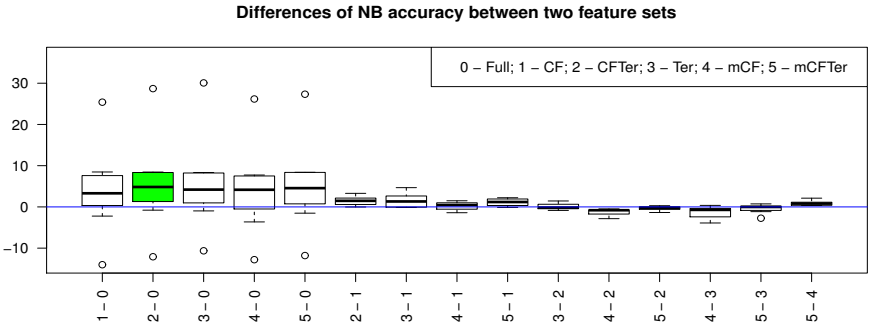


Figure 5.4: Friedman with Tukey post-hoc test for NB on eight datasets.

Figure 5.3 and Figure 5.4 show the boxplots of the differences between pairs of created feature sets in KNN and NB results respectively. In these figures, the "Full" and the five created sets are indexed from zero to five with the same order as shown in Table 5.5. If the difference between two feature sets is significant with $p\text{-}value < 0.05$, its corresponding boxplot is filled.

The first five boxes in Figure 5.3 showed that on average all the five created sets had better results than the full feature set. Two differences among the five were significant, i.e., CFTer (i.e. subset 2 or the combination set of constructed and selected features) with a *p-value 0.0002* and mCFTer (i.e. subset 5 or the combination set of all multiple constructed features from

all subtrees and selected features) with a *p-value 0.0014*. However, there was no difference between these two feature sets as shown in the boxplot (5-2). Furthermore, as can be seen from Table 5.5, the size of CFTer was much smaller than mCFTer on all datasets. Using CFTer, KNN ran much faster than using mCFTer. Therefore, among the five created sets, the combination of constructed and selected features is the best feature set in improving KNN performance in terms of classification accuracy and computation time.

This finding is also held for NB. In Figure 5.4, (2-0) is the only filled boxplot, which indicates that CFTer is the only feature set that helps NB achieve significantly better performance than Full on all datasets. Note that while subset 5 (mCFTer) helped KNN achieve significantly better performance than Full on all datasets, it could not do the same thing for NB. This may be related to the fact that features constructed from all subtrees can be highly correlated or redundant, which makes NB's assumption on features to be conditionally independent become invalid.

An investigation on the training and test results of the five combinations of constructed and selected features reveals the reason why CFTer is the best feature subset among them. Comparing the training results of CF with CFTer in Table 5.4 showed that CF had higher KNN and NB accuracy than CFTer on almost all datasets. These results were expected since the evolutionary process of GP aimed to maximise the classification accuracy of CF. However, the test results of KNN and NB using CF and CFTer in Table 5.5 showed an opposite pattern on almost all datasets. This indicated that the single constructed feature might overfit the training data and thus its performance on unseen test data was slightly degraded. Therefore, combining it with the selected features helped to alleviate the overfitting problem.

The results again confirmed the ability of GP in selecting informative features and constructing new features that can significantly reduce the feature set size from thousands to tens of features while improving or at least maintaining the classification performance of the common learning algorithms.

5.5 Further Analysis

This section will analyse the feature clusters created by RFC and constructed features to further investigate the effectiveness of the proposed method as well as its generalisation ability.

5.5.1 Cluster Analysis

To validate the structure of the clusters generated by the proposed algorithm, this section investigates the cohesion or compactness within each cluster as well as the separation or isolation between different clusters.

Silhouette analysis [201] is a popular method to study both the cohesion and the separation of clusters. Equation (5.6) displays the calculation of the silhouette coefficient of a feature i in which a_i is the average distance of feature i to all other features in the same cluster, and b_i is the minimum average distance of feature i to other clusters. Given that c_i is the cluster that includes feature i, and c_k is other clusters, $CC(f_i, f_j)$ is the correlation coefficient (CC) between features i and j, a_i and b_i are calculated based on Equations (5.7) and (5.8). Since CC (see Equation (5.3)) measures the correlation level or similarity between 2 features and has a value between 0 and 1, (1 - CC) is used as a distance or dissimilarity measure between them.

$$s_i = \frac{(b_i - a_i)}{max(a_i, b_i)} \tag{5.6}$$

where

$$a_i = \frac{1}{size(c_i)} \sum_{j=1}^{size(c_i)} (1 - CC(f_i, f_j)), i \neq j \tag{5.7}$$

$$b_i = \min_{\forall c_k \neq c_i} (\frac{1}{size(c_k)} \sum_{j=1}^{size(c_k)} (1 - CC(f_i, f_j))) \tag{5.8}$$

The value of the silhouette coefficient ranges from -1 to 1, where -1 is the worst and 1 is the best case. The average silhouette coefficient (ASC) of all features is an overall measure indicating the goodness of a clustering. Since

Table 5.6: Results of cluster analysis

Dataset	#Features	#Clusters	%Dimensionality reduction	ASC
Colon	2000	104.10	0.95	0.80
DLBCL	5469	819.20	0.85	0.96
Leukemia	7129	901.30	0.87	0.98
CNS	7129	79.30	0.99	1.00
Prostate	10509	1634.80	0.84	0.85
Ovarian	15154	601.20	0.96	0.31
Alizadeh	1095	93.60	0.91	0.94
Yeoh	2526	97.60	0.96	1.00

the experiments were conducted based on a 10-fold CV framework on each dataset, the average of ASC over 10 folds was calculated. Table 5.6 shows the original number of features, the average number of clusters generated with the redundancy level of 0.9, the percentage of dimensionality reduction, and the average of ASC over 10 folds of each dataset.

As can be seen from the fourth column of Table 5.6, all datasets obtained at least 84% of dimensionality reduction after applying the proposed feature clustering algorithm. The number of input features into GP was significantly reduced with the largest reduction of 99% on CNS and 96% on Ovarian and Yeoh. The third column of Table 5.6 also showed differences in the number of clusters generated on different datasets regardless of its original number of features. For example, CNS had a much smaller number of clusters than Colon although its original feature set size was more than three times larger than Colon. This again confirms that it is very difficult to predefine an appropriate number of clusters for each dataset to maintain a certain level of redundancy among all features in the same cluster.

Results of the silhouette coefficients shown in the last column of the table were 0.8 or above on all datasets except for Ovarian. This indicates that the clusters generated by RFC were compact and separated. Only on Ovarian, this coefficient was low. An investigation on this dataset showed that most

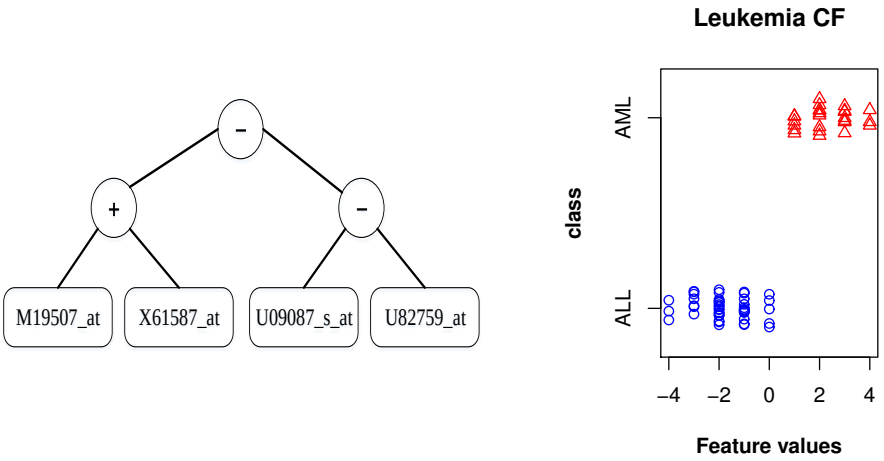


Figure 5.5: Leukemia constructed feature.

of the features in this dataset have their correlation coefficient with other features higher than 0.5. Therefore, features in different clusters might still be highly correlated but with a lower level than the predefined threshold (0.9). However, even though its silhouette coefficient was low (0.31), the results of this dataset shown in Table 5.2 revealed that the feature clustering method enabled the constructed feature to perform significantly better than the one constructed from the whole feature thanks to the significant reduction of 96% in dimensionality.

5.5.2 The Constructed Features

This section investigates the reason why the constructed and selected features can achieve good performance by showing a constructed feature by a GP run on a dataset as a typical example. Leukemia was chosen because the size of the constructed feature (or GP tree size) on this dataset is smaller than others. This is also a challenging dataset as can be seen from its results in Tables 5.2 and 5.3.

Figure 5.5 shows the GP tree of the constructed feature and its values

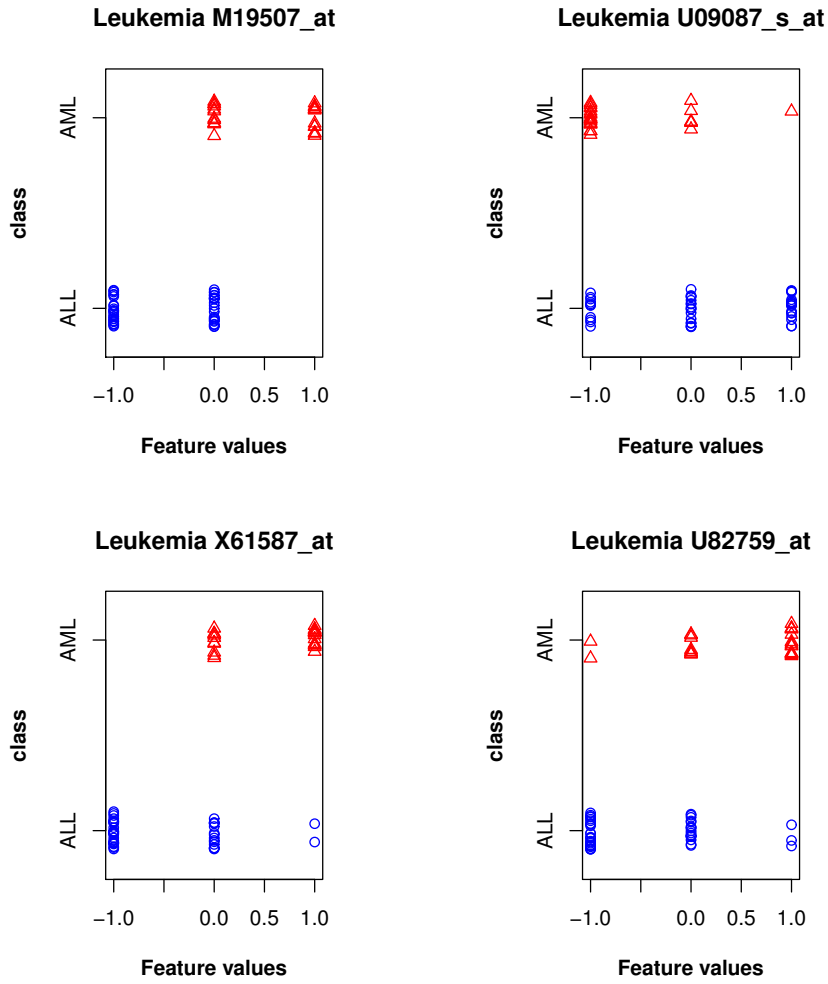


Figure 5.6: Leukemia original features.

returned by CGPFC in the run of seed 1 on fold 5 of the Leukemia dataset. It was constructed from four original features, which are feature *M19507_at, X61587_at, U09087_s_at, and U82759_at* whose values are plotted in Figure 5.6. Note that this dataset has two classes of leukemia patients, namely acute lymphoblastic leukemia (ALL) or acute myelogenous leukemia (AML).

It can be seen from these scatter plots in Figure 5.6 that the selected features had low impurity or high correlation to the class labels. Specifically,

in feature *M19507_at*, the value -1 is corresponding to class ALL while the value 1 is corresponding to class AML. In other words, these selected features were relevant to the problem. By combining the four selected features, the constructed feature split instances in the two classes into two completely separate intervals. Therefore, using this constructed feature, GP and other learning algorithms could easily learn a classifier that can classify all instances into its correct class. However, results in Table 5.2 showed that this ideal case did not always happen in all runs and all folds. We will investigate the reason in the following section.

5.5.3 Generalisability

During the evolutionary process, an evolved constructed feature or GP tree is evaluated based on the classification accuracy of itself on the training set. When GP reaches the maximum number of generations or the best solution (i.e. the GP tree that obtains 100% accuracy on training data) is found, the best solution found so far is returned. A comparison between the training accuracy which is also the fitness value and the test accuracy obtained by the GP tree or the constructed feature was conducted to see if the returned solutions can generalise well to the unseen test data. Table 5.7 presents the average training accuracy, test accuracy and the difference between them over the 50 runs. In this table, the datasets are listed in an ascending order by the number of instances to better show the influence of the sample size or the number of training instances on the generalisability of the proposed method.

As can be seen from Table 5.7, in general, the constructed features obtained optimal solutions on almost all training sets. However, their performances on unseen test set were quite different, especially for the first three datasets. This phenomenon suggested that in some cases the constructed features were overfitted to the training data. Values in the last column of the table roughly revealed a descending order of differences between training and test accuracies from very large gaps in the first three datasets to very small ones in the last two, which was negatively correlated with the number of instances shown in

Table 5.7: Results of training and test accuracy

Dataset	#Instances	Training Accuracy	Test Accuracy	Difference
Alizadeh	42	100.00	78.46	21.54
CNS	60	100.00	54.51	45.49
Colon	62	100.00	70.94	29.06
Leukemia	72	100.00	89.56	10.44
DLBCL	77	100.00	83.53	16.47
Prostate	102	100.00	85.84	14.16
Yeoh	248	99.90	96.00	3.90
Ovarian	253	100.00	98.62	1.38

the second column. This indicated that the smaller the number of instances was, the severer the overfitting problem would be.

However, this rule did not always hold. For example, the results of the first two datasets shown in Table 5.7 showed that constructed features on Alizadeh had better generalisation than CNS although it has only 42 instances while CNS has 60. This phenomenon can be explained by investigating the distributions of each feature in these two datasets. Figures 5.7 shows the distributions of 50 features from the two datasets. Results in the figure showed that compared with Alizadeh, CNS features had a much more skewed distribution with a lot of outliers scattering far away from its mean value. In the experiments, CNS was divided into 10 folds, each of which has 6 instances. Therefore, it was likely that the distributions of the training and the test folds were very different. As a result, the constructed or selected features based on the training fold could not generalise well to correctly predict the unseen data in the test fold.

This hypothesis is confirmed by investigating the accuracies of each classification algorithm on each data fold of the CNS dataset. The results of full feature was used to leave out the effect of feature selection and construction. Table 5.8 shows the test accuracies of KNN, NB and DT obtained on each fold. The last row presents the largest difference between accuracies obtained by each learning algorithm on 10 folds. The results showed that all the three learning algorithms obtained very different accuracies on different folds. Some had as high accuracy as 87.5%, some had even worse results than the random

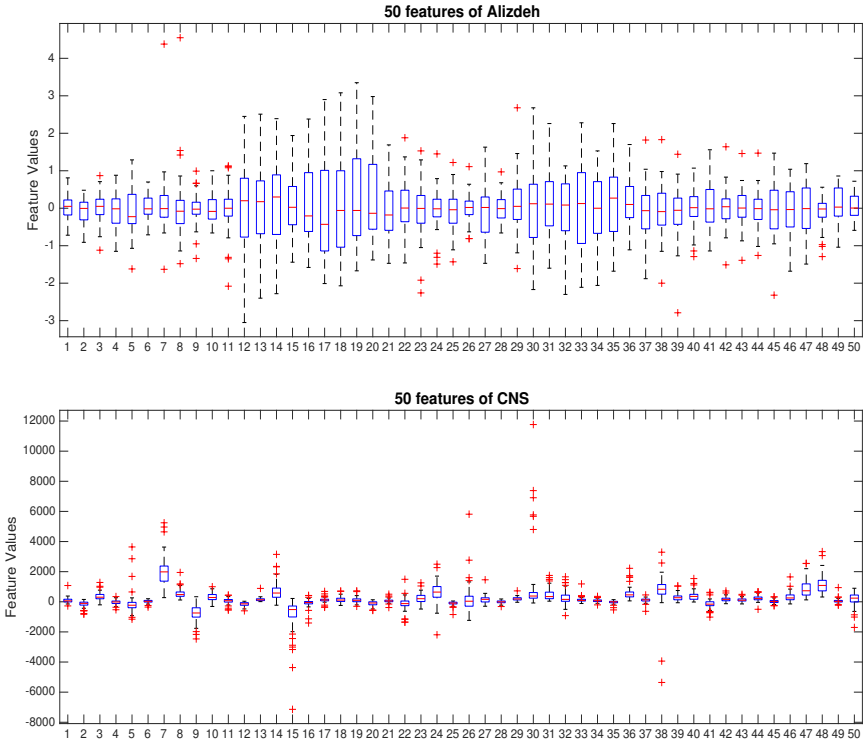


Figure 5.7: Distributions of 50 features of Alizadeh and CNS.

guess (37.5%). Therefore, the gap between the obtain accuracies in different folds became very high with the maximum of 62.5% in KNN.

