Bare-Bone Particle Swarm Optimisation for Simultaneously Discretising and Selecting Features for High-Dimensional Classification

Binh Tran*, Bing Xue, and Mengjie Zhang

School of Engineering and Computer Science, Victoria University of Wellington,
PO Box 600, Wellington 6140, New Zealand
{binh.tran,bing.xue,mengjie.zhang}@ecs.vuw.ac.nz

Abstract. Feature selection and discretisation have shown their effectiveness for data preprocessing especially for high-dimensional data with many irrelevant features. While feature selection selects only relevant features, feature discretisation finds a discrete representation of data that contains enough information but ignoring some minor fluctuation. These techniques are usually applied in two stages, discretisation and then selection since many feature selection methods work only on discrete features. Most commonly used discretisation methods are univariate in which each feature is discretised independently; therefore, the feature selection stage may not work efficiently since information showing feature interaction is not considered in the discretisation process. In this study, we propose a new method called PSO-DFS using bare-bone particle swarm optimisation (BBPSO) for discretisation and feature selection in a single stage. The results on ten high-dimensional datasets show that PSO-DFS obtains a substantial dimensionality reduction for all datasets. The classification performance is significantly improved or at least maintained on nine out of ten datasets by using the transformed "small" data obtained from PSO-DFS. Compared to applying the two-stage approach which uses PSO for feature selection on the discretised data, PSO-DFS achieves better performance on six datasets, and similar performance on three datasets with a much smaller number of features selected.

Keywords: Particle swarm optimisation, feature discretisation, feature selection, classification, high-dimensional data

1 Introduction

Feature selection is an important technique in data preprocessing, especially for datasets with thousands to tens of thousands of features. High-dimensional datasets, such as text, image and gene expression data, are automatically collected by machines. Therefore, they usually contain a significant number of irrelevant and redundant features, which negatively affects not only the learning process but also the system memory. To deal with this problem, feature selection

* Corresponding authors.

has been applied to select only informative features. Results of many studies have shown the effectiveness of applying feature selection [1, 2] in general as well as on high-dimensional data [3, 4]. Existing feature selection methods can be generally classified into filter and wrapper approaches [5]. Filter approaches evaluate features based on their intrinsic characteristics. On the other hand, wrapper approaches use a learning algorithm to measure the classification performance of the selected features. Although filters are said to be faster than wrappers, they usually obtain lower classification accuracy than wrappers.

Besides feature selection, feature discretisation is also important in preprocessing data especially for high-dimensional data because of the following important reasons. Firstly, many commonly used machine learning techniques can only be applied or work efficiently on discrete data. High-dimensional datasets usually include continuous features that are automatically collected at the interval or ratio level such as images and gene expression data. Therefore, continuous or real-value features are required to be partitioned or discretised into a number of sub-ranges. Each sub-range is considered as a category. This process is called discretisation. Secondly, 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 [6]. The report in [7] showed that feature discretisation helps learning be more accurate and faster. Finally, discrete features are more compact than continuous ones. Discretisation can reduce a significant amount of memory required to store data, enabling learning algorithms to work efficiently. As a result, feature selection and feature discretisation have been used as popular preprocessing techniques to obtain more compact datasets that are better representatives of the learning task. The use of these techniques has shown to improve classification performance, memory requirements and computation time [8].

Although many discretisation methods have been proposed, most of them are univariate, which means that only one continuous feature is discretised at a time. 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 problems in which feature interdependency occurs [9]. In these cases, multivariate discretisation is needed to consider multiple features at a time. However, it would certainly increase the time complexity for discretisation. Therefore, it is necessary to have an efficient search technique that can simultaneously discretise multiple features in a reasonable time.

A common practice using feature selection and discretisation in data preprocessing is to apply discretisation before the selection process. However, 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. Therefore, combining these two processes into a single stage may obtain better representation for the learning task.

Particle swarm optimisation (PSO), proposed by Kennedy and Eberhart [10], is a meta-heuristic algorithm inspired by social behaviours found in birds flock-

ing. Each particle flies from one position to another in the problem search space with its own velocity. PSO has been applied to different tasks including feature selection using different versions such as continuous PSO [11, 12] and binary PSO [13–15]. However, the performance of these PSO versions strongly depend on the control parameters for particles' velocity such as inertia weight, acceleration coefficients and velocity clamping [16]. Although many adaptive and dynamic methods have been proposed to overcome this problem, it is still challenging since setting these parameters depends on individual applications and needs to be adjusted for different problems. In 2003, Kennedy [17] suggested updating particles' position using a simple Gaussian sampling around the mean of the individual best position and its neighbours' best position. This "bare-bone" PSO (BBPSO) eliminates the use of velocity and all of the parameters mentioned above. The performance of this new method on function optimisation problems has shown to be superior than the canonical one. This method also attained promising results in [18] which proposed binary BBPSO for feature selection. While PSO has been widely used for feature selection, it has not been applied to feature discretisation and selection at the same time.