The results showed that some test folds might have very different distributions to their corresponding training folds. As a result, it is difficult for learning algorithms to learn a model that can perform well on the test folds. In addition, with a small number of instances in one test fold (less than ten instances in many datasets), one misclassified instance can significantly decrease the classification accuracies. Combination of this small sample size problem and the skew distribution of the features makes CNS and Colon become the most challenging datasets with the highest level of overfitting as shown in Table 5.7. This explanation is concordant to the result of the Ovarian dataset where the gap between training and test accuracies is just

Table 5.8: Results of "Full" feature set on each fold of Colon dataset

Fold	KNN	NB	DT
0	50.00	50.00	50.00
1	62.50	75.00	50.00
2	62.50	62.50	50.00
3	62.50	87.50	50.00
4	62.50	50.00	37.50
5	37.50	87.50	25.00
6	75.00	37.50	25.00
7	87.50	37.50	62.50
8	25.00	75.00	37.50
9	62.50	50.00	75.00
Max difference	**62.50**	**50.00**	**50.00**

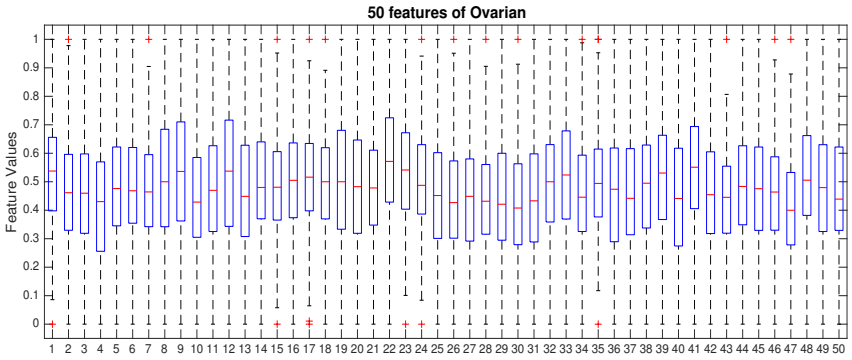


Figure 5.8: Distributions of 50 features of Ovarian.

1.38%. The boxplot of the first 50 features of Ovarian in Figure 5.8 shows that Ovarian features have rather symmetric distributions without many outliers.

Furthermore, investigating the Yeoh dataset showed that given a significant number of instances, GP still achieved a good generalisation despite the fact that Yeoh features have very skew distribution with a lot of outliers as shown in Figure 5.9. In this case, although outliers existed, the training set was still large enough (about 220 instances) to be representative of the whole distribution.

In conclusion, the proposed algorithm can cope with small sample size as well as skew distribution to obtain a good generalisation as long as the given training data has enough information to represent the distribution of

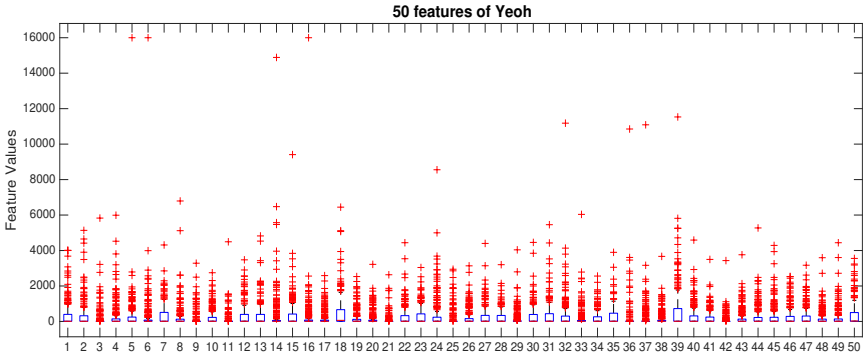


Figure 5.9: Distributions of 50 features of Yeoh.

the unseen data.

5.6 Chapter Summary

The goal of this chapter was to apply feature clustering to GP for feature construction in classification in order to improve its performance on high-dimensional data. The goal has been achieved by proposing a new feature clustering algorithm to cluster redundant features in the same group based on a correlation or redundancy level. Then the best feature from each cluster is fed into GP to construct a single new high-level feature. Performance of the constructed feature is tested on three different classification algorithms. The results on eight gene expression datasets having two classes have shown that the proposed method can construct a single feature that can help these learning algorithms improve classification performance of using thousands of the original features. Comparisons between CGPFC and the standard GP for feature construction also showed that feature clustering helps GP construct features with better discriminating ability than those created from the whole feature set. The results also showed the effectiveness and efficiency of the proposed feature clustering technique in automatically determining the number of feature clusters.

Investigation on the performance of different created feature sets from the

single GP tree showed that all the five feature sets help the three learning algorithms improve their classification accuracy. Furthermore, thanks to their much smaller sizes, these feature sets also reduce the learning time and simplify the learned classifiers. Among the five feature sets, the combination of the single constructed feature and the selected ones help KNN and NB significantly improve their performance on all datasets. For DT, adding the selected features does not make any difference in its performance compared to using only the single constructed one most likely due to the built-in feature selection ability of DT.

In this chapter, GP has shown promise in feature construction. However, only one constructed feature may not be enough to represent the complex feature space of the original thousands of features, which prevents it from achieving higher classification accuracy. On the other hand, creating multiple constructed features from a single tree does not make a significant difference. Therefore, multiple feature construction using multi-tree GP should be considered to create a larger number of independently constructed features with better discriminating ability. Furthermore, using the embedded approach, GP may construct features that are too overfitted to the training data since the constructed feature is evaluated based on its performance as a classifier on the whole training set. This problem is even worse when the number of examples or instances given for training is small. Next chapter will investigate multiple feature construction using multiple-tree GP with filter and/or wrapper approaches for feature evaluation.

6

Class-Dependent Multiple Feature Construction Using GP

6.1 Introduction

The previous chapter has shown the ability of GP in constructing features with better discriminating power. However, it may not enough for a single feature to represent the whole complex feature space of the original high-dimensional data. Meanwhile, the previous chapter showed that multiple features constructed based on all possible subtrees of a single GP tree did not obtain significantly better performance than the single feature constructed from the whole tree.

Another approach to multiple feature construction using single-tree GP is to run GP multiple times as in [174], where each time constructs one feature focusing on discriminating instances from one class from all other classes. For presentation convenience, it is called 1TGPFC here because it used the single-tree representation. 1TGPFC has shown to be effective on UCI datasets

with tens of features. However, it requires to run GP many times to construct one set of new features and the number of constructed features is limited to the number classes of the problem, which may be inefficient and insufficient for high-dimensional data. Furthermore, since each feature is independently constructed in different GP runs, it is impossible to determine the interaction of the newly constructed features, which may be redundant, overlapping to each other. As a result, the performance of the whole feature set may not be as good as expected. This problem can be overcome by using multi-tree representation, where each individual contains many trees, each of which represents a constructed feature.

Multiple-tree GP was proposed and shown effective to construct multiple features [80, 125, 213]. However, the datasets used in these studies are quite small with about tens of features. Therefore, they may not scale well to high-dimensional data. Furthermore, these methods usually construct features from all the original features, which may not be effective since some particular features may have better ability than other features to distinguish instances of one class from other classes [237]. For example, a feature may be good at distinguishing samples of class A from those of class B, C and D, but may not be good at differentiating samples of classes B from those of C and D. Therefore, it may be more difficult to construct a better discriminating feature when combing features that are relevant to different classes. Recently, class-dependent features have been considered in both conventional [77, 155, 237] and PSO based feature selection method [270]. Results of these methods showed that evaluating features in a class-based context led to a better performance. However, the research of such approaches in feature construction in general as well as using GP is still limited.

6.1.1 Chapter Goals

This chapter proposes two multiple feature construction methods using multi-tree GP for high-dimensional data, one is class-independent (MGPFC) and one is class-dependent (CDFC). The proposed methods aim to construct a

small number of new high-level features, which are expected to improve the classification performance of common learning algorithms including k-Nearest Neighbour (KNN), Naive Bayes (NB) and Decision Tree (DT). Performance of the class-dependent constructed features by CDFC will be tested and compared with the original feature set, the class-independent features constructed by MGPFC, the single feature constructed by CGFPC in Chapter 5 and those constructed by 1TGPFC [174]. Specifically, the following research objectives will be investigated:

- How to construct class-dependent features;
- Whether the features constructed by the proposed methods achieve better classification accuracy than the original full feature set;
- Which of the two proposed methods performs better in terms of classification accuracy and computation time.
- Whether the better proposed method here achieves better performance than the single feature construction method in Chapter 5 (CGPFC); and
- Whether the better proposed method outperforms the multiple feature construction method using single-tree GP [174] (1TGPFC).

6.1.2 Chapter Organisation

The remaining of this chapter is organised as follows. Section 6.3 describes the proposed multi-tree GP based feature construction algorithms including the class-independent one (MGPFC) and the class-dependent one (CDFC). Section 6.4 discusses the datasets, parameter settings and experiment configuration used to test the performance of the proposed methods. Results of the experiments are shown and discussed in Section 6.5. Further analyses are presented in Section 6.6. Finally, a summary of this chapter is given in Section 6.7.

6.2 The Proposed Method: MGPFC

6.2.1 Representation

The aim of this method is to construct a very small number of features that can improve or at least maintain the discriminating ability of the original set with thousands to tens of thousands of features. The number of new features will be set proportional to the number of classes of the problem since we hypothesise that the more classes the problem has, the more complex the solution space might be. The number of constructed features m is defined based on a given construction ratio r which can be 1, 2, 3, etc. For example, if the given construction ratio r is equal to 2, then 4 features will be constructed for a binary-class problem. Figure 6.1 shows an example of a GP individual in this case.

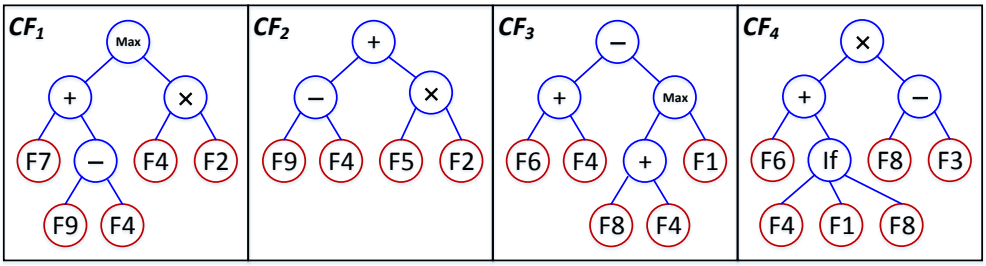


Figure 6.1: MGPFC representation.

Using this individual, 4 features are generated as follows.

- $CF_1 = Max(F_7 + (F_9 - F_4), F_4 \times F_2)$
- $CF_2 = (F_9 - F_4) + (F_5 \times F_2)$
- $CF_3 = (F_6 + F_4) - Max(F_8 + F_4, F_1)$
- $CF_4 = (F_6 + If(F_4, F_1, F_8)) \times (F_8 - F_3)$

6.2.2 MGPFC Fitness Function

To evaluate an individual, its m constructed features are used to transform the training set into a new training set with m features. The discriminating ability of the transformed dataset indicates how good the constructed features are. While a wrapper measure based on a classification algorithm can be a good indicator for searching good feature sets, the resulted feature set may not be general for other classification algorithms. On the other hand, a filter measure is based on the intrinsic characteristics of the data, its solutions may be effective for many learning algorithms, however, with the price of lower classification accuracy than wrapper approaches. Therefore, a hybrid approach that combines both wrapper and filter was proposed to synthesise their strengths. Decision Tree is used to evaluate the classification performance of the constructed feature set as it is a fast learning algorithm. For filter approach, a distance measure is chosen because it is simple and multivariate which means it can evaluate the discriminating ability of a set of features at a time.

To evaluate an individual with m trees, its constructed features are used to transform the training set into a new dataset with m features. The discriminating ability of the transformed training set will be used to determine the fitness of the individual. Equation (6.1) describes the fitness function which combines the classification performance and a distance measure using a weighting coefficient α.

$$Fitness = \alpha \cdot Accuracy + (1 - \alpha) \cdot Distance \tag{6.1}$$

where *Accuracy* is the average accuracy of DT over K-fold (K=3) cross-validation (CV) on the transformed training set. To avoid overfitting, this K-fold CV is repeated L times (L=3) with different data splitting similar to [125]. Totally, $K \times L$ models were built to evaluate the set of constructed features. Therefore, 9 models are trained in each evaluation, which requires an equivalent running time as a 10F-CV. Because many high-dimensional datasets are unbalanced, the same balanced accuracy [187] as previous chapters was

used. However, for reading convenience, it is presented again in Equation (6.2) in which c is the number of classes, TP_i and S_i are the number of correctly identified instances and the number of total instances of class i.

$$balanced\_accuracy = \frac{1}{c}\sum_{i=1}^{c}\frac{TP_i}{|S_i|} \tag{6.2}$$

The *Distance* measure [11] is calculated based on Equation (6.3) which is the same as in Chapters 3 and 4. However, the measure used in this method to approximate the distance between two vectors are different from the previous two chapters. Therefore, its description is repeated here for reading convenience. As shown in Equation (6.3), *Distance* is used to maximise the distance of instances *between* class (D_b) and minimise the distance of instances *within* the same class (D_w). D_b is approximated by the average distance between an instance and its nearest miss which is the nearest instance of other classes. D_w is approximated by the average distance between an instance and its farthest hit which is of the same class. Let S be the training set, D_b and D_w are calculated based on Equations (6.4) and (6.5).

$$distance = \frac{1}{1+e^{-5(D_b-D_w)}} \tag{6.3}$$

$$D_b = \frac{1}{|S|}\sum_{i=1}^{|S|}\min_{\{j|j\neq i, class(V_i)\neq class(V_j)\}} Dis(V_i,V_j) \tag{6.4}$$

$$D_w = \frac{1}{|S|}\sum_{i=1}^{|S|}\max_{\{j|j\neq i, class(V_i)=class(V_j)\}} Dis(V_i,V_j) \tag{6.5}$$

$$Czekanowski(V_i,V_j) = 1 - \frac{2\sum_{d=1}^{n}\min(V_{id},V_{jd})}{\sum_{d=1}^{n}(V_{id}+V_{jd})} \tag{6.6}$$

where $Dis(V_i,V_j)$ can be any measure used to approximate the distance between two vectors V_i and V_j. Czekanowski distance [40] was used in this method to evaluate the dissimilarity of two vectors because it is calculated

based on the shared portion between two vectors as shown in Equation (6.6). Therefore, its value is bounded in the interval [0,1]. As a result, D_b and D_w values also fall in [0,1] interval and their difference $(D_b - D_w)$ will fall in [-1,1]. This difference was used as the second component in the fitness function after using a logistic function with coefficient -5 to transform it into a value of [0,1] interval. Note that the Czekanowski distance can only be used with non-negative values. Therefore, the constructed features are scaled into the range of [0..1] before applying the formula.

6.2.3 MGPFC Overall Algorithm

Algorithm 7: The pseudo code of MGPFC

```
input  : Training set, Construction ratio r
output : The best constructed features
begin
    m ← r × Nbr_Classes ;
    Randomly initialise individuals, each has m trees;
    while Maximum generations is not reach do
        for i = 1 to Population Size do
            transf_train ← Transform training set based on the constructed
              features in individual i;
            Apply learning algorithm on transf_train to get average accuracy;
            Calculate distance on the normalised transf_train data using
              Equation (6.3);
            Calculate fitness of individual i using Equation (6.1);
        end
        Select parent individuals using tournament;
        Create offspring individuals by applying crossover or mutation to the
          selected parents;
        Place new individuals into the population of the next generation;
    end
    Return the constructed features in the best individual;
end
```

The overall algorithm as shown in Algorithm 7 returns a set of constructed

features for a given training set and a construction ratio r. First of all, the number of constructed features m is determined based on r and the number of classes of the given problem. Then, GP starts by randomly initialising individuals, each having m trees using the predefined function set and the terminal set. Each individual corresponds to a new candidate solution which is m constructed features. These features are used to transform the training set where accuracy and distance will determine the fitness of the corresponding individual (lines 6 to 9). After evaluation, a normal selection and evolutionary process is conducted. This is repeated until the maximum number of generations is reached. Then the set of constructed features in the best individual will be returned as the final solution.

6.3 The Proposed Method: CDFC

6.3.1 Representation

This method aims at constructing multiple features using multi-tree GP representation. Each constructed feature will focus on discriminating instances of one class from the other classes. Similar to MGPFC, the number of constructed features m is equal to the user-given construction ratio r multiplied by the number of classes. However, different from MGPFC, each constructed feature is associated with one class. Figure 6.2 shows an example of a GP individual with $r = 1$ for a three-class problem. In this case, while CF_1 is evolved towards a high-level feature that can distinguish instances of $Class_1$ to other classes, CF_2 and CF_3 focus on $Class_2$ and $Class_3$, respectively.

6.3.2 Class-Dependent Terminal Sets

Since one constructed feature aims at discriminating instances of one class from the other classes, it should be constructed based on features that are relevant to its associated class. In other words, the constructed features of different classes are expected to choose original features from different subsets.

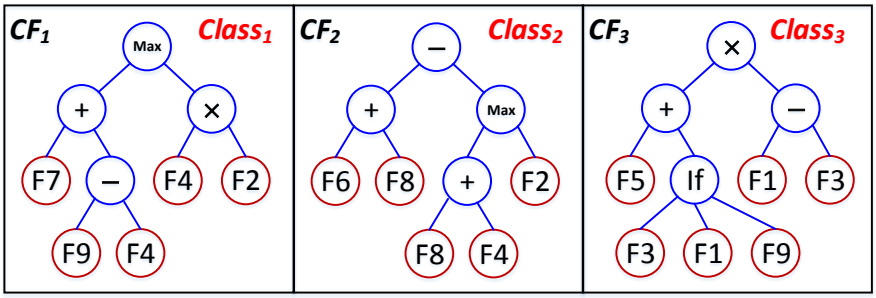


Figure 6.2: Representation of a class-dependent GP individual with construction ratio 1.

Therefore, different trees in an individual will have different terminal sets, each of which comprises of features that are relevant to the corresponding class. Since there might be features important to different classes, it is likely to have overlap between different terminal sets.

A feature f is said to be relevant to discriminating a class c if its values appeared in class c are significantly different from its values in other classes. In CDFC, t-Test is used to measure how relevant a feature f is to class c. First of all, values of f will be divided into two groups, one comprises values belonging to class c and one from other classes. Then, the relevant measure $Rel_{f,c}$ is calculated based on Equation (6.7). $Rel_{f,c}$ is set to 0 if the two groups are not significantly different (i.e. p-value ≥ 0.05). Otherwise, it is equal to the absolute of t-value divided by p-value. Therefore, the larger the value of $Rel_{f,c}$, the more relevant the feature f to discriminating class c.

$$Rel_{f,c} = \begin{cases} 0, & \text{if p-value} \geq 0.05 \\ \frac{|\textbf{t-value}(f_{class=c}, f_{class \neq c})|}{\textbf{p-value}}, & \text{otherwise} \end{cases} \tag{6.7}$$

For each class c, features are ranked by its $Rel_{f,c}$ values. Then half of the top-ranked features will be used to form the terminal set of class c. By doing so, the method not only eliminates irrelevant features but also narrows the search space so that the searching process will be more efficient.

6.3.3 CDFC Fitness Function

Since different constructed features in an individual focus on different classes, they need to be evaluated separately. Therefore, the evaluation time will become very high if CDFC uses the same hybrid fitness function as MGPFC because the computation cost of the wrapper measure is very high. Therefore, a new fitness function is proposed in CDFC using two filter measures that are simple and fast to calculate and can provide a good indicator of data discriminating ability. Equation (6.8) shows the fitness function that maximises two measures combined using a weight α. The size of the GP individual (indSize) is also used here as a pressure for small-tree preference. Therefore, its weight is set to a very small value (10^{-7}) to limit its effect on only cases when two individuals have the same information gain and distance.

$$Fitness = \alpha \cdot AvgIG + (1 - \alpha) \cdot Distance - 10^{-7} \cdot indSize \tag{6.8}$$

The first measure (AvgIG) is the average information gain (IG) measured by mutual information of the constructed features and the binary-class label (i.e. the focused class versus all the other classes). AvgIG is calculated based on Equation (6.9) where f_{max} is the best feature with the highest IG among m constructed features. The f_{max}'s IG is added to bias toward those candidates that have the better f_{max}. IG of feature f is calculated based on unconditional and conditional entropy H as in Equation (6.10), i.e. mutual information.

$$AvgIG = \frac{\sum_{i=1}^{m} IG(f_i, class) + IG(f_{max}, class)}{m + 1} \tag{6.9}$$

$$IG(f, class) = H(class) - H(class|f) \tag{6.10}$$

Although IG is a good measure of feature relevancy, it can only evaluate features individually. Therefore, it cannot show how good the whole set of constructed features in discriminating instances of different classes. Therefore, IG is combined with distance measure to overcome this limitation. CDFC uses the same *Distance* measure [11] as used in MGPFC method, which was described by Equation (6.3).

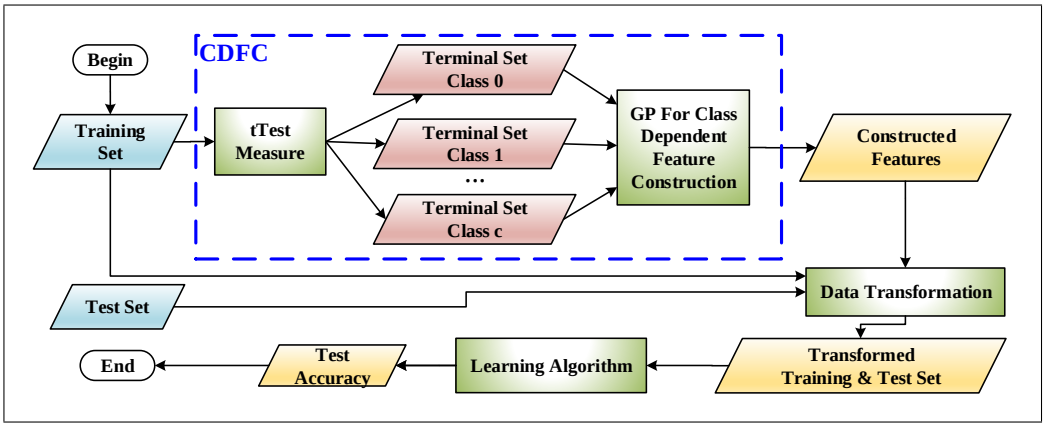


Figure 6.3: CDFC overall system.

6.3.4 CDFC Overall System

Figure 6.3 shows the overall system of CDFC. Based on the training set, CDFC forms one terminal set for one class using the relevance measure in Equation (6.7). These terminal sets are then put into GP for class-dependent feature construction. In each GP individual, the tree corresponding to class i will use the ith terminal set. When the stopping criterion is met, the best individual is used to generate the constructed features. The training and test sets will be transformed based on these features and used to evaluate the performance of CDFC.

6.4 Experiment Design

6.4.1 Datasets

Eight gene expression datasets with thousands of features are used to examine the performance of the proposed method on high-dimensional data. Details about these datasets are shown in Table 1.1 on Page 21. Among the eight datasets, six are binary-class problems (Colon, DLBCL, Leukemia, CNS, Prostate, and Ovarian), one has three classes (Leukemia1), and the last one

has four classes (SRBCT). Note that the last two datasets were not used in the experiment of Chapter 5 because the single feature construction method proposed in Chapter 5 used embedded approach, where GP-tree was used as a binary-class classifier to evaluate the constructed feature.

Before being fed to the feature construction methods, these datasets are discretised in the same way as in Chapter 5 (described in Section 5.3 on Page 155) to reduce noise generated during the data collection in laboratories as suggested in [63].

6.4.2 Experiment Configuration

The performance of MGPFC and CDFC were tested by comparing the discriminating ability of the constructed feature versus the original features. The better method was used to compare with the clustering-based feature construction method (CGPFC) proposed in Chapter 5, which has been shown to produce significantly better results than standard GP for feature construction. It was also compared with another multiple feature construction method using single-tree GP proposed by Neshatian et al. [174], which is called 1TGPFC here for representation convenience. All the comparisons were done based on the classification accuracy of three common learning algorithms including K-nearest neighbour (KNN), Naive Bayes (NB) and Decision Tree (DT).

Due to the small number of instances in each dataset, 10-fold CV is used to generate training and test set, where one is used for testing, and the other 9 for training as shown in Figure 2.3. As GP is a stochastic algorithm, 50 independent runs of each method with 50 different random seeds are executed on each training set. Average of 500 results (50×10) are used for comparisons.

Experiments were runs on PC with Intel Core i7-4770 CPU @ 3.4GHz, running Ubuntu 4.6 and Java 1.7 with a total memory of 8GB. The results of 50 runs from each method were compared using Wilcoxon statistical significance test [241], which is a the non-parametric rank sum test, with a significance level of 0.05.

Table 6.1: Parameter settings

Function set	$+$, $-$, $\times$, max, if	Generations	50
Population Size	#features x β	Crossover Rate	0.8
Initial Population	Ramped Half-and Half	Mutation Rate	0.2
Maximum Tree Depth	8	Elitism Size	1
Selection Method	Tournament Method	Tournament Size	7
Construction ratio r	2	Fitness weighting α	0.8

6.4.3 Parameter Settings

Table 6.1 describes the parameter settings of all GP based methods used in the experiments. The function set comprises of 3 arithmetic operators ($+$, $-$, $\times$), a max function which returns the maximum values from the two inputs and an if function which returns the second argument if the first argument is positive and returns the third argument otherwise. No constant values are used in the terminal set for simplicity. The population size is set proportional to the dimensionality of the problem using a coefficient β, which is set to 3 for the Colon dataset and to 2 for all the others due to memory limit. The construction ratio r used to determine the number of constructed features is experimentally chosen as 2. The fitness weight α in both MGPFC and CDFC is set to 0.8 in order to bias fitness values towards the accuracy (in MGPFC) and information gain measure (in CDFC).

6.5 Results and Discussions

Table 6.2 shows the average test accuracy of KNN, NB and DT using the constructed features obtained from 50 independent runs of CDFC compared with "Full" (i.e. using the original feature set) and MGPFC. For each learning algorithm, the best (B), the mean and standard deviation (M±Std) results are displayed. The best result among the three compared methods on each dataset is highlighted. In addition, the Wilcoxon significance test is applied to the results with a 5% significance level. Its results for KNN, NB and DT

Table 6.2: Results of the constructed features

Dataset	Method	#F	B-KNN	M±Std-KNN	S_1	B-NB	M±Std-NB	S_2	B-DT	M±Std-DT	S_3
Colon (62)	Full	2000	74.29	(+)	–	72.62	(=)	–	74.29	(+)	–
	MGPFC	4	85.47	71.95 ± 5.39	–	85.48	71.23 ± 5.83	–	85.48	71.17 ± 5.25	–
	CDFC	4	88.81	**81.87 ± 3.08**		90.47	**83.52 ± 3.11**		87.38	**78.03 ± 4.00**	
DLBCL (77)	Full	5469	84.46	(–)	–	81.96	(–)	–	80.89	(–)	–
	MGPFC	4	97.32	89.39 ± 3.53	–	97.32	89.18 ± 3.54	–	96.07	87.47 ± 4.32	–
	CDFC	4	98.75	**96.03 ± 1.96**		98.75	**95.25 ± 2.19**		95.00	**90.76 ± 3.01**	
Leukemia (72)	Full	7129	88.57	(–)	–	91.96	(–)	–	91.61	(=)	=
	MGPFC	4	97.32	92.92 ± 2.29	–	96.07	92.77 ± 1.90	–	95.89	91.16 ± 2.50	=
	CDFC	4	98.57	**94.83 ± 1.71**		97.32	**94.07 ± 1.60**		94.82	90.72 ± 2.47	
CNS (60)	Full	7129	56.67	(=)	–	58.33	(=)	–	50.00	(–)	–
	MGPFC	4	71.67	57.97 ± 7.43	–	71.67	58.37 ± 7.46	–	78.33	57.43 ± 7.83	–
	CDFC	4	73.33	**65.10 ± 4.20**		73.33	**66.17 ± 3.75**		68.34	**61.03 ± 4.87**	
Prostate (102)	Full	10509	81.55	(–)	–	60.55	(–)	–	86.18	(=)	–
	MGPFC	4	92.27	86.39 ± 3.14	–	92.18	85.95 ± 2.89	–	91.27	85.32 ± 2.91	–
	CDFC	4	95.18	**92.81 ± 1.59**		96.09	**92.82 ± 1.50**		94.09	**88.04 ± 2.52**	
Ovarian (253)	Full	15154	91.28	(–)	–	90.05	(–)	–	98.41	(–)	–
	MGPFC	4	100.00	99.35 ± 0.42	–	100.00	99.36 ± 0.50	=	100.00	**98.96 ± 0.60**	=
	CDFC	4	100.00	**99.73 ± 0.31**		100.00	**99.55 ± 0.34**		100.00	98.78 ± 0.69	
Leukemia1 (72)	Full	5327	88.57	(=)	–	88.75	(–)	–	94.46	(+)	+
	MGPFC	6	95.89	89.39 ± 3.57	–	95.89	90.86 ± 2.72	–	95.89	90.65 ± 2.36	+
	CDFC	6	97.32	**93.47 ± 1.82**		97.32	**93.07 ± 2.22**		95.89	89.72 ± 2.70	
SRBCT (83)	Full	2308	80.83	(–)	–	97.50	(+)	+	72.36	(–)	–
	MGPFC	8	95.14	87.71 ± 3.31	–	94.03	87.31 ± 3.97	–	91.39	83.12 ± 4.08	–
	CDFC	8	100.00	**95.88 ± 1.94**		100.00	95.08 ± 2.39		94.17	**88.01 ± 3.48**	

are displayed in column S_1, S_2, and S_3, respectively. "+" or "-" indicates that the corresponding method is significantly better or worse than the proposed method CDFC. "=" means they have similar performance. In other words, the more "–", the better the class-dependent feature construction method. The significance test results of MGPFC against Full are displayed in parentheses on the row of Full.

6.5.1 Results of MGPFC

As can be seen from Table 6.2, the number of features constructed by MGPFC was obviously way smaller than the original number of features. However, when using the constructed features, KNN obtained a significantly better accuracy with 4% to 8% higher than using Full on five out of the eight datasets. Its performance was similar to Full on two datasets, namely CNS and Leukemia1, and worse on Colon. However, the best accuracy obtained by KNN on these datasets were 15%, 7% and 9% higher than Full, respectively.

When using the MGPFC constructed features, NB also witnessed a similar pattern as KNN, where its performance was significantly better than Full on five datasets, similar on two and worse on one. The largest improvement was on the Prostate dataset with 25% on average and 32% in the best case.

DT also had a significantly higher accuracy on four datasets with an impressive increase of more than 10% and 7% on the average accuracy of SRBCT and CNS, respectively. The best accuracies that DT achieved on these two datasets are 19% and 28% higher than Full. It had a similar result on other two datasets and worse on the remaining two. On Colon and Leukemia1, its average accuracy was dropped 3.1% and 3.8%, respectively. However, in the best case of these two datasets, it still achieved 11% and 1.4% higher than the Full.

In general, over the 24 pairs of comparisons of MGPFC against Full on eight datasets, the constructed feature set by MGPFC won 14, drew 6 and lost 4. The results indicated that GP could construct an extremely small number of new features that helped the three learning algorithms obtained notably better performance than using the original set of thousands of features.

6.5.2 Results of CDFC

6.5.2.1 CDFC versus Full

All the "-"s appeared in column S_1 of Table 6.2 showed that the constructed features helped KNN achieve significantly higher accuracy than using full feature sets on all the eight datasets. The highest improvement was on the SRBCT dataset with 15% on average and 20% in the best case, reaching 100% accuracy. The modest improvement was still 5% on average and 9% in the best case on Leukemia1. The results showed that the discriminating ability of the constructed features was much higher than the original all features although the number of constructed features was negligible to the original dimensionality.

For NB, the features constructed by CDFC also obtained better performance than Full on almost all datasets. For example, using the 4 constructed features on the Prostate dataset, NB achieved 32% higher accuracy than using the whole 10,509 features. Similarly, the improvement on Colon and DLBCL was 11% and 14% on average with 18% and 17% in the best case, respectively. Only on SRBCT, CDFC had about 2.4% lower accuracy than Full. However, the best accuracy achieved by CDFC was 100% which was 2.5% higher than the Full accuracy.

Compared with using Full, DT using features constructed by CDFC also had significantly better performance on six datasets, similar on one and worse on the remaining one. An improvement of at least 10% on average accuracy was achieved on three datasets, namely SRBCT, CNS and DLBCL, with the best accuracy improved from 15% to 22%. Only on Leukemia1, CDFC obtained about 5% lower average accuracy than Full. However, the best result was still better than Full.

In general, over the 24 comparisons with Full using the three learning algorithms on eight datasets, CDFC won 21, drew 1 and lost 2 in terms of average accuracy. However, in term of the best accuracy, CDFC outperformed Full in all 24 cases except for the tie result of DT on Leukemia1. Results

showed that CDFC could construct a very small number of features with a high discriminating ability, which can generalise well to the three learning algorithms in most cases.

6.5.2.2 CDFC versus MGPFC

As shown in Table 6.2, although both methods constructed the same number of features for each dataset (since the construction ratio was set to 2 for both methods), KNN using features constructed by CDFC achieved significantly better performance than using those of MGPFC on all datasets. The highest improvement of 10% on average was found on Colon, where MGPFC failed to maintain the Full accuracy. The results showed that using terminal sets comprising of features that are relevant to a specific class, CDFC achieved much better results than MGPFC, allowing it to obtain the best KNN accuracy on all datasets.

Similarly, NB using features constructed by CDFC achieved significantly better accuracy on 7 datasets than using those constructed by MGPFC. Among these datasets, CDFC further improved the performance of MGPFC from 6% to 12% on 5 datasets. Only on Ovarian, CDFC obtained a similar accuracy as MGPFC.

For DT, features constructed by CDFC obtained significantly better performance than those of MGPFC on five datasets, namely Colon, DLBCL, CNS, Prostate and SRBCT with a further improvement of 3% to 7%. For the remaining three datasets, CDFC had similar performance as MGPFC on two and worse on one.

In general, compared with MGPFC using the three learning algorithms, CDFC won 20, drew 3 and lost 1 out of the 24 comparisons.

Comparisons between the three learning algorithms revealed an interesting phenomenon. Recall that the features constructed by both MGPFC and CDFC were optimised towards DT performance by using either DT accuracy (MGPFC) or information gain (CDFC) in their fitness functions. However, the results showed that the improvement of DT performance was not as

much as KNN and NB, making its accuracy usually lower than the other two learning algorithms. This may relate to the characteristics of DT where classification decision highly depends on the split values inside the internal nodes of the DT trees. These values are learned based on the training data, which may not be generalised well on the unseen test data of these datasets due to their skew distributions with many outliers as discussed in Chapter 5.

Another interesting observation from the results of CDFC and MGPFC was that although DT used IG as its base measure, it had less power than the average IG measure in evaluating GP individual with multiple constructed features. This finding is contradictory with common practice where wrapper approaches are preferred to filter ones because they usually produce better classification performance. However, the results showed that CDFC obtained significantly higher accuracy than MGPFC in almost all cases. A closer investigation on the evolutionary process of MGPFC showed that in a set of multiple constructed features that gave very good DT accuracy, there might exist very bad or even constant-value features. The reason is that DT does not use all features in building the DT tree. Therefore, the returned classification accuracy does not reflect the goodness of all constructed features. On the other hand, the average IG shows the average relevance level of all the constructed features of the individual. Therefore, using average IG as a fitness measure, GP can better evaluate individuals and choose those that comprise more good features to create better offspring in its evolutionary process.