Goals

The aim of this study is to propose a new approach to the use of BBPSO for simultaneously discretising features and selecting relevant features in a single stage for high-dimensional continuous data. For presentation convenience, we call the new method PSO-DFS. PSO-DFS will be examined and compared with using all original feature set and the corresponding two-stage approach (feature discretisation and then feature selection) on ten public available high-dimensional datasets of varying difficulty. More specifically, we would like to investigate the following research objectives:

1. How to discretise multiple features simultaneously in BBPSO so that the discriminating power of the feature set is improved;
2. How to perform discretise and select features in a single stage;
3. Whether the features generated by PSO-DFS can produce better classification performance than using all features;
4. Whether PSO-DFS can outperform the corresponding two-stage approach in terms of classification accuracy, the number of features and the computation time.

2 Background and Related Work

2.1 Feature Discretisation

In order to discretise a continuous feature into a discrete one, a discretisation method determines the number of intervals and the corresponding cut-points for each interval. A large number of discretisation methods can be found in the literature. They can be categorised based on different axes [19–21] such as *direct* versus *incremental*, *supervised* versus *unsupervised*. While *direct* methods determine these intervals based on a user-defined parameter, *incremental* methods

apply some criteria to further split or merge intervals forming *splitting* or *merging* methods, respectively. Methods which use class labels in the discretisation process are *supervised*; otherwise, they are *unsupervised*. Discretisation methods can also be categorised into *global* or *local* based on whether the entire instance space or a subset of instances is used in each discretisation step. While in *dynamic* methods, the discretisation process is done while the learner is building the model, *static* methods separate these two processes. Discretisation methods are also categorized into *univariate* where each feature is discretised independently and *multivariate* where multiple features are discretised at the same time so that feature interaction is also considered in the discretisation process [19].

Two simple unsupervised discretisation methods are equal-width and equal-frequency binning methods, which require a user-defined number of intervals m. While the former method partitions features into m intervals with the same width, the latter partitions features into m intervals that have the same number of instances. Although these binning methods are simple and easy to implement, they are sensitive to a given m which is usually unknown. They may not give good results on non-uniform distribution features and features with outliers, which are extreme values that strongly affect the ranges [22].

To overcome the shortcoming of unsupervised methods, supervised discretisation takes into account the interdependence between the discrete values and their class labels. Different ways of using class labels to find cut-points with higher class coherence have been proposed. A simple example is 1R [23] where cut-points have to lie between sorted instances of different classes and each bin has at least six instances except the right most bin. Many ways of evaluating a cut-point are proposed based on information theory [24, 25], statistical measures [26, 27], classification error rates [28, 29], etc. Readers are referred to more comprehensive review in [19, 21, 20, 30].

2.2 Minimum Description Length

Fayyad and Irani's minimum description length (MDL) [25] is a univariate incremental splitting discretisation method which uses the minimum description length principle (MDLP) as the stopping criterion. The algorithm starts with one interval containing all values of the feature and recursively partitions this interval until the criterion is met.

In each discretisation step, a cut-point is chosen to partition the corresponding interval into two sub-intervals. The algorithm considers all candidate cut-points which lie between instances of different classes. The best cut-point is the one with the highest information gain. Given S as the set of instances, T is a candidate cut-point of Feature A, S_1 and S_2 are the resulting subsets after partitioning S by T, information gain of T is calculated based on Equation (1). A cut-point is only accepted if its information gain satisfies the MDLP criterion as shown in Equation (2), where details about this equation can be seen from [25].

$$Gain(T, A; S) = E(S) - \frac{|S_1|}{|S|}E(S_1) - \frac{|S_2|}{|S|}E(S_2) \tag{1}$$

$$Gain(T, A; S) > \frac{log_2(|S| - 1)}{|S|} + \frac{\delta(T, A; S)}{|S|} \quad (2)$$

where

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

and $|S|$ is the number of instances in the given set S, $E(S)$ is the entropy of S, and k_S is the number of classes appeared in S.

If the best cut-point of an interval is accepted by the MDLP criterion, then a recursive discretisation step is applied to each new sub-interval; otherwise, the discretisation process stops.

2.3 Feature Selection via Discretisation

Chi2 [31] is one of the first methods proposing selecting features via discretisation. It is an improvement of the ChiMerge [32] method. ChiMerge is a bottom up method which starts with each interval having one distinct value of the feature. In each iteration, it merges the pair of adjacent intervals with the lowest χ^2 test result. The merging process is continued until all pairs of intervals have χ^2 values exceeding the parameter determined by a predefined significant level. Chi2 has two phases. Phase 1 is a general version of ChiMerge. Instead of using a predefined significant level, Chi2 automatically determines this value from the data by gradually decreasing the significant level from 0.5. Consistency is used as a stopping criterion. After this first phase, each feature has a different significant level. Starting with the significant level determined in phase 1, phase 2 is a fining process used to further merge features in a round robin fashion until the inconsistency of the data above a given limit. At the end, if all intervals of a feature were merged into one, that feature was discarded.