In summary, features constructed by CDFC had significantly better performance than those constructed by MGPFC on 20 cases, similar on 3 and worse on 1. Note that results of CDFC had smaller standard deviation than MGPFC in almost all cases. This indicated that by constraining the terminal sets to class relevant features, the performance of CDFC was better and more stable than MGPFC.

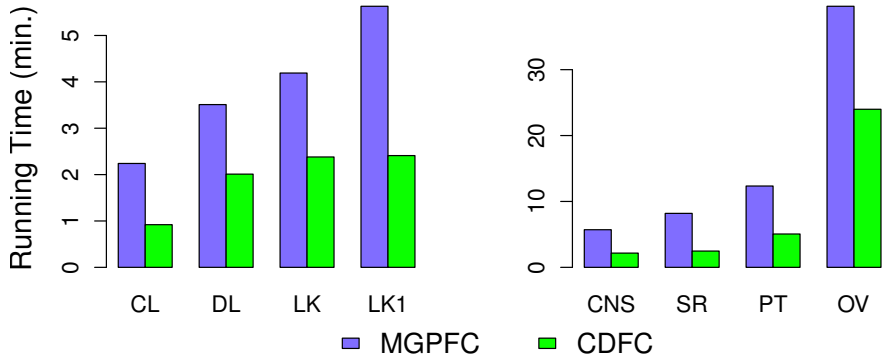


Figure 6.4: Computation time of CDFC versus MGPFC.

6.5.3 Computation Time

Figure 6.4 shows the average running time to complete a single run for MGPFC and CDFC. Results in the figure showed that the CDFC running time was less than half of MGPFC on five out of the eight datasets, and less than 60% on the remaining three.

Note that both methods used the same population size and the maximum number of generations. In other words, they had the same number of evaluations. However, CDFC running time was much shorter than MGPFC on all datasets. The main reason behind this reduction is the computation time of the fitness evaluations. While the CDFC fitness function comprises of two filter measures, distance and IG, MGPFC combines distance with a wrapper measure which is an average accuracy of DT over 3-fold cross-validation within the training set. Although IG is the base measure of DT, computation time of the average IG of each constructed feature is still much lower. Therefore, the running time of CDFC was much faster than MGPFC. This again confirmed the efficiency of filter measures.

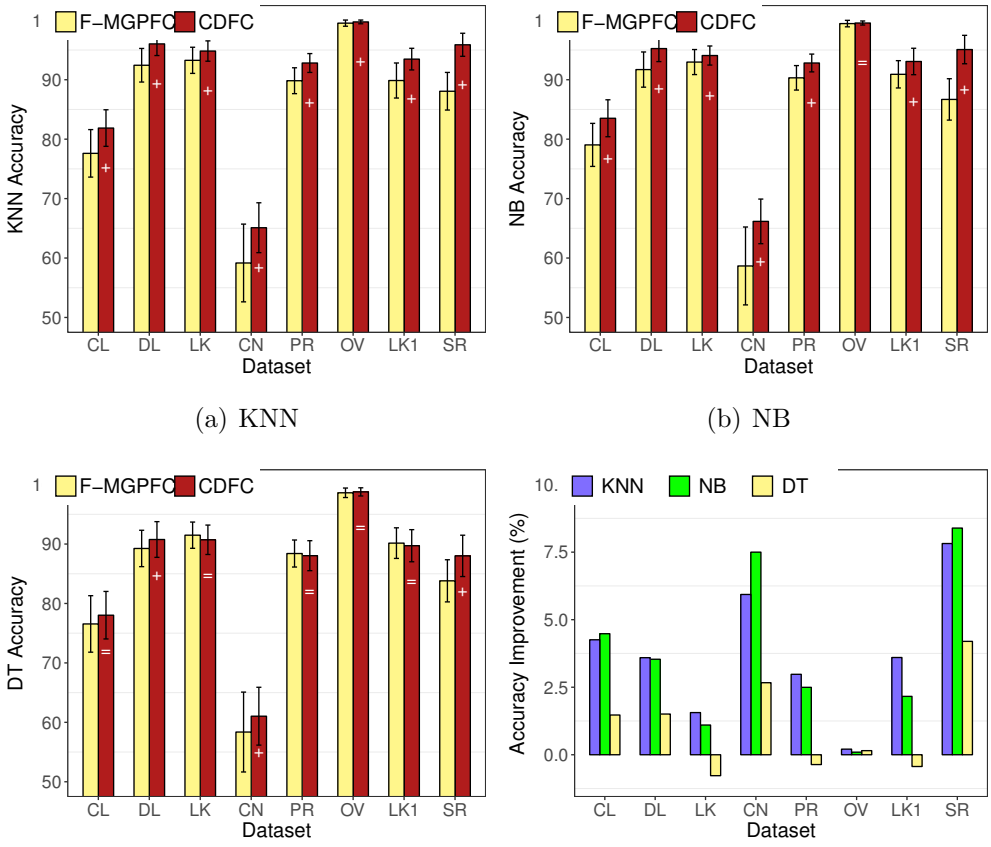
6.6 Further Analysis

6.6.1 Class-Dependent versus Class-Independent FC

Since CDFC differs from MGPFC not only in the class-dependent feature construction strategy but also in the fitness function, the superior performance of CDFC over MGPFC might be contributed by either components or both of them. Therefore, the effectiveness of the proposed class-dependent feature construction strategy was confirmed by another set of experiments where CDFC is compared with *F-MGPFC*, which is the same method as MGPFC except that its fitness function uses the distance measure combined with the average IG by the same fitness weight as CDFC.

The average results over the 50 runs of F-MGPFC compared with CDFC are shown in Figure 6.5. The first three sub-figures present the average accuracy of KNN, NB and DT of each method. In these figures, each group of bars shows the results of the features constructed by F-MGPFC and CDFC on each dataset. On the CDFC bars, results of the Wilcoxon significance test with a 5% significance level comparing CDFC against CGPFC are displayed. "+" and "−" mean that CDFC is significantly better or worse than CGPFC. "=" means that they are similar. The last subfigure shows the average accuracy improvement of each learning algorithm on each dataset.

As can be seen in Figure 6.5(a), using CDFC constructed features, KNN obtained significantly higher average accuracies than using F-MGPFC constructed features on all datasets with the biggest gap of 7.9% on SRBCT which is a four-class problem. A similar pattern can be seen for NB in Figure 6.5(b) with a significant improvement made on seven out of the eight datasets. SRBCT was also the one that NB obtained the highest average improvement of more than 8% on average. For DT, significantly higher accuracies were found on three datasets and the remaining five had similar performance as F-MGPFC. The compared results of different learning algorithms in Figure 6.5(d) showed that among the three algorithms, KNN had the highest improve-

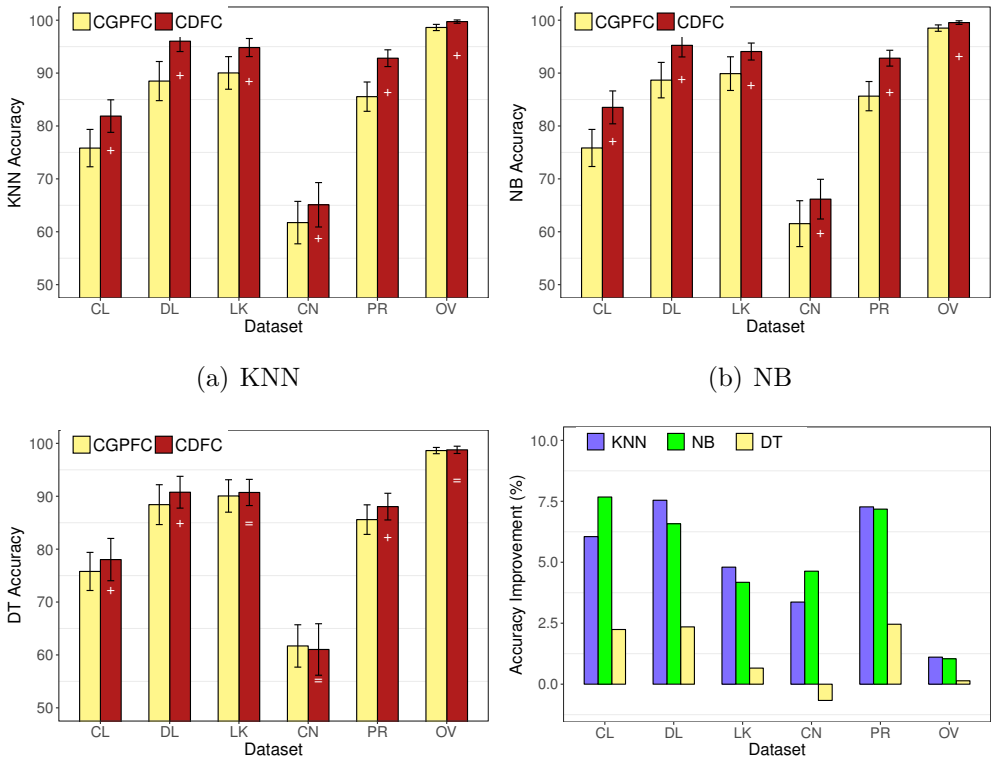


(a) KNN (b) NB

(c) DT (d) Average accuracy Improvement

Figure 6.5: Results of class-dependent (CDFC) versus class-independent (F-MGPFC) feature construction.

ments on five datasets and NB on the remaining three. In general, features constructed by CDFC helped the three learning algorithms either obtain a significantly better or similar classification performance on all datasets. This indicated that by constructing features from relevant features to one class, thus narrowing the GP search space, and evaluating each constructed feature against the corresponding class only, CDFC could construct features with better discriminating ability.



(a) KNN (b) NB

(c) DT (d) Accuracy Improvement

Figure 6.6: Results of multiple feature construction (CDFC) versus single feature construction (CGPFC)

6.6.2 Multiple versus Single Feature Construction

This section will check the hypothesis that was stated at the beginning of the chapter that multiple constructed features may better represent the original problem than a single constructed feature. Since the clustering-based single feature construction method (CGPFC) developed in Chapter 5 has shown to have better performance than standard GP, the performance of the four features constructed by CDFC will be compared with the single feature constructed by CGPFC.

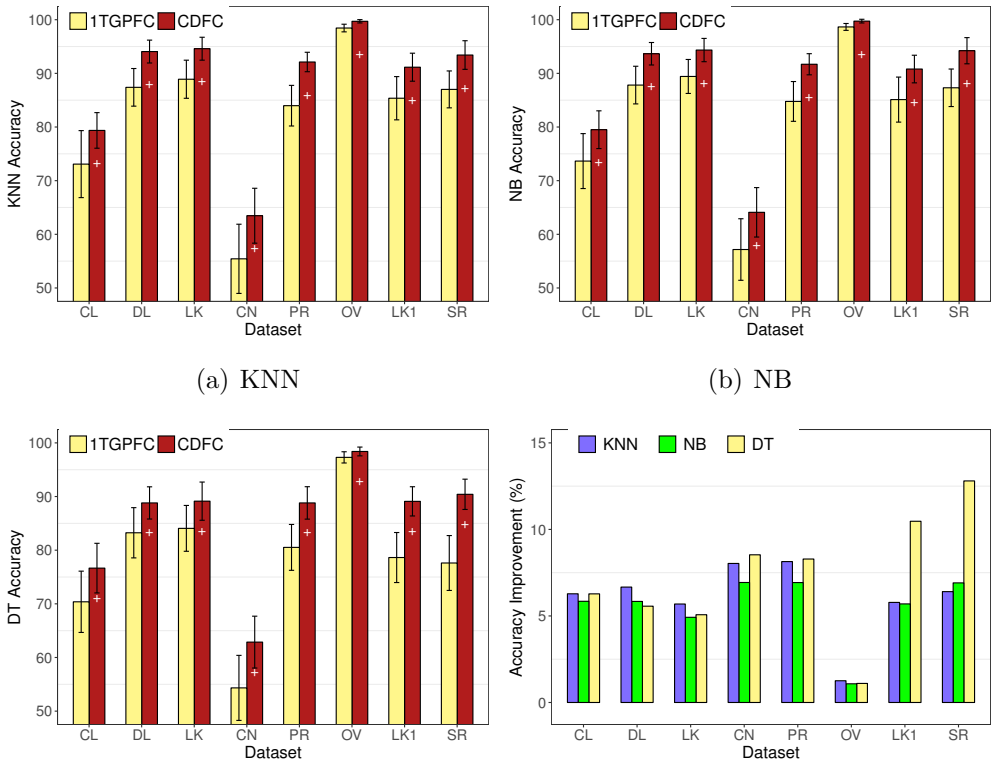
Figure 6.6 shows the compared results of CDFC and CGPFC over the 50 runs on six binary-class datasets. Since CGPFC was designed to construct

features for binary-class problems only, the two multiple-class datasets, namely Leukemia1 and SRBCT, are left out in this comparison.

Results in Figure 6.6(a) showed that the CDFC constructed features obtained a significantly higher KNN accuracy than CGPFC on all datasets with the largest gap of 7.5% on DLBCL. Similar pattern was observed in Figure 6.6(b) for NB with an improvement made on all the datasets. For DT, significantly higher accuracies were found on 3 datasets and similar accuracies were found on the remaining three. In general, compared with the single feature constructed by CGPFC, the four features constructed by CDFC helped the three learning algorithms either obtained a significantly better or similar classification performance on all datasets. The compared results of different learning algorithms in Figure 6.6(d) showed that among the three algorithms, KNN had the highest improvements on four out of the six datasets. Furthermore, as can be seen in the first three subfigures of Figure 6.6, the error bars of CDFC were smaller than the corresponding error bars of CGPFC in almost all cases. This indicated that using the CDFC constructed features, the three learning algorithms obtained more stable results than using the one constructed by CGPFC.

6.6.3 Multi-Tree GP versus Single-Tree GP for Multiple Feature Construction

This section will compare the performance of CDFC with the method proposed by Neshatian et al. [174] where single-tree GP was used to construct multiple features. It is named 1TGPFC here for presentation convenience. In this method, GP was run multiple times, and each time constructed one feature focusing on discriminating instances from one class to other classes. Therefore, two features will be constructed for a binary-class problem. 1TGPFC also follows a filter approach. We ran 1TGPFC with the same settings for GP parameters as in CDFC. In this set of experiments, CDFC was run with the construction ratio 1 to construct the same number of features as 1TGPFC.



(a) KNN (b) NB

(c) DT (d) Accuracy Improvement

Figure 6.7: Results of multiple-tree GP (CDFC) versus single-tree GP (1TGPFC) for multiple feature construction.

The average results over the 50 runs of both methods are displayed in Figure 6.7.

Results from Figure 6.7 showed that using the CDFC constructed features, all the three learning algorithms obtained significantly higher accuracies than using the ones created by 1TGPFC on all datasets. KNN obtained the largest increase of 8.1% average accuracy on Prostate, NB had 7.2% biggest gap on CNS, and DT achieved 13% increase on SRBCT. Comparisons between the three learning algorithms in Figure 6.7(d) showed that KNN had the highest improvement on four datasets. On the other four datasets, DT had the largest difference of 10.5% and 13% on Leukemia1 and SRBCT, which

are the multiple-class problems.

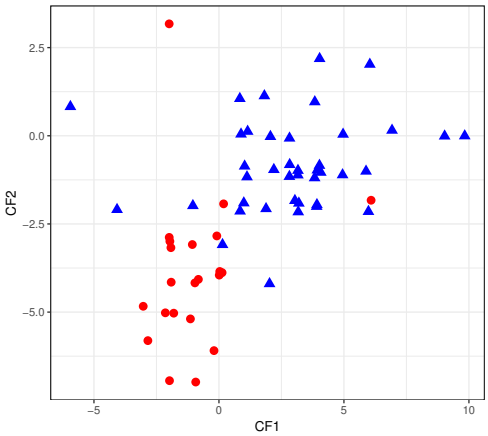
Although 1TGPFC used an impurity measure in its fitness function to minimise the impurity of the interval including constructed feature values of the focused class, its created features did not perform as well as expected, leading to significantly worse results than CDFC. This result may be related to the fact that by using a multi-tree representation to construct multiple features simultaneously, CDFC can evaluate the constructed features as a whole during the evolutionary process. This enabled it to optimise the discriminating power of the whole feature set, taking into account possible interactions between the constructed features. This ability is obviously impossible when running GP separately to construct one feature at a time as in 1TGPFC.

6.6.4 Constructed Features

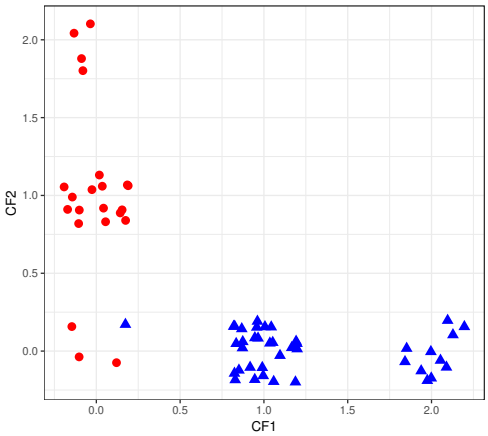
This section will confirm this hypothesis by having a closer look at the features constructed by both the CDFC and 1TGPFC methods. A fair comparison is made by choosing the set of constructed features that had the worst performance among 50 results from 50 runs obtained by each method on the first pair of training and test folds. The reason for comparing the worst results of both method is that the difference between the best results is not large enough to be visually distinguished. The transformed data of seven datasets (excluding SRBCT with more than 3 constructed features) using the constructed features by both methods are plotted in Figures 6.8 to 6.10.

As can be seen from these figures, the constructed features by both methods were quite good in grouping instances of different classes into separate clouds. However, as can be seen from the right column of each figure, the data clouds produced by CDFC were more compact and separated from each other than those created by 1TGPFC, which are shown in the left column of the figures. Therefore, with the new representation obtained from CDFC, it would be much easier for the three classification algorithms to find a model that achieved higher accuracies.

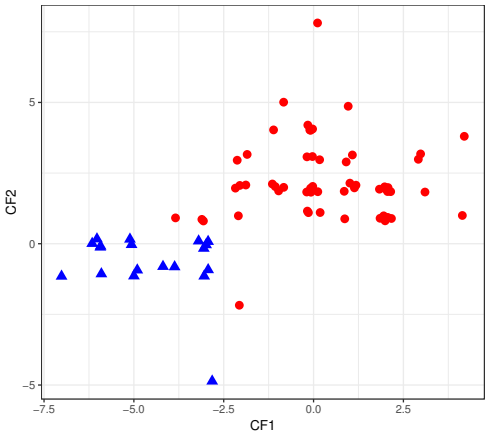
Since both methods constructed features focusing on discriminating in-



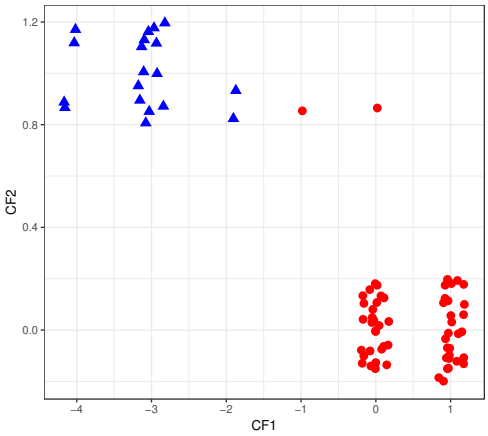
(a) 1TGPFC on Colon



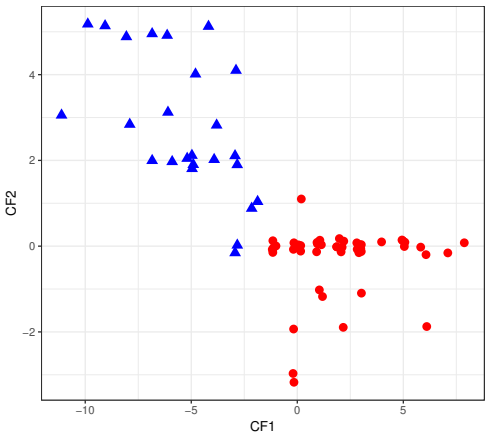
(b) CDFC on Colon



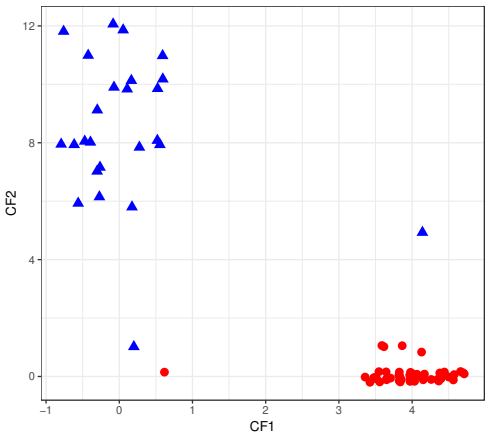
(c) 1TGPFC on DLBCL



(d) CDFC on DLBCL

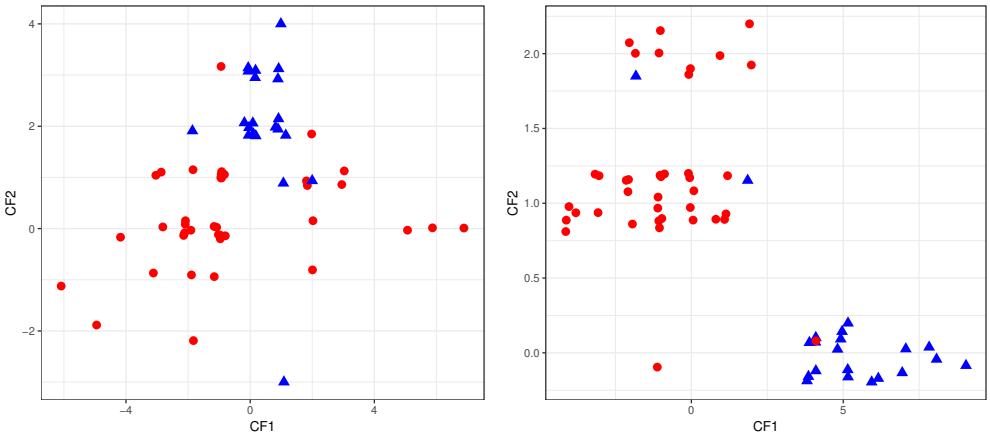


(e) 1TGPFC on Leukemia

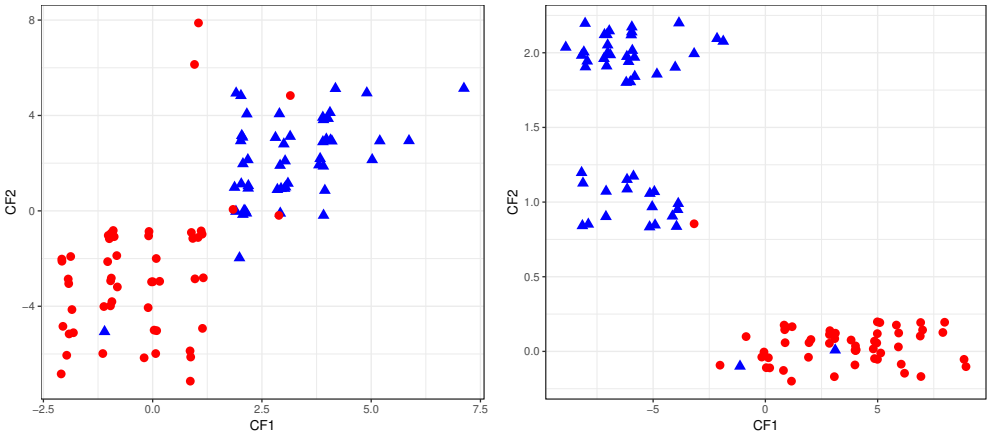


(f) CDFC on Leukemia

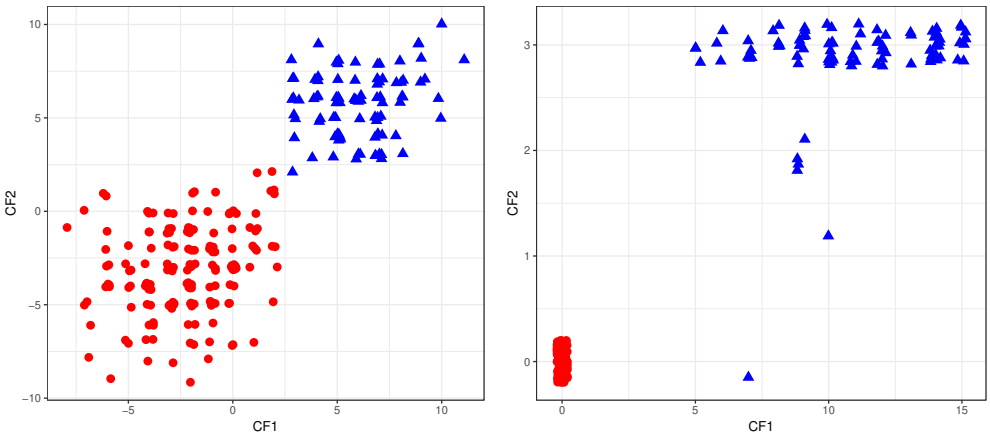
Figure 6.8: Constructed features on Colon, DLBCL and Leukemia.



(a) 1TGPFC on CNS (b) CDFC on CNS

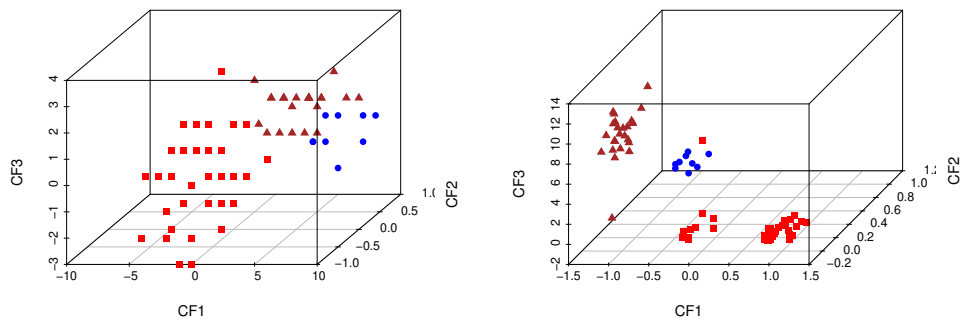


(c) 1TGPFC on Prostate (d) CDFC on Prostate



(e) 1TGPFC on Ovarian (f) CDFC on Ovarian

Figure 6.9: Constructed features on CNS, Prostate and Ovarian.



(a) 1TGPFC on Leukemia1 (b) CDFC on Leukemia1

Figure 6.10: Constructed features on Leukemia1.

stances of one class from the others, contributions to the superior results of CDFC mainly came from its fitness function. While both methods used an entropy-based measure to minimise the impurity of the constructed feature values within each class, CDFC incorporated an additional distance measure to evaluate the whole set of constructed features. The aim of the distance measure is to maximise the distance between instances of different classes and minimise the distance between instances of the same class. This is the reason why the data clouds produced by CDFC are more compact and scattered far away from each other.

6.6.5 Comparison with Feature Selection Results

The results of CDFC have shown that feature construction is a promising approach to dimensionality reduction on high-dimensional data. This section will compare the results of CDFC with the best results obtained by feature selection methods in Chapters 3 and 4 using all the four datasets that have been used in all the experiments of feature selection and feature construction methods in this thesis, namely DLBCL, Prostate, Leukemia1 and SRBCT.

As shown in Table 4.6, the best accuracy obtained by all the proposed feature selection methods on DLBCL is 93.72% using 1417 selected features. On the other hand, the results of the 4 features constructed by CDFC on this dataset obtained 96.03% as shown in Table 6.2. Similarly, results of the 4 features constructed from Prostate obtained 1% higher accuracy than the best feature subset of 65 features selected by PPSO. On Leukemia1, CDFC could construct 6 features to achieve 2% higher accuracy than 32 features selected by PSO-CLS. Only on SRBCT, CDFC constructed 8 features that obtained 4% lower average accuracy than 60 features selected by PSO-CLS. However, in the best case, the 8 constructed features still achieved 100% accuracy as PSO-CLS. The results indicated that although the set of constructed feature is extremely small, its performance is comparable to tens or even thousands of selected features. The intense reduction in dimensionality obtained by the proposed feature construction methods not only significantly speed up the classification methods but also substantially reduce the storage space for these datasets. Once the constructed features are learnt, the process of transforming features is extremely fast. This opens an opportunity for high-dimensional data to be processed on online systems.

6.7 Chapter Summary

The goal of this chapter was to propose a new multiple feature construction approach using GP which can produce a very small number of high-level features to improve the performance of common learning algorithms on high-dimensional data. Two methods were proposed using multi-tree representation, one for class-independent feature construction (MGPFC) and one for class-dependent (CDFC). In MGPFC, a hybrid evaluation method was used to synthesise the strength of wrapper and filter approach. In CDFC, different terminal sets were generated to construct different class-dependent features. The t-Test is used as a relevance measure to choose features that are relevant to a given class. A new filter-based evaluation method was also proposed in

CDFC to better evaluate a set of constructed features and to speed up the evaluation process as well.

Performance of the MGPFC and CDFC constructed features has shown to be significantly better than the original set using KNN, NB and DT. Results on the eight high-dimensional datasets also show that CDFC is not only more effective but also more efficient than MGPFC. The proposed strategies in CDFC demonstrate that by forming the GP terminal set with class-relevant features and a good fitness function, the proposed filter method can achieve better performance than the hybrid approach MGPFC.

Results of CDFC has also shown that it outperforms the single feature construction method CGPFC proposed in Chapter 5 and the multiple feature construction method using single tree GP (1TGPFC) proposed in [174]. Further investigation and analysis have shown that multiple constructed features better represent the original high-dimensional data than a single one. Compared with the single-tree GP approach to multiple feature construction, multi-tree GP has the benefit of evaluating all constructed features in a single run. Comparison with the proposed feature selection methods in previous chapters has shown that CDFC can construct an extremely small number of features that obtained comparable or even better results than the selected features.

7

Conclusions

This thesis focuses on PSO and GP for high dimensionality reduction in classification. The overall goal was to investigate and improve the performance of PSO for feature selection and GP for feature construction on high-dimensional data in terms of feature subset size, classification accuracy and computation time. This goal has been successfully achieved by developing a number of new PSO based feature selection and GP based feature construction methods to automatically evolve sets of selected or constructed features that are much smaller than the original feature set while achieving classification performance better than or similar to the original one. The proposed methods were examined and compared with existing methods on a range of high-dimensional classification problems of varying difficulties. The results show that the newly proposed methods can be effectively and efficiently used for feature selection and construction in classification on high-dimensional data.

The remainder of this chapter provides conclusions for each of the research objectives of this thesis, highlights the findings in each contribution, and outlines potential research directions for future work.

7.1 Achieved Objectives

This thesis has led to the following:

- The proposal of two new PSO based feature selection methods with two new local search strategies to enhance PSO performance in feature selection on high-dimensional data by balancing its global and local search capability. Both local search strategies are applied to *pbest* in order to intensify the search in the area surrounding this newly found solution: while the first one employs a random flipping mechanism to produce a new candidate solution, the second one relies on general knowledge in feature selection to remove redundant features and add more relevant features. A new hybrid evaluation method is proposed to synthesise the strength of both wrapper and filter methods. To speed up the running time of the local search process, a fast fitness evaluation mechanism is also proposed. The results show that the proposed PSO based feature selection algorithms can reduce the number of features by one to two orders of magnitude. The feature subsets selected by the proposed methods either outperform or achieve similar results to using all features, two traditional feature selection methods, standard PSO based feature selection method, and another PSO based feature selection method that uses resetting *gbest* strategy.

- The proposal of the first PSO based feature selection via discretisation method, which integrates multivariate discretisation and feature selection in a single stage. Two new representations are proposed to either directly evolve a cut-point for each feature or choose one from multiple potentially good cut-points. The results of the conducted experiments show that the proposed approach selects a much smaller number of features and achieves better performance than using all original features, four two-stage approaches with traditional feature selection methods, one traditional feature selection via discretisation method, and standard

PSO based feature selection method.

- The proposal of the first clustering-based feature construction method using single-tree GP for high-dimensional data. A new feature clustering method is proposed to cluster similar or redundant features into the same group from which only the best feature is fed into GP to construct a new feature. The experiment results show that the proposed method can construct a single feature that can help common learning algorithms improve classification performance over using all the original features and the one constructed from the whole feature set. Investigation on five different created feature sets from the single GP tree shows that the combination of the single constructed feature and the selected ones achieved the best performance among the five.

- The proposal of two new multiple feature construction methods using multi-tree GP, the first one constructs class-independent features and the second one constructs class-dependent features. A new GP representation is proposed to construct a small number of features that is proportional to the number of classes in the dataset. New fitness functions are proposed to evaluate the whole set of constructed features. By constructing features that focus on distinguishing instances in one class from the others using features that are relevant to the focussed class, the second feature construction method achieves a significantly better classification accuracy than using all features, the first proposed method without considering the class dependency, the single feature construction method proposed in Chapter 5, and a multiple feature construction method in the literature.

7.2 Main Conclusions

Overall, this thesis finds that PSO and GP can be used effectively for feature selection and construction, respectively, on high-dimensional classification

problems.

This section discusses the main conclusions for the four research objectives drawn from the four contribution chapters (Chapter 3 to Chapter 6)

7.2.1 Using Local Search in PSO for Feature Selection

Chapter 3 proposes two PSO based feature selection methods using two new local search strategies to intensively search for better solution surrounding the newly found *pbest* during the evolutionary process of PSO.

7.2.1.1 Local Search Effectiveness

PSO is well-known with its global search capability. However, it is usually difficult for PSO to fine-tune its solution, especially in the search space with thousands of features as in high-dimensional classification problems. This thesis finds that a local search strategy guided by general knowledge in feature selection can significantly improve the performance of PSO for feature selection on these problems.

The randomly flipping mechanism (PSO-RLS) helps PSO find much smaller feature subsets while maintaining the classification performance of the selected features. Using a correlation-based measure to guide the local search, PSO-CLS further helps PSO achieve much better classification accuracies with 15 to 85 times fewer features in most cases. The superior performance of PSO-CLS to PSO-RLS is contributed by two components, an informative search and a better evaluation method, which are essential for an effective feature selection algorithm. By measuring the relevance and redundancy of the features in the current *pbest*, the correlation-based local search can remove redundant features and add more relevant ones to the feature subsets, allowing it to find a much smaller and better feature subset for the given *pbest*. The second contribution is the hybrid evaluation method, which will be discussed at the end of this subsection.