PEAR [33] is also a supervised and univariate discretisation method. It performs simultaneously feature discretisation and selection. In this method, cut-points are chosen for a feature if they lie between instances of different classes and produces intervals in which the majority class has at least a predefined number of instances called minperint. Furthermore, two consecutive intervals should not have the same majority class occurrence and the ratio between these occurrences needs to be higher than a predefined ratio called mintofuse. Then, features are ranked based on the number of cut-points. Features with small numbers of cut points are considered as relevant features and therefore selected. Result on a medical image dataset using 17% best features from a total of 140 showed that it maintained the precision of the full feature set. Its result was better than Relief [34] with the same number of features. However, domain knowledge or a significant number of trials need to be done to choose appropriate values for the parameters.

2.4 Particle Swarm Optimisation

PSO [10] is a population-based algorithm proposed by Kennedy and Eberhart in 1995. In this section, we will describe the standard continuous PSO and its variance, the bare-bone PSO [17].

Continuous PSO. PSO maintains a swarm of particles. These particles "fly" from one position to another in the search space based on the information shared by each other to find better solutions. The solution is represented in the position which is a vector of D real numbers, where D is the dimensionality of the problem. Each particle also has another vector of the same size called velocity showing the speed and direction that the particle should move in each dimension. At each iteration, velocity and position of a particle are adjusted based on the two best positions, one is the best position it has explored so far called *pbest* and the other best is shared from its neighbours called *gbest*. Equations (4) and (5) are used to update these vectors.

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i} * (p_{id}^{t} - x_{id}^{t}) + c_2 * r_{2i} * (p_{gd}^{t} - x_{id}^{t}) \tag{4}$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{5}$$

where v_{id}^{t} and x_{id}^{t} are velocity and position of particle i in dimension d at time t, respectively. p_{id} and p_{gd} are *pbest* and *gbest* positions in dimension d. c_1 and c_2 are acceleration constants, and r_1 and r_2 are random values uniformly distributed in [0, 1]. These constants are important to control the behaviour of the particle [16]. It determines the type of trajectory the particle travels. w is the inertia weight controlling the impact of the last velocity to the current velocity.

Bare-bone PSO. From Equation (4) and (5), we can see that PSO operates by sampling points on the search space. It uses discovered knowledge to guide the search. The position updating reflects how particles select the next point to explore in the search space. To investigate the trajectory of the particle swarm, Kennedy [17] plotted all the points that were visited in one million iterations of a standard PSO where *pbest* and *gbest* were set as constants. The obtained histogram is a tidy bell curve centered midway between these two best positions. This observation suggests that the difference between these two best points, *pbest* and *gbest*, is an important parameter for scaling the amplitude of particle's trajectory. The step size for particles' movement should be a function of consensus between these two points. Therefore, Kennedy [17] proposed bare-bone PSO (BBPSO) which uses a Gaussian sampling based on *pbest* and *gbest* to update particle positions as follows:

$$x_{id}^{t+1} = \begin{cases} \mathcal{N}\left(\frac{p_{id}^{t}+g_{id}^{t}}{2}, |p_{id}^{t} - g_{id}^{t}|\right), & rand() < 0.5 \\ p_{id}^{t}, & \text{otherwise} \end{cases} \tag{6}$$

In Equation (6), $\mathcal{N}(\mu, \sigma)$ is a random number generator using Gaussian distribution with the mean μ centered between *pbest* and *gbest* and the standard deviation σ equal to the absolute difference between them. A probability $rand()$ is used to retain the previous best position *pbest* in order to speed up convergence.

BBPSO eliminates the velocity component of the canonical PSO algorithm. The advantages of this strategy is not only that PSO does not need to optimize another D-dimensional vector (the velocity), but also there is no lag between when an adaptation is needed and when it occurs.

Both PSO and BBPSO have been used for feature selection, which can be seen from [1, 35, 18] (not detailed here due to the page limit). However, they have not been used for feature discretisation or simultaneously feature discretisation and feature selection, which could potentially improved the performance.

3 The Proposed Approach

This section describes the proposed approach for simultaneously feature discretisation and feature selection. To achieve these two tasks, a key component needs to be designed is the particle representation which represents a candidate solution for feature discretisation and selection, and also requires a new updating mechanism.

PSO Representation. As a binary discretisation method, our method evolves one cut-point for each feature. Therefore, a candidate solution, i.e. a particle's position, is encoded as a vector of length D, which is the number of original features in the dataset. Each element in the vector represents a cut-point for the corresponding feature. Therefore, each particle's position is a vector of real numbers (i.e. cut-points) whose values need to be in the range $[Min..Max]$ of the corresponding feature values. Fig. 1 shows an example of a particle's position. During the updating process of the algorithm, if the updated value of a dimension corresponding to feature F is greater than Max_F, then it is set to Max_F. Similarly, it is set to Min_F if the updated value is smaller than Min_F.

Fig. 1. Particle representation of PSO-DFS.