7.2.1.2 Local Search Efficiency

One of the main concerns of combining local search into PSO is the high computation time that local search added into an already expensive process of PSO when applying to high-dimensional data, especially in wrapper approaches. However, this thesis finds that a fast fitness evaluation strategy can be used to reduce the evaluation time during the local search process while maintaining its accurate calculation. By using this strategy, the proposed methods can evaluate hundreds of thousands of solutions more in a reasonable time. Furthermore, when using local search to remove redundant features, the number of features was greatly reduced, leading to a shorter evaluation time. As a results, the proposed method uses a similar running time as standard PSO. In addition, the effectiveness of the local search helps PSO find better *pbest* with smaller and better feature subsets, which can direct particles to more fruitful areas with smaller dimensionality in the search space. This enables PSO to reach better solutions in a shorter time. In fact, PSO-CLS has a similar computation time to standard PSO on five datasets and a much smaller running time than PSO-RLS on eight out of the ten datasets.

The frequency of applying local search is also an important factor influencing the running time. It is found that a high frequency requires a longer running time but does not necessarily lead to a better solution. It is the informative search and a good evaluation method that a local search needs to better guide the evolution in fine-tuning a given solution.

7.2.1.3 Hybrid versus Wrapper Evaluation

This thesis finds that a clever combination of a wrapper and a filter measure can create a better evaluation procedure for feature selection. A combination of KNN and distance measure is proposed to evaluate the fitness of a feature subset. While the classification accuracy can measure how well the performance is, the distance measure can approximate how far these features can separate instances of different classes and unite those of the same class.

By combining these two measures, PSO-CLS can better evaluate feature subsets or particles. It will then select a better particle to lead the search, allowing the algorithm to explore more fruitful areas of the huge search space. Therefore, smaller feature subsets with better classification performance are found by PSO-CLS.

A PSO based feature selection method using a wrapper approach is already expensive in terms of the computational cost, especially on high-dimensional data. Therefore, adding another filter measure may make it more expensive. However, it is found that by choosing a classification algorithm and a filter measure that are based on the same calculation which in this case is a distance measure, the proposed hybrid evaluation method requires no or very little increase in the computation time.

7.2.2 PSO Based Feature Selection Via Discretisation

The *first* discretisation-based feature selection using PSO is proposed in this thesis (Chapter 4). It is found that using PSO to perform multivariate discretisation and feature selection at the same time achieves smaller feature subsets with better discriminating ability than the two-stage approach where discretisation is done before feature selection. By evaluating all the cut-points of the selected features at the same time, the proposed methods can consider possible interactions between the discrete version of the selected features which is not obtainable in the two-stage approach. Therefore, better cut-points can be chosen to minimise the information lost during discretisation, which helps PSO to choose better feature subsets.

Chapter 4 further confirms the findings of the hybrid evaluation method concluded from Chapter 3.

7.2.2.1 Representation

This thesis finds that using PSO to choose cut-points from a set of potentially good ones (PPSO) can obtain smaller feature subsets with similar or better classification performance than using PSO to directly evolve a cut-point from

the range of feature values (EPSO). The main reason is that the search space of discretisation and feature selection on high-dimensional data in EPSO is much larger than PPSO. By predefining some potentially good cut-points, PPSO significantly narrows the PSO search space into a much smaller and more fruitful space. This strategy helps PSO achieve better performance.

7.2.2.2 Initialisation

It is found that in such a large search space of feature selection on high-dimensional data, the initialisation mechanism plays an important role in helping PSO find better discretisation and feature selection solutions in a shorter time. By using a predefined subset size and the potentially best cut-point of each feature, the proposed initialisation method helps PSO reach better solutions in early stages, which can then benefit the whole evolutionary process. This thesis also finds that cut-points created by the minimum description length principle are good starting points for PSO to search for better solutions.

7.2.3 Cluster-Based Feature Construction Using Single-Tree GP

This thesis proposes the *first* clustering-based feature construction method using single-tree GP for high-dimensional data. It is found that using feature clustering to reduce redundant features in the terminal set can significantly improve GP performance in feature construction on high-dimensional problems. By choosing a representative feature from each group of similar features, GP can effectively reduce its search space. This enables GP to focus on choosing appropriate operators and features to construct a better one. It is also found that by eliminating redundant features, GP selects a much smaller number of original features and constructs a new feature with a higher discriminating ability. The number of selected features is reduced by half or more in most cases.

7.2.3.1 Feature Clustering

This thesis finds that using a predefined redundancy threshold to group similar features can automatically determine the number of feature clusters, which is not a trivial task, especially on high-dimensional data with thousands of features. It is found that by guaranteeing features in the same cluster to have their correlation level higher than the predefined threshold, the proposed feature clustering algorithm effectively and efficiently creates compact and separated clusters in most cases.

7.2.3.2 Combinations of Constructed and Selected Features

This thesis investigates the performance of five different combinations of selected and constructed features from a single GP tree. It is found that among the five feature sets, the combination of the single constructed feature and the selected ones can significantly improve the performance of KNN and NB.

7.2.3.3 Generalisability

Overfitting is a common issue in GP for feature construction, especially when the number of instances is small. This thesis finds that GP can cope with small sample size and features with skew distribution to obtain a good generalisation as long as the given training data has enough information to represent the distribution of the unseen data.

7.2.4 Class-Dependent Feature Construction Using Multi-Tree GP

This thesis proposes a new class-dependent feature construction method using multi-tree GP for high-dimensional data. It is found that the class-dependent constructed features achieve better performance than the class-independent ones. By forming the terminal set with class-relevant features, the proposed

method can narrow the search space and hence improve GP performance for feature construction.

7.2.4.1 Multi-tree Representation

This thesis finds that using multi-tree GP representation to construct multiple features can achieve a better set of constructed features than running single-tree GP multiple times. The main reason is that constructing multiple features in a single run allows GP to evaluate the whole new feature set at once taking into account possible interactions between the constructed features, which is not obtainable when constructing them in separate runs.

7.2.4.2 Wrapper and Filter Approaches

It is found that evaluating a feature set using a classification algorithm where there is a built-in feature selection process, such as DT, may not reflect the goodness of all features in the set. Very bad or even features with constant values may exist in a very fit individual. Therefore, using this measure may mislead GP to choose these trees for breeding, resulting in performance deterioration. By replacing DT classification accuracy with a corresponding filter measure, e.g. the average Information Gain of each feature as in this thesis, the fitness of an individual can reflect the goodness of every single constructed feature. Switching from a wrapper to a filter approach also significantly reduces the evaluation cost, hence the running time of the proposed method.

This thesis also finds that by combining a distance measure in the evaluation method to maximise the distance between instances of different classes and minimise the distance between instances of the same class, the constructed features obtain better discriminating ability.

7.3 Future Work

This section highlights key areas of future work.

7.3.1 New PSO Representation for Higher Dimensionality

This thesis focuses mainly on classification problems with thousands to tens of thousands of features. However, datasets with millions of features become more and more popular in many domains, and the number of features can grow even larger with the trend of big data. In current PSO, the particle representation is fixed-length based on the number of features. Therefore, the high computation cost and a large memory requirement may hinder PSO from working on these big datasets. A smaller or dynamic representation is needed to reduce the resource consumption, enabling PSO to work with even larger scale problems.

7.3.2 New GP Representation for Feature Construction

Representation defines the search space or scope of the problem. The number of features constructed by the proposed methods in this thesis is predefined and fixed during the evolutionary process. This may prevent GP to explore solutions with bigger or smaller numbers of constructed features. A dynamic representation which enables GP to automatically evolve any number of constructed features could be helpful in obtaining better feature sets for high-dimensional data.

7.3.3 PSO Based Feature Selection via Multi-Interval Discretisation

This thesis proposes the *first* discretisation-based feature selection using PSO, which has shown promise in simultaneous discretisation and feature selection. The capability of PSO for multivariate discretisation has not been fully investigated with binary discretisation, where only one cut-point is evolved for each feature. New representation for multi-interval discretisation

can be proposed to evolve multiple cut-points for each feature. In other words, features are discretised into multiple intervals or discrete values, which may be required for a better representation. With this ability, PSO can achieve even better feature subsets.

7.3.4 Feature Evaluation Methods for Feature Selection and Construction

One of the main problems of applying EC techniques, such as PSO and GP, to feature selection and construction on high-dimensional data is its high computational cost, which comes mainly from the evaluation process, especially when the number of instances is large. On the other hand, overfitting can become serious when the number of instances used to evaluate candidate solutions is small. Therefore, different strategies are needed to improve the performance of feature evaluation methods, which encourages more EC applications.

7.3.4.1 Hybrid Approaches

This thesis has shown combining different wrapper and filter measures can better evaluate the discriminating power of a feature set in an acceptable running time. More effective and efficient evaluation methods should be investigated on these high-dimensional problems.

7.3.4.2 Instance Selection or Creation

While a large number of instances can affect the efficiency of the evaluation procedure, a small number of instances can affect its effectiveness. Therefore, an adaptive strategy is needed to automatically adjust this number so that the evaluation method can maximise its performance on high-dimensional or large-scale datasets. Representative instances could be chosen to speed up the evaluation function while new instances or prototype could be created to make the training data more representative.

7.3.5 Multi-objective Approach to Feature Selection and Construction

This thesis only adopts single-objective approaches in developing feature selection and construction methods. As can be seen from this thesis, high-dimensional data usually contains a large number of irrelevant or redundant features. Removing these features helps improve the discriminating ability of the feature set. However, at some stage, further removing features may degrade its ability in distinguishing instances of different classes. Therefore, an appropriate multi-objective approach to feature selection and construction on high-dimensional data could provide multiple informed solutions showing the trade-off between the number of features and the classification accuracy. more and better solutions.

7.3.6 Integration of PSO and GP for Dimensionality Reduction

PSO and GP have shown promise in feature selection and construction for high-dimensional classification. New approaches of integrating these techniques should be investigated to create an automatic dimensionality reduction method, where feature selection or construction or both can be automatically chosen and applied to produce the best representation for a given dataset.

7.3.7 Transfer Learning

As can be seen from this thesis, the small number of instances in gene expression data is a challenging issue not only for feature selection, feature construction but also for classification algorithms. However, these datasets may share some common knowledge which can be learnt from the larger datasets and adaptively applied to the smaller ones. This can be achieved by transfer learning which is learning based on the knowledge transferred from

related and previously solved problems. GP has shown promise in not only feature construction but also transfer learning. The ability to transfer good building blocks from GP constructed features of the large datasets may help GP improve its learning capability on the smaller ones. Our future work will also investigate this approach.

7.3.8 Ensemble Learning

With thousands of features, the solution space of high-dimensional data may be too complex to represent in a few constructed features. Their feature space may need a large number of high-level and diverse features to better represent the problem. Ensemble learning has become one of the state-of-the-art techniques in machine learning; however, it has not been applied to feature construction or data transformation. New ensemble learning methods or methods using the idea of ensemble learning should be proposed for better addressing these tasks.

Bibliography

[1] Abdul-Rahman, S., Mohamed-Hussein, Z.-A., and Bakar, A. Integrating rough set theory and particle swarm optimisation in feature selection. In *the 10th International Conference on Intelligent Systems Design and Applications (ISDA)* (2010), pp. 1009–1014.

[2] Abedini, M., Kirley, M., and Chiong, R. Incorporating feature ranking and evolutionary methods for the classification of high-dimensional dna microarray gene expression data. *The Australasian medical journal 6*, 5 (2013), 272–279.

[3] Aghdam, M. H., Ghasem-Aghaee, N., and Basiri, M. E. Text feature selection using ant colony optimization. *Expert Systems with Applications 36*, 3, Part 2 (2009), 6843–6853.

[4] Ahluwalia, M., and Bull, L. Coevolving functions in genetic programming: classification using k-nearest-neighbour. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2* (1999), Morgan Kaufmann Publishers Inc., pp. 947–952.

[5] Ahmed, S., Zhang, M., and Peng, L. Genetic programming for biomarker detection in mass spectrometry data. In *Advances in Artificial Intelligence*, M. Thielscher and D. Zhang, Eds. Springer, 2012, pp. 266–278.

[6] Ahmed, S., Zhang, M., and Peng, L. Enhanced feature selection for biomarker discovery in lc-ms data using gp. In *IEEE Congress on Evolutionary Computation* (2013), IEEE, pp. 584–591.

[7] Ahmed, S., Zhang, M., and Peng, L. Feature Selection and Classification of High Dimensional Mass Spectrometry Data: A Genetic Programming Approach. In *Proceedings of Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics* (2013), Springer Berlin Heidelberg, pp. 43–55.

[8] Ahmed, S., Zhang, M., and Peng, L. Improving feature ranking for biomarker discovery in proteomics mass spectrometry data using genetic programming. *Connection Science 26*, 3 (2014), 215–243.

[9] Ahmed, S., Zhang, M., and Peng, L. A new gp-based wrapper feature construction approach to classification and biomarker identification. In *IEEE Congress on Evolutionary Computation* (2014), pp. 2756–2763.

[10] Ahmed, S., Zhang, M., Peng, L., and Xue, B. Multiple feature construction for effective biomarker identification and classification using genetic programming. In *Proceedings of Genetic and evolutionary computation conference* (2014), ACM, pp. 249–256.

[11] Al-Sahaf, H., Al-Sahaf, A., Xue, B., Johnston, M., and Zhang, M. Automatically evolving rotation-invariant texture image descriptors by genetic programming. *IEEE Transactions on Evolutionary Computation 21*, 1 (2017), 83–101.

[12] Al-Sahaf, H., Zhang, M., Johnston, M., and Verma, B. Image descriptor: A genetic programming approach to multiclass texture classification. In *IEEE Congress on Evolutionary Computation* (2015), pp. 2460–2467.

[13] Alba, E., Garcia-Nieto, J., Jourdan, L., and Talbi, E. G. Gene selection in cancer classification using PSO/SVM and GA/SVM

hybrid algorithms. In *IEEE Congress on Evolutionary Computation (CEC)* (2007), pp. 284–290.

[14] Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing 17*, 255-287 (2010), 11.

[15] Alfred, R. *A Genetic-Based Feature Construction Method for Data Summarisation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 39–50.

[16] Almuallim, H., and Dietterich, T. G. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence 69(1-2)* (1994), 279–306.

[17] Ambroise, C., and McLachlan, G. J. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences 99*, 10 (2002), 6562–6566.

[18] Angeline, P. J. *Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences*. Springer Berlin Heidelberg, 1998, pp. 601–610.

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

[20] Babaoglu, I., Findik, O., and Ulker, E. A comparison of feature selection models utilizing binary particle swarm optimization and genetic algorithm in determining coronary artery disease using support vector machine. *Expert Systems with Applications 37*, 4 (2010), 3177 – 3183.

[21] Bäck, T., Fogel, D. B., and Michalewicz, Z. *Evolutionary computation 1: Basic algorithms and operators*, vol. 1. CRC press, 2000.

[22] Baker, L. D., and McCallum, A. K. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (1998), ACM, pp. 96–103.

[23] Ball, G., Mian, S., Holding, F., Allibone, R., Lowe, J., Ali, S., Li, G., McCardle, S., Ellis, I. O., Creaser, C., et al. An integrated approach utilizing artificial neural networks and seldi mass spectrometry for the classification of human tumours and rapid identification of potential biomarkers. *Bioinformatics 18*, 3 (2002), 395–404.

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

[25] Banka, H., and Dara, S. A Hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation. *Pattern Recognition Letters 52* (2015), 94–100.

[26] Bengio, Y. Learning deep architectures for ai. *Foundations and trends in Machine Learning 2*, 1 (2009), 1–127.

[27] Bhanu, B., and Krawiec, K. Coevolutionary construction of features for transformation of representation in machine learning. In *Proceedings of Genetic and Evolutionary Computation Conference* (2002), Press, pp. 249–254.

[28] Bharti, K. K., and kumar Singh, P. A survey on filter techniques for feature selection in text mining. In *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012* (2014), Springer, pp. 1545–1559.

[29] Bharti, K. K., and Singh, P. K. Opposition chaotic fitness mutation based adaptive inertia weight bpso for feature selection in text clustering. *Applied Soft Computing 43* (2016), 20 – 34.

[30] Bhowan, U., Johnston, M., Zhang, M., and Yao, X. Reusing genetic programming for ensemble selection in classification of unbalanced data. *IEEE Transactions on Evolutionary Computation 18*, 6 (Dec 2014), 893–908.

[31] Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. On the effectiveness of discretization on gene selection of microarray data. In *The International Joint Conference on Neural networks (IJCNN),* (2010), IEEE, pp. 1–8.

[32] Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. A review of feature selection methods on synthetic data. *Knowledge and information systems 34*, 3 (2013), 483–519.

[33] Bonabeau, E., Dorigo, M., and Theraulaz, G. *From Natural to Artificial Swarm Intelligence.* Oxford University Press, 1999.

[34] Breiman, L., Friedman, J., Olshen, R., and Stone, C. *Classification and Regression Trees.* Wadsworth and Brooks, 1984.

[35] Butler-Yeoman, T., Xue, B., and Zhang, M. Particle swarm optimisation for feature selection: A hybrid filter-wrapper approach. In *IEEE Congress on Evolutionary Computation* (2015), pp. 2428–2435.

[36] Butterworth, R., Piatetsky-Shapiro, G., and Simovici, D. A. On feature selection through clustering. In *ICDM* (2005), vol. 5, pp. 581–584.

[37] Cano, A., Nguyen, D. T., Ventura, S., and Cios, K. J. ur-CAIM: improved CAIM discretization for unbalanced and balanced data. *Soft Computing* (2014), 1–16.

[38] Catlett, J. On changing continuous attributes into ordered discrete attributes. In *Machine Learning: Proceedings of European Working Session on Learning (EWSL), Porto, Portugal* (Berlin, Heidelberg, 1991), Springer Berlin Heidelberg, pp. 164–178.