To achieve *feature discretisation*, a feature's original continuous values are compared with the corresponding cut-point value in the particle's position vector, then these continuous values are converted/discretised to either 0 or 1 depending whether their continuous values are larger than the cut-point value or not. To achieve *feature selection*, we consider a feature to be relevant to the target concept if its discrete version has a better discriminating power indicated by an improvement in the classification performance of the learning algorithm. Therefore, if a feature is discretised to a single interval (i.e. all the original continuous values are converted to the same discrete value), which means that it is useless in differentiating instances of different classes, it can be considered irrelevant and should be discarded. In our method, if the cut-point of a feature equals to its minimum or maximum value, it is discretised into one interval. For example, in Fig. 1, Features F_3 and F_5 are not selected because the cut-point of F_3 equals to its maximum value and that of F_5 equals to its minimum value.

To *update the position*, we proposed to use the updating mechanism of the BBPSO instead of standard PSO that is usually used in PSO-based methods for feature selection. In most existing PSO-based feature selection methods, the PSO representation is a vector of real numbers whose values are all in the same range [0, 1] representing the probability to select features. A feature is selected if its probability is greater than a predefined threshold. Therefore, two evolved probabilities, one is slightly greater than the threshold and the other is significantly greater than the threshold, have the same effect on the solution, which may limit the performance of PSO for feature selection. In the new representation for feature discretisation and selection, a different evolved value leads to a different discrete feature. Therefore, the search needs to be fine tuned so that an appropriate value of cut-point can be found. Since BBPSO use a Gaussian random generator to explore new positions between the *pbest* and the *gbest*, it is likely to obtain this behaviour. The new position is sampled around the mean of *pbest* and *gbest* with a standard deviation equal to the absolute difference 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.

PSO Initialisation. To speed up the evolutionary process, each particle's initial position is a random feature subset with the size restricted to 50 for two-class problems and 150 for multi-class problems. These values are taken as suggested in previous studies [36]. For each randomly selected feature, its cut-point is initialised using the best binary cut-point calculated based on MDLP (see Section 2.2) [25]. For those features that are not selected in the initial candidate solutions, their corresponding dimensions will be initialised to the corresponding maximum values. In the initialisation procedure, information gain of the best cut-point is used as a probability to choose features.

Fitness Function. The optimisation process is guided by the classification accuracy. To evaluate a particle, its evolved cut-points are used to discretise features in the training set. Features that are discretised into a single interval, which means all instances have the same feature value, will be discarded. Classification accuracy of the transformed training set is used to evaluate the performance of the particle. By evaluating the cut-points of multiple features together, the joint contribution of all cut-points is taken into account. Since many of these datasets are unbalanced data, fitness values are calculated based on the balanced classification accuracy [37] as follows:

$$fitness = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{|S_i|} \tag{7}$$

where n 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 $1/n$.

The Overall Approach. Fig. 2 shows an overview of the proposed PSO approach to discretisation and feature selection in one stage. Fig. 3 shows the two-stage approach (named PSO-FS) in which features are first discretised and then selected. In both systems, the input dataset is first divided into a training set and a test set. The training set is used to find the discretisation scheme as well as select relevant features. Based on the output scheme, the training and the test sets are transformed as inputs to the classification algorithm to evaluate the performance of both methods. The pseudo-code of PSO-DFS is presented in Algorithm 1.

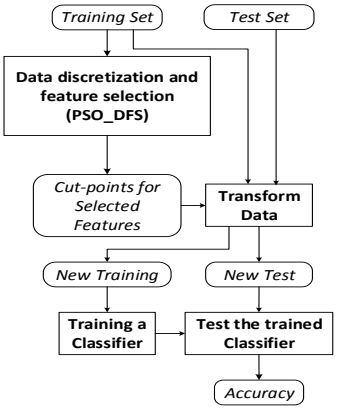

Fig. 2. Overview of the PSO-DFS system.

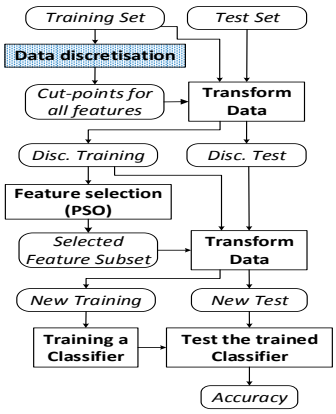

Fig. 3. Overview of the PSO-FS system.

Algorithm 1: The pseudo code of PSO-DFS

```
Input  : Training set with continuous data
Output: Cut-points for the selected features
begin
    Calculate the best binary cut-point for each feature based on the MDLP principle [25];
    Initialise particles;
    while Maximum iterations or stopping criterion is not met do
        for i = 1 to Population Size do
            P_i ← Position of particle i;
            Tr_i ← Transform training set based on P_i;
            F_i ← Evaluate the accuracy of Tr_i using Eq. (7);
            if F_i is better than pbest's fitness then
                Update pbest ;
            end
        end
        Update gbest of the swarm;
        for i = 1 to Population Size do
            for j = 1 to Dimensionality do
                Update dimension j of particle i's position using BBPSO updating
                mechanism as shown in Eq. (6) ;
            end
        end
    end
    Return cut-points for the selected features from gbest's position;
end
```

Table 1. Datasets

Dataset	#Features	#Instances	#Classes	%Smallest class	%Largest class
SRBCT	2,308	83	4	13	35
DLBCL	5,469	77	2	25	75
9Tumor	5,726	60	9	3	15
Leukemia 1	5,327	72	3	13	53
Brain Tumor 1	5,920	90	5	4	67
Leukemia 2	11,225	72	3	28	39
Brain Tumor 2	10,367	50	4	14	30
Prostate	10,509	102	2	49	51
Lung Cancer	12,600	203	5	3	68
11Tumor	12,533	174	11	4	16

4 Experiment Design

Datasets. Ten gene expression datasets with thousands of features are used to examine the performance of the proposed method on high-dimensional data. These datasets are publicly available on `http://www.gems-system.org`. Details about these datasets are shown in Table 1.

Baseline Methods. To test the effectiveness of PSO-DFS in data discretisation and feature selection, we compared the classification performance using the PSO-DFS transformed data and the original dataset. To see if combining both feature discretisation and feature selection in a single stage achieves better results than applying them in two stages, we also compared PSO-DFS with using PSO for feature selection on the discretised data i.e. the two-stage method (PSO-FS). In PSO-FS, data is first discretised by MDLP [25], then, PSO runs on the discrete data to find the best feature subset.

Parameter Settings and Termination Criteria. Table 2 shows the parameter settings used in the experiments for both PSO-FS and PSO-DFS. Because the numbers of features in the datasets are quite different, ranging from about two thousand to twelve thousand, the search spaces of these problems are very different. As a result, we set the population size proportional to the number of original features, i.e. $pop\_size = \#features/20$. However, due to the limitation of computer memory, this number is restricted to 300. The stopping criterion is either PSO reached a maximum iteration of 70 or the *gbest* does not improved after 10 iterations.

Table 2. Parameters for PSO

Parameters	Settings
Population Size	#features/20 (restriction to 300)
Maximum iterations	70
Stopping criterion	*gbest* not improved for 10 iterations
Communication topology	Fully connected

Experiment Configuration. Since PSO-DFS is proposed as a wrapper method, classification performance of a specific classification algorithm will be used as a measure to evaluate the particles. In this study, we use k-nearest neighbour (KNN) with k=1 as it is simple, non-parametric and used in previous papers on these datasets [38, 4].

Due to the small numbers of instances in these datasets, two loops of cross validation (CV) are used to avoid feature selection bias as suggested in [5]. The

outer loop uses a stratified 10-fold CV on the whole dataset. One fold is kept as the unseen data to evaluate the performance of each method, and the remaining 9 folds are used to form the training set for feature discretisation and feature selection. In the fitness function, an inner loop of 10-fold CV on the training set is used to evaluate the evolved solution of each particle during the evolutionary process.

Since PSO is a stochastic method, 30 independent runs are executed with different random seeds for each method. Therefore, a totally 300 runs (30 runs x 10 fold CV) are executed for each method on each dataset. Experiment 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 30 runs from each method are compared using Wilcoxon test, a pair-wise statistical significance test, with the significance level 0.05.

Table 3. KNN average results over the 30 independent runs.

Dataset	Method	Size	Best	Mean±Std	Sig. Test
SRBCT	Full	2,308		87.08	-
	PSO-FS	150.00	97.50	91.31 ± 2.71	-
	PSO-DFS	**137.25**	100.00	**96.89 ± 1.64**	
DLBCL	Full	5,469		83.00	-
	PSO-FS	101.84	96.67	80.03 ± 6.13	-
	PSO-DFS	**42.75**	94.17	**85.18 ± 5.46**	
9Tumor	Full	5,726		36.67	-
	PSO-FS	954.99	55.00	45.95 ± 4.93	-
	PSO-DFS	**138.54**	65.00	**58.22 ± 3.12**	
Leuk1	Full	5,327		79.72	-
	PSO-FS	150.00	92.22	81.60 ± 4.72	-
	PSO-DFS	**135.92**	95.56	**93.37 ± 1.83**	
Brain1	Full	5,920		72.08	=
	PSO-FS	317.34	78.75	71.00 ± 3.06	-
	PSO-DFS	**150.73**	79.17	**72.79 ± 3.48**	
Leuk2	Full	11,225		89.44	=
	PSO-FS	150.00	93.89	86.11 ± 3.97	-
	PSO-DFS	**139.94**	94.44	**89.93 ± 2.79**	
Brain2	Full	10,367		62.50	-
	PSO-FS	417.92	82.08	69.11 ± 5.89	=
	PSO-DFS	**152.78**	83.75	**70.76 ± 5.30**	
Prostate	Full	10,509		**85.33**	+
	PSO-FS	777.39	90.33	85.20 ± 2.35	=
	PSO-DFS	**54.93**	90.33	83.74 ± 3.55	
Lung	Full	12,600		78.05	-
	PSO-FS	686.23	85.73	**81.72 ± 2.08**	=
	PSO-DFS	**150.80**	85.58	80.60 ± 2.42	
11Tumor	Full	12,533		71.42	-
	PSO-FS	1,638.84	86.07	**82.62 ± 1.70**	+
	PSO-DFS	**149.93**	83.68	79.29 ± 2.11	