[39] Cervante, L., Xue, B., Zhang, M., and Shang, L. Binary particle swarm optimisation for feature selection: A filter based approach. In *IEEE Congress on Evolutionary Computation (CEC'12)* (2012), pp. 881–888.

[40] Cha, S.-H. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences 1* (2007), 300.

[41] Chakraborty, B. Genetic algorithm with fuzzy fitness function for feature selection. In *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics (ISIE)* (2002), vol. 1, pp. 315–319 vol.1.

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

[43] Chakraborty, B., and Chakraborty, G. Fuzzy consistency measure with particle swarm optimization for feature selection. In *IEEE International Conference on Systems, Man, and Cybernetics* (2013), pp. 4311–4315.

[44] Chandrashekar, G., and Sahin, F. A survey on feature selection methods. *Computers & Electrical Engineering 40* (2014), 16–28.

[45] Chao, S., and Li, Y. Multivariate interdependent discretization for continuous attribute. In *Third International Conference on Information Technology and Applications* (2005), vol. 1, IEEE, pp. 167–172.

[46] Chen, D., Chan, K. C. C., and Wu, X. Gene expression analyses using genetic algorithm based hybrid approaches. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* (June 2008), pp. 963–969.

[47] Chen, Q., Chen, Y., and Jiang, W. Genetic particle swarm optimization-based feature selection for very-high-resolution remotely sensed imagery object change detection. *Sensors*, 8 (2016).

[48] Cheng, R., and Jin, Y. A competitive swarm optimizer for large scale optimization. *IEEE Transactions on Cybernetics 45*, 2 (Feb 2015), 191–204.

[49] Cheng, R., and Jin, Y. A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences 291* (2015), 43 – 60.

[50] 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*, 29 (2008), 29–38.

[51] Chuang, L.-Y., Tsai, S.-W., and Yang, C.-H. Improved binary particle swarm optimization using catfish effect for feature selection. *Expert Systems with Applications 38*, 10 (2011), 12699–12707.

[52] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning 20*, 3 (1995), 273–297.

[53] Cover, T., and Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1 (1967), 21–27.

[54] da Silva, S. F., Ribeiro, M. X., do E.S. Batista Neto, J., Traina-Jr., C., and Traina, A. J. Improving the ranking quality of medical image retrieval using a genetic feature selection method. *Decision Support Systems 51*, 4 (2011), 810 – 820.

[55] Dai, Y., Xue, B., and Zhang, M. New representations in pso for feature construction in classification. In *Applications of Evolutionary Computation*. Springer, 2014, pp. 476–488.

[56] Dash, M. Feature selection via set cover. In *Proceedings of IEEE Knowledge and Data Engineering Exchange Workshop* (Nov 1997), pp. 165–171.

[57] Dash, M., and liu, H. Feature selection for classification. *Intelligent data analysis 1*, 4 (1997), 131–156.

[58] Dash, M., and Liu, H. Consistency-based search in feature selection. *Artificial Intelligence 151* (2003), 155–176.

[59] de Souto, M. C., Costa, I. G., de Araujo, D. S., Ludermir, T. B., and Schliep, A. Clustering cancer gene expression data: a comparative study. *BMC Bioinformatics 9*, 1 (2008), 497–510.

[60] Deegalla, S., and Bostrom, H. Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In *the 5th International Conference on Machine Learning and Applications* (2006), pp. 245–250.

[61] Deng, L., and Yu, D. Deep learning: methods and applications. *Foundations and Trends in Signal Processing 7*, 3–4 (2014), 197–387.

[62] Dhillon, I. S., Mallela, S., and Kumar, R. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of machine learning research 3*, Mar (2003), 1265–1287.

[63] Ding, C., and Peng, H. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology 3*, 02 (2005), 185–205.

[64] Dorigo, M., and Di Caro, G. Ant colony optimization: a new meta-heuristic. In *IEEE Congress on Evolutionary Computation* (1999), vol. 2, pp. 1470–1477.

[65] Dougherty, J., Kohavi, R., and Sahami, M. Supervised and unsupervised discretization of continuous features. In *Machine learning: proceedings of the 12th international conference* (1995), vol. 12, pp. 194–202.

[66] Eberhart, R., and Kennedy, J. A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micro Machine and Human Science* (1995), pp. 39–43.

[67] Efron, B. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American Statistical Association 78*, 382 (1983), 316–331.

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

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

[70] Fayyad, U. M., and Irani, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In *the 13th International Joint Conference on Articial Intelligence* (1993), vol. 2, Morgan Kaufmann Publishers, pp. 1022–1027.

[71] Ferreira, A., and Figueiredo, M. Unsupervised joint feature discretization and selection. In *Proceedings of the 5th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), Las Palmas de Gran Canaria, Spain*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 200–207.

[72] Ferreira, A. J., and Figueiredo, M. A. Efficient feature selection filters for high-dimensional data. *Pattern Recognition Letters 33*, 13 (2012), 1794–1804.

[73] Ferreira, A. J., and Figueiredo, M. A. An unsupervised approach to feature discretization and selection. *Pattern Recognition 45*, 9 (2012), 3048–3060.

[74] Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of human genetics 7*, 2 (1936), 179–188.

[75] Flores, J. L., Inza, I., and Larrañaga, P. Wrapper discretization by means of estimation of distribution algorithms. *Intelligent Data Analysis 11*, 5 (2007), 525–545.

[76] Foithong, S., Pinngern, O., and Attachoo, B. Feature subset selection wrapper based on mutual information and rough sets . *Expert Systems with Applications 39*, 1 (2012), 574–584.

[77] Fragoso, R. C. P., Pinheiro, R. H. W., and Cavalcanti, G. D. C. Class-dependent feature selection algorithm for text categorization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (2016), pp. 3508–3515.

[78] Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[79] Garcia, S., Luengo, J., Sáez, J. A., Lopez, V., and Herrera, F. A survey of discretization techniques: taxonomy and empirical

analysis in supervised learning. *Knowledge and Data Engineering, IEEE Transactions on 25*, 4 (2013), 734–750.

[80] Garcia-Limon, M., Escalante, H. J., Morales, E., and Morales-Reyes, A. Simultaneous generation of prototypes and features through genetic programming. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation* (2014), ACM, pp. 517–524.

[81] Garcia-Torres, M., Gomez-Vela, F., Melian-Batista, B., and Moreno-Vega, J. M. High-dimensional feature selection via feature grouping: A Variable Neighborhood Search approach. *Information Sciences 326* (2016), 102–118.

[82] Geurts, P., Fillet, M., De Seny, D., Meuwis, M.-A., Malaise, M., Merville, M.-P., and Wehenkel, L. Proteomic mass spectra classification using decision tree based ensemble methods. *Bioinformatics 21*, 14 (2005), 3138–3145.

[83] Ghamisi, P., and Benediktsson, J. A. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *IEEE Geoscience and Remote Sensing Letters 12*, 2 (2015), 309–313.

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

[85] Grzymala-Busse, J. W. Discretization based on entropy and multiple scanning. *Entropy 15*, 5 (2013), 1486–1502.

[86] Gu, S., Cheng, R., and Jin, Y. Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing 22*, 3 (2018), 811–822.

[87] Guan, J., Han, F., and Yang, S. A new gene selection method for microarray data based on pso and informativeness metric. In *the*

9th International Conference on Intelligent Computing Theories and Technology (2013), pp. 145–154.

[88] Gui, J., Sun, Z., Ji, S., Tao, D., and Tan, T. Feature selection based on structured sparsity: A comprehensive study. *IEEE transactions on neural networks and learning systems* (2017), 1490–1507.

[89] Gupta, A., Gupta, A., and Sharma, K. Clustering based feature selection methods from fmri data for classification of cognitive states of the human brain. In *the 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (2016), IEEE, pp. 3581–3584.

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

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

[92] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine learning 46*, 1-3 (2002), 389–422.

[93] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter 11*, 1 (2009), 10–18.

[94] Hall, M. A. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 7th International Conference on Machine Learning* (2000), pp. 359–366.

[95] Holland, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[96] Holte, R. C. Very simple classification rules perform well on most commonly used datasets. *Machine learning 11*, 1 (1993), 63–90.

[97] Hong, J.-H., and Cho, S.-B. Efficient huge-scale feature selection with speciated genetic algorithm. *Pattern Recognition Letters 27*, 2 (2006), 143–150.

[98] Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology 24*, 6 (1933), 417.

[99] Hsu, H.-H., and Hsieh, C.-W. Feature selection via correlation coefficient clustering. *Journal of Software 5*, 12 (2010), 1371–1377.

[100] Hsu, H.-H., Hsieh, C.-W., and Lu, M.-D. Hybrid feature selection by combining filters and wrappers. *Expert Systems with Applications 38*, 7 (2011), 8144–8150.

[101] Hu, Y.-J. *Constructive Induction: Covering Attribute Spectrum.* Springer US, Boston, MA, 1998, pp. 257–272.

[102] Huang, C.-L., and Dun, J.-F. A distributed pso–svm hybrid system with feature selection and parameter optimization. *Applied Soft Computing 8*, 4 (2008), 1381–1391.

[103] Huang, J., Cai, Y., and Xu, X. A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recognition Letters 28*, 13 (2007), 1825–1844.

[104] Inbarani, H. H., Azar, A. T., and Jothi, G. Supervised hybrid feature selection based on pso and rough sets for medical diagnosis. *Computer methods and programs in biomedicine 113*, 1 (2014), 175–185.

[105] Jaba Sheela, L., and Shanthi, D. V. An approach for discretization and feature selection of continuous-valued attributes in medical images

for classification learning. *International Journal of Computer Theory and Engineering 1*, 2 (2009), 154–158.

[106] Jain, I., Jain, V. K., and Jain, R. Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification. *Applied Soft Computing 62* (2018), 203–215.

[107] Jakulin, A., and Bratko, I. Testing the significance of attribute interactions. In *Proceedings of the 21st International Conference on Machine Learning (ICML)* (2004), ACM, pp. 52–59.

[108] Jamian, J. J., Abdullah, M. N., Mokhlis, H., Mustafa, M. W., and Bakar, A. H. A. Global particle swarm optimization for high dimension numerical functions analysis. *Journal of Applied Mathematics 2014* (2014).

[109] Jaskowiak, P. A., and Campello, R. J. A cluster based hybrid feature selection approach. In *Brazilian Conference on Intelligent Systems (BRACIS)* (2015), IEEE, pp. 43–48.

[110] Jensen, R. Performing feature selection with ACO. In *Swarm Intelligence in Data Mining*. Springer, 2006, pp. 45–73.

[111] Jirapech-Umpai, T., and Aitken, S. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC bioinformatics 6*, 1 (2005), 148–158.

[112] John, G. H., Kohavi, R., and Pfleger, K. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning* (1994), Morgan Kaufmann, pp. 121–129.

[113] Kabir, M. M., Shahjahan, M., and Murase, K. A new local search based hybrid genetic algorithm for feature selection. *Neurocomputing 74*, 17 (2011), 2914–2928.

[114] Kanan, H. R., and Faez, K. An improved feature selection method based on ant colony optimization (ACO) evaluated on face recognition system. *Applied Mathematics and Computation 205*, 2 (2008), 716–725.

[115] Kennedy, J. Bare bones particle swarms. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS).* (2003), pp. 80–87.

[116] Kennedy, J., and Eberhart, R. A discrete binary version of the particle swarm algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics* (1997), vol. 5, pp. 4104–4108.

[117] Kennedy, J., and Spears, W. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In *IEEE World Congress on Computational Intelligence* (1998), pp. 78–83.

[118] Kira, K., and Rendell, L. A. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (1992), pp. 129–134.

[119] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* (1995), pp. 1137–1143.

[120] Kohavi, R., and John, G. H. Wrappers for feature subset selection. *Artificial Intelligence 97*, 1–2 (1997), 273–324.

[121] Koller, D., and Sahami, M. Toward optimal feature selection. In *Proceedings of the 13th International Conference on International Conference on Machine Learning* (San Francisco, CA, USA, 1996), Morgan Kaufmann Publishers Inc., pp. 284–292.

[122] Kononenko, I. Estimating attributes: analysis and extensions of relief. In *Machine Learning: ECML-94* (1994), Springer, pp. 171–182.

[123] Kotsiantis, S., and Kanellopoulos, D. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering 32*, 1 (2006), 47–58.

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

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

[126] Krawiec, K. *Evolutionary Feature Programming: Cooperative learning for knowledge discovery and computer vision*. Wydawnictwo Politechniki Poznanskiej, 2004.

[127] Krier, C., François, D., Rossi, F., Verleysen, M., and Chesnay Cedex, F. Feature clustering and mutual information for the selection of variables in spectral data. In *European Symposium on Artificial Neural Networks (ESANN)* (2007), pp. 157–162.

[128] Lane, M., Xue, B., Liu, I., and Zhang, M. Particle swarm optimisation and statistical clustering for feature selection. In *AI 2013: Advances in Artificial Intelligence*, vol. 8272 of *Lecture Notes in Computer Science*. Springer, 2013, pp. 214–220.

[129] Lane, M., Xue, B., Liu, I., and Zhang, M. Gaussian based particle swarm optimisation and statistical clustering for feature selection. In *Evolutionary Computation in Combinatorial Optimisation (EvoCOP)*, vol. 8600. Springer Berlin Heidelberg, 2014, pp. 133–144.

[130] Langdon, W. B., and Buxton, B. F. Genetic programming for mining dna chip data from cancer patients. *Genetic Programming and Evolvable Machines 5*, 3 (2004), 251–257.

[131] Langley, P. Selection of relevant features in machine learning. In *AAAI Technique Report FS-94-02* (1994), pp. 127–131.

[132] Langley, P., Bradshaw, G. L., and Simon, H. A. *Rediscovering chemistry with the BACON system*. Springer, 1983.

[133] Lanzi, P.-L. Fast feature selection with genetic algorithms: a filter approach. In *IEEE International Conference on Evolutionary Computation* (1997), pp. 537–540.

[134] Lazar, C., Taminau, J., Meganck, S., Steenhoff, D., Coletta, A., Molter, C., de Schaetzen, V., Duque, R., Bersini, H., and Nowe, A. A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 9*, 4 (July 2012), 1106–1119.

[135] Lecocke, M., and Hess, K. An empirical study of univariate and genetic algorithm-based feature selection in binary classification with microarray data. *Cancer informatics 2* (2006), 313–327.

[136] Li, X., and Yao, X. Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In *2009 IEEE Congress on Evolutionary Computation* (May 2009), pp. 1546–1553.

[137] Li, X., and Yao, X. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation 16*, 2 (2012), 210–224.

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

[139] Liu, H., Hussain, F., Tan, C. L., and Dash, M. Discretization: An enabling technique. *Data mining and knowledge discovery 6*, 4 (2002), 393–423.

[140] Liu, H., and Motoda, H. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[141] Liu, H., Motoda, H., Setiono, R., and Zhao, Z. Feature selection: An ever evolving frontier in data mining. *Journal of Machine Learning Research - Proceedings Track* (2010), 4–13.

[142] Liu, H., and Setiono, R. Chi2: feature selection and discretization of numeric attributes. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence* (Nov 1995), pp. 388–391.

[143] Liu, H., and Setiono, R. A probabilistic approach to feature selection - a filter solution. In *International Conference on Machine Learning* (1996), vol. 96, pp. 319–327.

[144] Liu, H., and Setiono, R. Feature selection and classification-a probabilistic wrapper approach. In *Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES* (1997), pp. 419–424.

[145] Liu, H., and Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering 17* (2005), 491–502.

[146] Liu, X., and Shang, L. A fast wrapper feature subset selection method based on binary particle swarm optimization. In *IEEE Congress on Evolutionary Computation (CEC)* (2013), pp. 3347–3353.

[147] Lloyd, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2 (March 1982), 129–137.

[148] Lustgarten, J. L., Gopalakrishnan, V., Grover, H., and Visweswaran, S. Improving classification performance with discretization on biomedical datasets. In *AMIA Annual Symposium Proceedings* (2008), pp. 445–449.

[149] MacLean, D., Wollesen, E., and Worzel, B. Listening to data: Tuning a genetic programming system. *Genetic programming theory and practice II* (2005), 245–262.

[150] Mahanta, P., Ahmed, H. A., Kalita, J. K., and Bhattacharyya, D. K. Discretization in gene expression data analysis: a selected survey. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology* (2012), ACM, pp. 69–75.

[151] Mao, Q., and Tsang, I. W.-H. A feature selection method for multivariate performance measures. *IEEE transactions on pattern analysis and machine intelligence 35*, 9 (2013), 2051–2063.

[152] Marcano-Cedeno, A., Quintanilla-Domínguez, J., Cortina-Januchs, M., and Andina, D. Feature selection using sequential forward selection and classification applying artificial metaplasticity neural network. In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society* (2010), IEEE, pp. 2845–2850.

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

[154] Martínez, A. M., and Kak, A. C. Pca versus lda. *IEEE transactions on pattern analysis and machine intelligence 23*, 2 (2001), 228–233.

[155] Mettes, P., Tan, R., and Veltkamp, R. A bottom-up approach to class-dependent feature selection for material classification. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)* (2014), vol. 2, pp. 494–501.

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

[157] Ming, H. A rough set based hybrid method to feature selection. In *International Symposium on Knowledge Acquisition and Modeling* (2008), pp. 585–588.

[158] Mishra, D., Rath., A., Acharya., M., and Jena., T. Rough aco: A hybridized model for feature selection in gene expression data. *International Journal of Computer Communication and Technology 1* (2009), 85–98.