5 Results and Discussions

Table 3 shows the experimental results of PSO-FS and PSO-DFS. "Full" means KNN using the original full set of continuous features. "Size" shows the average number of features selected by each method over the 30 runs. The best, the average and the standard deviation of the test accuracies achieved by each method on each dataset are displayed in the fourth and the fifth columns. The test accuracy is calculated using Equation (7). The smallest subset size and the

best classification accuracy in each dataset are bold. The last column displays the statistical Wilcoxon significance test results of the corresponding method over the proposed method. "+" or "–" means the result is significantly better or worse than the proposed method and "=" means they are similar in the Wilcoxon tests. In other words, the more "–", the better the proposed method.

5.1 PSO-DFS versus Full

According to Table 3, the average number of features selected by PSO-DFS is always the smallest and significantly smaller than the total number of features. Less than 1% of total number of features is selected in Prostate and DLBCL, 1% to 3% in other seven datasets and 6% in SRBCT.

With the discretised and selected features, the classification performance is significantly improved over using all continuous features. An increase of more than 7% in accuracy is achieved on five out of ten datasets and the highest improvement is 22% on 9Tumor. PSO-DFS obtains a similar accuracy as using original features on Brain1 and Leuk2 and 2% lower accuracy on Prostate with much smaller feature subsets (on average 54.93 features from 10,509 features). However, the best accuracies of PSO-DFS on these three datasets are still 7%, 5% and 5% higher than using all features, respectively.

The results indicate that PSO-DFS can simultaneously discretise and select relevant features so that the discriminating power of the feature set is either significantly improved or maintained with a much smaller number of features.

5.2 PSO-DFS versus PSO-FS

It can be observed in Table 3 that PSO-DFS always selects a much smaller number of features than PSO-FS. With the transformed features, PSO-DFS outperforms PSO-FS on six datasets with the highest improvement of about 12% in accuracy on 9Tumor and Leuk1. While PSO-FS degrades the performance of KNN on DLBCL, PSO-DFS still attains a better performance than using the original feature set on this dataset with only half of the number of features. In general, PSO-DFS obtains the highest average accuracy in seven datasets.

The biggest difference between these two methods can be seen in the 9Tumor dataset. While PSO-FS selects 955 features to achieve an improvement of 9% in classification accuracy, PSO-DFS improves 22% accuracy with only 139 features. A similar pattern can be seen in the first six datasets. In Brain2, Prostate and Lung, both methods obtain a similar accuracy. However, PSO-DFS selects less than half of the number of features selected by PSO-FS. In Prostate and Lung, PSO-DFS selects only 54 and 150 features on average compared to 777 and 686 features selected by PSO-FS. In 11Tumor, PSO-DFS attains 3% lower accuracy than PSO-FS with only 149 features while PSO-FS selects more than 1,600 features.

5.3 Further Analysis

To further analyse the performance of PSO-FS and PSO-DFS, we look at the training accuracy or fitness values of the returned solutions in the ten datasets.

Table 4. Fitness of the best solutions obtained by PSOFS and PSO-DFS.

Dataset	Method	Fitness	Dataset	Method	Fitness
SRBCT	PSO-FS	100.00 ± 0.00	Leuk2	PSO-FS	100.00 ± 0.00
	PSO-DFS	100.00 ± 0.00		PSO-DFS	100.00 ± 0.00
DLBCL	PSO-FS	100.00 ± 0.00	Brain2	PSO-FS	99.73 ± 0.12
	PSO-DFS	100.00 ± 0.00		PSO-DFS	98.38 ± 0.27
9Tumor	PSO-FS	97.49 ± 0.23	Prostate	PSO-FS	98.89 ± 0.10
	PSO-DFS	95.03 ± 0.22		PSO-DFS	98.56 ± 0.14
Leuk1	PSO-FS	100.00 ± 0.00	Lung	PSO-FS	97.77 ± 0.05
	PSO-DFS	100.00 ± 0.00		PSO-DFS	97.10 ± 0.14
Brain1	PSO-FS	100.00 ± 0.00	11Tumor	PSO-FS	99.80 ± 0.08
	PSO-DFS	99.33 ± 0.29		PSO-DFS	96.21 ± 0.19

Table 4 shows the average fitness of the best solutions and its standard deviation obtained by both methods in the 30 runs.

It can be seen from the obtained fitness that both methods have converged to the optimal solutions in all runs on 4 datasets, namely SRBCT, DLBCL, Leuk1, and Leuk2. However, as seen in Table 3, PSO-DFS achieves significantly better test accuracy on these four datasets than PSO-FS. This indicates that the solutions evolved by PSO-DFS generalise better to unseen data than those of PSO-FS.

In the other six datasets, neither of the two methods achieves the optimal solutions for the training data. Since both methods stop running if *gbest* fitness does not improve after 10 iterations, they may need a better stopping criteria or a more effective mechanism to jump out of these local optima. In these problems, PSO-DFS obtains a slightly lower fitness with a much smaller feature subset than PSO-FS. The big difference between PSO-DFS and PSO-FS feature set sizes in these cases indicates that PSO-DFS might need to select more features to achieve a better performance.

Comparing the test accuracy in Table 3 and the training accuracy (or fitness) in Table 4, we can see that there is a big gap between these accuracies in most cases with the biggest difference in 9Tumor. This indicates that overfitting has occurred with different levels of effect in different datasets. The reason of this phenomenon is that the features' distribution in these datasets is very skew. Therefore, the training and test sets may have different distributions. The model learned from training data may not be generalised to the test data. This effect is worse in datasets with a small number of instances. Therefore, with only 50 and 60 instances, Brain2 and 9Tumor are the most affected cases. In addition, the class imbalance issue in these datasets also makes them challenging problems. With 9 classes, 9Tumor has worse results than Brain2 which has 4 classes.

In general, the results show that the proposed approach of combining data discretisation and feature selection in one stage performs better than separate these two steps in different stages. PSO-DFS can create a more compact and better discriminating representation for data than PSO-FS. This confirms our hypothesis that individually discretising features in the first stage of PSO-FS may lose important information including feature interaction. Since PSO-DFS evaluates the cut-points of all features simultaneously, such information is taken into account.

Table 5. Computation time (in minutes) of PSOFS and PSO-DFS.

Dataset	Method	Time	Dataset	Method	Time
	PSO-FS	1.14		PSO-FS	7.85
SRBCT	PSO-DFS	1.60	Leuk2	PSO-DFS	4.95
	PSO-FS	3.69		PSO-FS	6.95
DLBCL	PSO-DFS	2.04	Brain2	PSO-DFS	2.95
	PSO-FS	16.74		PSO-FS	30.23
9Tumor	PSO-DFS	5.09	Prostate	PSO-DFS	4.29
	PSO-FS	3.01		PSO-FS	129.20
Leuk1	PSO-DFS	3.61	Lung	PSO-DFS	17.01
	PSO-FS	10.14		PSO-FS	192.11
Brain1	PSO-DFS	5.09	11Tumor	PSO-DFS	13.00

5.4 Computation Time

The average time in minutes to complete one run for PSO-FS and PSO-DFS is shown in Table 5. Using wrapper approach, both methods use the classification accuracy of KNN classifier running with 10-fold CV on the training data to guide the search. It is noticed that while PSO-FS only needs to transform the training set based on the selected features, PSO-DFS has to do both discretisation and selection in every particle's evaluation process. Therefore, its running time was expected to be higher than PSO-FS. However, the observation from Table 5 reflects an opposite trend. Compared to PSO-FS, PSO-DFS has a lower running time on eight datasets. PSO-DFS spends less than half of the running time used by PSO-FS on most datasets with only about one tenth on Lung and 11Tumor. A detailed inspection of the evolutionary process revealed that this is because PSO-DFS selected a significant smaller number of features than PSO-FS.

6 Conclusions and Future Work

This paper proposes a new PSO-based method for feature discretisation and feature selection in a single stage. To achieve feature discretisation, a new PSO encoding scheme is proposed to evolve cut-points for multiple features simultaneously. Feature selection is accomplished by removing features that are discretised into only one interval. PSO-DFS is tested and compared with the two-stage approach, PSO-FS, in which features are individually discretised and then selected.

Experimental results on ten high-dimensional datasets show that PSO-DFS can effectively discretise multiple features to significantly improve or maintain the classification performance on most cases. Through discretisation, a much smaller number of relevant features is selected at the same time. Comparison between PSO-FS and PSO-DFS shows that conducting feature discretisation and feature selection in a single stage is more effective than applying these techniques in two different stages.

Although PSO-DFS obtained better solutions than PSO-FS for preprocessing data, it may still get stuck in local optima. Overcoming this problem enables PSO-DFS to achieve even better solutions. Multiple interval discretisation is another promising direction to investigate. Our future work will focus on these directions.