[159] Mistry, K., Zhang, L., Neoh, S. C., Lim, C. P., and Fielding, B. A micro-ga embedded pso feature selection approach to intelligent facial emotion recognition. *IEEE Transactions on Cybernetics 47*, 6 (2017), 1496–1509.

[160] Mitchell, T. M. *Machine Learning*, 1 ed. McGraw-Hill, Inc., New York, NY, USA, 1997.

[161] Mohamad, M., Omatu, S., Deris, S., and Yoshioka, M. A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data. *Information Technology in Biomedicine 15*, 6 (2011), 813–822.

[162] Mohamad, M., Omatu, S., Deris, S., Yoshioka, M., Abdullah, A., and Ibrahim, Z. An enhancement of binary particle swarm optimization for gene selection in classifying cancer classes. *Algorithms for Molecular Biology 8*, 1 (2013), 1–11.

[163] Mohemmed, A., Zhang, M., and Johnston, M. Particle Swarm Optimization based Adaboost for face detection. In *IEEE Congress on Evolutionary Computation* (2009), pp. 2494–2501.

[164] Moradi, P., and Gholampour, M. A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Applied Soft Computing 43* (2016), 117–130.

[165] Muharram, M., and Smith, G. The Effect of Evolved Attributes on Classification Algorithms. In *AI 2003: Advances in Artificial Intelligence*, vol. 2903. Springer Berlin Heidelberg, 2003, pp. 933–941.

[166] Muharram, M., and Smith, G. Evolutionary constructive induction. *IEEE Transactions on Knowledge and Data Engineering 17* (2005), 1518–1528.

[167] Murthy, S. K., Kasif, S., and Salzberg, S. A system for induction of oblique decision trees. *Journal of Articial Intelligence Research 2* (1994), 1–32.

[168] Nag, K., and Pal, N. A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification. *IEEE Transactions on Cybernetics 46*, 2 (2016), 499–510.

[169] Narendra, P., and Fukunaga, K. A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on Computers C-26*, 9 (1977), 917–922.

[170] Neshatian, K. *Feature Manipulation with Genetic Programming*. PhD thesis, Victoria University of Wellington, Wellington, New Zealand, 2010.

[171] Neshatian, K., and Zhang, M. Genetic programming for performance improvement and dimensionality reduction of classification problems. In *IEEE Congress on Computational Intelligence* (2008), pp. 2811–2818.

[172] Neshatian, K., and Zhang, M. Pareto front feature selection: Using genetic programming to explore feature space. In *Conference on Genetic and Evolutionary Computation* (2009), pp. 1027–1034.

[173] Neshatian, K., and Zhang, M. Using genetic programming for context-sensitive feature scoring in classification problems. *Connection Science 23*, 3 (2011), 183–207.

[174] Neshatian, K., Zhang, M., and Andreae, P. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation 16*, 5 (2012), 645–661.

[175] Neshatian, K., Zhang, M., and Johnston, M. Feature Construction and Dimension Reduction Using Genetic Programming. In *Advances in Artificial Intelligence*, vol. 4830. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 160–170.

[176] Nguyen, H. B., Xue, B., and Andreae, P. *A Hybrid GA-GP Method for Feature Reduction in Classification*. Springer International Publishing, 2017, pp. 591–604.

[177] Nguyen, H. B., Xue, B., Andreae, P., and Zhang, M. Particle swarm optimisation with genetic operators for feature selection. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)* (2017), pp. 286–293.

[178] Nguyen, H. B., Xue, B., Liu, I., Andreae, P., and Zhang, M. Gaussian transformation based representation in particle swarm optimisation for feature selection. In *Applications of Evolutionary Computation*. Springer, 2015, pp. 541–553.

[179] Nguyen, H. B., Xue, B., Liu, I., and Zhang, M. Filter based backward elimination in wrapper based pso for feature selection in classification. In *IEEE Congress on Evolutionary Computation* (2014), pp. 3111–3118.

[180] Nguyen, H. B., Xue, B., Liu, I., and Zhang, M. PSO and Statistical Clustering for Feature Selection: A New Representation. In *Simulated Evolution and Learning*. Springer, 2014, pp. 569–581.

[181] Novaković, J. Toward optimal feature selection using ranking methods and classification algorithms. *Yugoslav Journal of Operations Research 21*, 1 (2016), 119–135.

[182] Oliveira, L., Sabourin, R., Bortolozzi, F., and Suen, C. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In *the 16th International Conference on Pattern Recognition* (2002), vol. 1, pp. 568–571 vol.1.

[183] Omidvar, M. N., Li, X., Mei, Y., and Yao, X. Cooperative coevolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation 18*, 3 (June 2014), 378–393.

[184] Otero, F., Silva, M., Freitas, A., and Nievola, J. Genetic programming for attribute construction in data mining. In *Genetic Programming*, vol. 2610. Springer Berlin Heidelberg, 2003, pp. 384–393.

[185] Pal, S., and Chakraborty, B. Fuzzy Set Theoretic Measure for Automatic Feature Evaluation. *IEEE Transactions on Systems, Man and Cybernetics 16*, 5 (1986), 754–760.

[186] Pang-Ning, T., Steinbach, M., Kumar, V., et al. Introduction to data mining. In *Library of Congress* (2006).

[187] Patterson, G., and Zhang, M. Fitness functions in genetic programming for classification with unbalanced data. In *Proceedings of the 20th Australian Joint Conference on Artificial Intelligence (AI)*. Springer Berlin Heidelberg, 2007, pp. 769–775.

[188] Petalas, Y. G., Parsopoulos, K. E., and Vrahatis, M. N. Memetic particle swarm optimization. *Annals of Operations Research 156*, 1 (2007), 99–127.

[189] Poli, R. Analysis of the publications on the applications of particle swarm optimisation. *Artificial Evolution and Applications* (2008), 4:1–4:10.

[190] Poli, R., Langdon, W. B., McPhee, N. F., and Koza, J. R. *A field guide to genetic programming.* Lulu. com, 2008.

[191] Prasad, Y., Biswas, K., and Hanmandlu, M. A recursive pso scheme for gene selection in microarray data. *Applied Soft Computing. DOI: https://doi.org/10.1016/j.asoc.2018.06.019* (2018).

[192] Press, W. H., Teukolsky, S., Vetterling, W., and Flannery, B. Numerical recipes in c. *Cambridge University Press 1* (1988), 3.

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

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

[195] Qiao, Z., Zhou, L., and Huang, J. Z. Sparse linear discriminant analysis with applications to high dimensional low sample size data, 2009.

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

[197] Ramirez-Gallego, S., Garcia, S., Benitez, J., and Herrera, F. Multivariate discretization based on evolutionary cut points selection for classification. *IEEE Transactions on Cybernetics, 46*, 3 (2016), 595–608.

[198] Raymer, M. L., Punch, W. F., Goodman, E. D., and Kuhn, L. A. Genetic programming for improved data mining: Application to the biochemistry of protein interactions. In *Proceedings of the 1st Annual Conference on Genetic Programming* (Cambridge, MA, USA, 1996), MIT Press, pp. 375–380.

[199] Robnik-Sikonja, M., and Kononenko, I. Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning 53* (2003), 23–69.

[200] Rostami, M., and Moradi, P. A clustering based genetic algorithm for feature selection. In *Conference on Information and Knowledge Technology* (2014), pp. 112–116.

[201] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics 20* (1987), 53–65.

[202] Ruiz, R., Riquelme, J. C., and Aguilar-Ruiz, J. S. Incremental wrapper-based gene selection from microarray data for cancer classification. *Pattern Recognition 39*, 12 (2006), 2383–2392.

[203] Saeys, Y., Inza, I., and Larrañaga, P. A review of feature selection techniques in bioinformatics. *bioinformatics 23*, 19 (2007), 2507–2517.

[204] Sahu, B., and Mishra, D. A Novel Feature Selection Algorithm using Particle Swarm Optimization for Cancer Microarray Data. *Procedia Engineering 38*, 0 (2012), 27–31.

[205] Santana, L., Silva, L., Canuto, A., Pintro, F., and Vale, K. A comparative analysis of genetic algorithm and ant colony optimization to select attributes for an heterogeneous ensemble of classifiers. In *IEEE Congress on Evolutionary Computation* (2010), pp. 1–8.

[206] Schlimmer, J. C. Learning and representation change. In *Conference of Association for the Advancement of Artificial Intelligence (AAAI)* (1987), pp. 511–515.

[207] Schölkopf, B., Burges, C. J. C., and Smola, A. J., Eds. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, USA, 1999.

[208] Scholz, M., Fraunholz, M., and Selbig, J. *Nonlinear Principal Component Analysis: Neural Network Models and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 44–67.

[209] Shafti, L. S., and Pérez, E. *Genetic Approach to Constructive Induction Based on Non-algebraic Feature Representation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 599–610.

[210] Shen, Q., Shi, W.-M., and Ye, W. K. B.-X. A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification. *Talanta 71*, 4 (2007), 1679–1683.

[211] Shi, Y., and Eberhart, R. Empirical study of particle swarm optimization. In *IEEE Congress on Evolutionary Computation (CEC'99)* (1999), vol. 3, pp. 1945–1950.

[212] Slonim, N., and Tishby, N. The power of word clusters for text classification. In *the 23rd European Colloquium on Information Retrieval Research* (2001), vol. 1, pp. 200–211.

[213] Smith, M., and Bull, L. Genetic Programming with a Genetic Algorithm for Feature Construction and Selection. *Genetic Programming and Evolvable Machines 6* (2005), 265–281.

[214] Smith, M., and Bull, L. Improving the human readability of features constructed by genetic programming. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation* (2007), ACM, pp. 1694–1701.

[215] Smith, M. G., and Bull, L. *Using Genetic Programming for Feature Creation with a Genetic Algorithm Feature Selector*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 1163–1171.

[216] Song, Q., Ni, J., and Wang, G. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE Transactions on Knowledge and Data Engineering 25* (2013), 1–14.

[217] Statnikov, A., Aliferis, C. F., Tsamardinos, I., Hardin, D., and Levy, S. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics 21* (2005), 631–643.

[218] Statnikov, A., Tsamardinos, I., Dosbayev, Y., and Aliferis, C. F. Gems: a system for automated cancer diagnosis and biomarker discovery from microarray gene expression data. *International journal of medical informatics 74*, 7 (2005), 491–503.

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

[220] Stretch, C., Khan, S., Asgarian, N., Eisner, R., Vaisipour, S., Damaraju, S., Graham, K., Bathe, O. F., Steed, H., Greiner, R., and Baracos, V. E. Effects of sample size on differential gene expression, rank order and prediction accuracy of a gene signature. *PLOS ONE 8*, 6 (06 2013), 1–6.

[221] Stuart J. Russell, P. N. Artificial intelligence: A modern approach (second edition). *Pearson Education* (2003).

[222] Sun, Y. Iterative relief for feature weighting: algorithms, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Isntelligence 29*, 6 (2007), 1035–1051.

[223] Tan, F., Fu, X., Zhang, Y., and Bourgeois, A. G. A genetic algorithm-based method for feature subset selection. *Soft Computing-A Fusion of Foundations, Methodologies and Applications 12*, 2 (2008), 111–120.

[224] Tan, M., Tsang, I. W., and Wang, L. Minimax sparse logistic regression for very high-dimensional feature selection. *IEEE transactions on neural networks and learning systems 24*, 10 (2013), 1609–1622.

[225] Tan, M., Wang, L., and Tsang, I. W. Learning sparse svm for feature selection on very high dimensional datasets. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (2010), pp. 1047–1054.

[226] Tay, F. E., and Shen, L. A modified chi2 algorithm for discretization. *IEEE Transactions on Knowledge and Data Engineering 14*, 3 (2002), 666–670.

[227] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.

[228] Tran, B., Xue, B., and Zhang, M. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing 8*, 1 (2015), 3–15.

[229] Tran, B., Xue, B., and Zhang, M. A new representation in pso for discretization-based feature selection. *IEEE Transactions on Cybernetics 48*, 6 (2018), 1733–1746.

[230] Unler, A., and Alper Murat, R. B. C. mr2pso: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification. *Information Sciences 20* (2011), 4625–4641.

[231] 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.

[232] Urbanowicz, R. J., Meeker, M., Cava, W. L., Olson, R. S., and Moore, J. H. Relief-based feature selection: Introduction and review. *CoRR http://arxiv.org/abs/1711.08421* (2017).

[233] Urbanowicz, R. J., Olson, R. S., Schmitt, P., Meeker, M., and Moore, J. H. Benchmarking relief-based feature selection methods. *CoRR http://arxiv.org/abs/1711.08477* (2017).

[234] Vafaie, H., and De Jong, K. Genetic algorithms as a tool for restructuring feature space representations. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence* (1995), pp. 8–11.

[235] Van den Bergh, F., and Engelbrecht, A. P. A study of particle swarm optimization particle trajectories. *Information sciences 176*, 8 (2006), 937–971.

[236] Vignolo, L. D., Milone, D. H., and Scharcanski, J. Feature selection for face recognition based on multi-objective evolutionary wrappers. *Expert Systems with Applications 40*, 13 (2013), 5077 – 5084.

[237] Wang, L., Zhou, N., and Chu, F. A general wrapper approach to selection of class-dependent features. *IEEE Transactions on Neural Networks 19*, 7 (2008), 1267–1278.

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

[239] Waqas, K., Baig, R., and Ali, S. Feature subset selection using multi-objective genetic algorithms. In *the 13th International Conference on INMIC* (2009), pp. 1–6.

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

[241] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin 1*, 6 (1945), 80–83.

[242] Wu, B., Abbott, T., Fishman, D., McMurray, W., Mor, G., Stone, K., Ward, D., Williams, K., and Zhao, H. Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics 19*, 13 (2003), 1636–1643.

[243] Xing, E. P., Jordan, M. I., and Karp, R. M. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the 18th International Conference on Machine Learning* (2001), Morgan Kaufmann, pp. 601–608.

[244] Xu, D., and Tian, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science 2*, 2 (2015), 165–193.

[245] Xu, R., and Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on neural networks 16*, 3 (2005), 645–678.

[246] Xu, R.-F., and Lee, S.-J. Dimensionality reduction by feature clustering for regression problems. *Information Sciences 299* (2015), 42–57.

[247] Xu, Z., Jin, R., Ye, J., Lyu, M. R., and King, I. Non-monotonic feature selection. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 1145–1152.

[248] Xue, B. *Particle Swarm Optimisation for Feature Selection in Classification*. PhD thesis, Victoria University of Wellington, Wellington, New Zealand, 2014.

[249] Xue, B., Cervante, L., Shang, L., Browne, W., and Zhang, M. A multi-objective particle swarm optimisation for filter-based feature selection in classification problems. *Connection Science 24*, 2-3 (2012), 91–116.

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

[251] Xue, B., Zhang, M., and Browne, W. Novel initialisation and updating mechanisms in PSO for feature selection in classification. In *Applications of Evolutionary Computation*, vol. 7835 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 428–438.

[252] Xue, B., Zhang, M., and Browne, W. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics 43*, 6 (2013), 1656–1671.

[253] Xue, B., Zhang, M., and Browne, W. N. Multi-objective particle swarm optimisation for feature selection. In *Genetic and Evolutionary Computation Conference (GECCO'12), Philadelphia, PA, USA* (2012), ACM, pp. 81–88.

[254] Xue, B., Zhang, M., and Browne, W. N. Single feature ranking and binary particle swarm optimisation based feature subset ranking for feature selection. In *Proceedings of the 35th Australasian Computer Science Conference-Volume 122* (2012), Australian Computer Society, Inc., pp. 27–36.

[255] Xue, B., Zhang, M., and Browne, W. N. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing 18* (2014), 261–276.

[256] Xue, B., Zhang, M., Browne, W. N., and Yao, X. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation 20*, 4 (2016), 606–626.

[257] Xue, B., Zhang, M., Dai, Y., and Browne, W. N. Pso for feature construction and binary classification. In *Proceeding of the fifteenth*

annual conference on Genetic and evolutionary computation conference (2013), ACM, pp. 137–144.

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

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

[260] Yang, P., Li, J.-S., and Huang, Y.-X. HDD: a hypercube division-based algorithm for discretisation. *International Journal of Systems Science 42*, 4 (2011), 557–566.

[261] Yu, H., Gu, G., Liu, H., Shen, J., and Zhao, J. A modified ant colony optimization algorithm for tumor marker gene selection. *Genomics, Proteomics & Bioinformatics 7*, 4 (2009), 200–208.

[262] Yu, L., and Liu, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th on International Conference on Machine Learning (ICML)* (2003), pp. 856–863.

[263] Yu, L., and Liu, H. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research 5*, Oct (2004), 1205–1224.

[264] Yuan, M., and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68*, 1 (2006), 49–67.

[265] Yusta, S. C. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letter 30* (2009), 525–534.

[266] Zhang, Y., Gong, D., Hu, Y., and Zhang, W. Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputing 148* (2015), 150 – 157.

[267] 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* (2009), 2633–2644.

[268] Zhao, Z., and Liu, H. Searching for interacting features. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (2007), IJCAI'07, pp. 1156–1161.

[269] Zheng, Z. Constructing X-of-N Attributes for Decision Tree Learning. *Machine Learning 40* (2000), 35–75.

[270] Zhou, W., and Dickerson, J. A. A novel class dependent feature selection method for cancer biomarker discovery. *Computers in biology and medicine 47* (2014), 66–75.

[271] Zhu, Z., Ong, Y.-S., and Dash, M. Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognition 40*, 11 (2007), 3236–3248.

[272] Zhu, Z., Ong, Y.-S., and Zurada, J. M. Identification of full and partial class relevant genes. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on 7*, 2 (2010), 263–277.

[273] Zou, H., and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*, 2 (2005), 301–320.