References

1. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering **40** (2014) 16–28
2. Xue, B., Cervante, L., Shang, L., Browne, W., Zhang, M.: A multi-objective particle swarm optimisation for filter-based feature selection in classification problems. Connection Science **24** (2012) 91–116
3. Ferreira, A.J., Figueiredo, M.A.: Efficient feature selection filters for high-dimensional data. Pattern Recognition Letters **33** (2012) 1794–1804
4. Tran, B., Xue, B., Zhang, M.: Improved PSO for Feature Selection on High-Dimensional Datasets. In: Simulated Evolution and Learning. Volume 8886 of Lecture Notes in Computer Science. Springer (2014) 503–515
5. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97** (1997) 273–324
6. Ding, C., Peng, H.: Minimum redundancy feature selection from microarray gene expression data. Journal of bioinformatics and computational biology **3** (2005) 185–205
7. Dougherty, J., Kohavi, R., Sahami, M., et al.: Supervised and unsupervised discretization of continuous features. In: Machine learning: proceedings of the twelfth international conference. Volume 12. (1995) 194–202
8. Ferreira, A.J., Figueiredo, M.A.: An unsupervised approach to feature discretization and selection. Pattern Recognition **45** (2012) 3048–3060
9. Chao, S., Li, Y.: Multivariate interdependent discretization for continuous attribute. In: Third International Conference on Information Technology and Applications. Volume 1., IEEE (2005) 167–172
10. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. (1995) 39–43
11. Xue, B., Zhang, M., Browne, W.: Particle swarm optimization for feature selection in classification: A multi-objective approach. IEEE Transactions on Cybernetics **43** (2013) 1656–1671
12. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. Applied Soft Computing **18** (2014) 261–276
13. Cervante, L., Xue, B., Zhang, M., Shang, L.: Binary particle swarm optimisation for feature selection: A filter based approach. In: IEEE Congress on Evolutionary Computation (CEC'12). (2012) 881–888
14. Mohamad, M., Omatu, S., Deris, S., 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** (2011) 813–822
15. Zhou, W., Dickerson, J.A.: A novel class dependent feature selection method for cancer biomarker discovery. Computers in biology and medicine **47** (2014) 66–75
16. Van den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Information sciences **176** (2006) 937–971
17. Kennedy, J.: Bare bones particle swarms. In: Proceedings of IEEE Swarm Intelligence Symposium. (SIS'03), IEEE (2003) 80–87
18. Zhang, Y., Gong, D., Hu, Y., Zhang, W.: Feature selection algorithm based on bare bones particle swarm optimization. Neurocomputing **148** (2015) 150 – 157
19. Garcia, S., Luengo, J., Sáez, J.A., Lopez, V., Herrera, F.: A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Transactions on Knowledge and Data Engineering **25** (2013) 734–750

20. Liu, H., Hussain, F., Tan, C.L., Dash, M.: Discretization: An enabling technique. Data mining and knowledge discovery **6** (2002) 393–423
21. Kotsiantis, S., Kanellopoulos, D.: Discretization techniques: A recent survey. GESTS International Transactions on Computer Science and Engineering **32** (2006) 47–58
22. Catlett, J.: On changing continuous attributes into ordered discrete attributes. In: Machine learning–EWSL-91, Springer (1991) 164–178
23. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine learning **11** (1993) 63–90
24. Grzymala-Busse, J.W.: Discretization based on entropy and multiple scanning. Entropy **15** (2013) 1486–1502
25. Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. Machine Learning (1993)
26. Cano, A., Nguyen, D.T., Ventura, S., Cios, K.J.: ur-caim: improved caim discretization for unbalanced and balanced data. Soft Computing (2014) 1–16
27. Yang, P., Li, J.S., Huang, Y.X.: Hdd: a hypercube division-based algorithm for discretisation. International Journal of Systems Science **42** (2011) 557–566
28. Flores, J.L., Inza, I., Larrañaga, P.: Wrapper discretization by means of estimation of distribution algorithms. Intelligent Data Analysis **11** (2007) 525–545
29. Ramirez-Gallego, S., Garcia, S., Benitez, J.M., Herrera, F.: Multivariate discretization based on evolutionary cut points selection for classification. IEEE Transactions on Cybernetics (2015)
30. Mahanta, P., Ahmed, H.A., Kalita, J.K., 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, ACM (2012) 69–75
31. Liu, H., Setiono, R.: Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of the Seventh International Conference on Tools with Artificial Intelligence. TAI '95, IEEE Computer Society (1995) 88
32. Kerber, R.: Chimerge: Discretization of numeric attributes. In: Proceedings of the tenth national conference on Artificial intelligence, Aaai Press (1992) 123–128
33. Jaba Sheela, L., 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** (2009) 154–158
34. Kira, K., Rendell, L.A.: The feature selection problem: Traditional methods and a new algorithm. In: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press (1992) 129–134
35. Tran, B., Xue, B., Zhang, M.: Overview of Particle Swarm Optimisation for Feature Selection in Classification. In: Simulated Evolution and Learning. Volume 8886 of Lecture Notes in Computer Science. Springer (2014) 605–617
36. Zhu, Z., Ong, Y.S., Dash, M.: Markov blanket-embedded genetic algorithm for gene selection. Pattern Recognition **40** (2007) 3236–3248
37. Patterson, G., Zhang, M.: Fitness functions in genetic programming for classification with unbalanced data. In: AI 2007: Advances in Artificial Intelligence. Springer (2007) 769–775
38. Chuang, L.Y., Chang, H.W., Tu, C.J., Yang, C.H.: Improved binary PSO for feature selection using gene expression data. Computational Biology and Chemistry **32** (2008) 29–38