Memetic Computing manuscript No.
(will be inserted by the editor)

PSO with Surrogate Models for Feature Selection: Static and Dynamic Clustering-based Methods

Hoai Bach Nguyen · Bing Xue · Peter Andreae

Received: date / Accepted: date

Abstract Feature selection is an important but often expensive process, especially with a large number of instances. This problem can be addressed by using a small training set, i.e. a surrogate set. In this work, we propose to use a hierarchical clustering method to build various surrogate sets, which allows to analyze the effect of surrogate sets with different qualities and quantities on the feature subsets. Further, a dynamic surrogate model is proposed to automatically adjust surrogate sets for different datasets. Based on this idea, a feature selection system is developed using particle swarm optimization as the search mechanism. The experiments show that the hierarchical clustering method can build better surrogate sets to reduce the computational time, improve the feature selection performance, and alleviate overfitting. The dynamic method can automatically choose suitable surrogate sets to further improve the classification accuracy.

Keywords Surrogate Model · Feature Selection · Particle Swarm Optimization · Clustering · Classification

1 Introduction

Real-world machine learning problems are described by a large number of features but many of them negatively affect the learning performance. Feature selection aims to improve the quality of feature sets by selecting a small number of relevant features. Therefore, feature selection can reduce the dimensionality to avoid the "curse of dimensionality" (Friedman et al, 2001), leading to a better learning performance, faster training process and simpler learned model. This work focuses on feature selection for classification (Tang et al, 2014).

Feature selection (Guyon and Elisseeff, 2003) is a challenging task because of its large search space. An exhaustive search guarantees an optimal feature subset but is impractical in most situations. Evolutionary computation (EC) has been widely applied to feature selection because of its potential global search ability, especially genetic algorithms (GAs) and particle swarm optimization (PSO). In comparison with GAs, PSO has fewer parameters and is usually more efficient and effective in some areas (Eberhart and Shi, 1998). In GAs, the crossover and mutation operators contribute to its convergence to the optimal solution, but the two operators without a careful design might potentially break good groups of complementary features when solving feature selection problems. Therefore, PSO is used as a search method in this work.

Complex feature interactions also make feature selection a challenging task. A good fitness function, which measures feature subsets' qualities, should be able to capture feature interactions. According to evaluation criteria, feature selection can be divided into two categories: wrapper and filter approaches. In filters, feature subsets are evaluated based on the characteristics of the data, which is independent of any classification algorithm. However, most filter measures only cope with either continuous or discrete datasets. In addition, it is usually difficult for a filter measure to detect multi-way feature interactions. Nguyen et al (2016) attempt

Hoai Bach Nguyen
E-mail: Hoai.Bach.Nguyen@ecs.vuw.ac.nz

Bing Xue
E-mail: Bing.Xue@ecs.vuw.ac.nz

Peter Andreae
E-mail: Peter.Andreae@ecs.vuw.ac.nz

Evolutionary Computation Research Group, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

to compute multi-variate mutual information between a set of features, which results in promising results but with a high computation cost. Wrappers use a classification algorithm to evaluate feature subsets, which ensures to consider feature interactions. Therefore, wrappers usually achieve better classification performance than filters. In this work, we will use a wrapper approach to achieve feature selection.

In terms of computation cost, wrappers are usually more expensive than filters due to involving classification processes. This problem is alleviated in our previous work (Nguyen et al, 2017b), where a surrogate model for a PSO-based wrapper feature selection is proposed. In particular, an instance selection algorithm called DROP3 (Olvera-López et al, 2010) is applied to select a small number of training instances, which forms a surrogate training set. The surrogate set is used to quickly locate promising regions A local search based on the surrogate training set is developed to use information from previous iterations to improve the current *gbest*. The results show that although the proposed algorithm is less computationally intensive than using the original training set, it still can achieve similar or better classification performance. Although the initial design of surrogate training sets works well, there are several key factors that need to be further investigated. For example, DROP3 usually requires a nicely distributed training set and may result in missing informative instances or selecting noisy instances. Furthermore, the relationship between the surrogate training set and the whole training set is not fully investigated. We will continue our previous work on surrogate models for feature selection to address the above issues.

Goal: In this study, based on the previous work (Nguyen et al, 2017b), we aim to improve and further investigate the surrogate model for feature selection, which is expected to increase the classification accuracy while still having a low computation cost. Particularly, a clustering algorithm is utilized instead of DROP3 to produce a better surrogate training set. Furthermore, the relationship between surrogate and full training sets is also investigated, from which a dynamic surrogate model is proposed so that it can adapt with different datasets to select small number of features with high discriminating abilities. Specifically, we will investigate the following questions:

- whether applying the clustering algorithm can improve the qualities of selected features over using DROP3,
- whether a bigger surrogate training set can lead to the better evolved feature subsets. Note that when the surrogate training set's size is increased, it will be more similar to the original training set, and
- whether the proposed dynamic surrogate model can rely on characteristics of datasets to select suitable surrogate training sets, which helps to evolve better feature subsets in a short training time.

2 Background

2.1 Particle swarm optimization

In 1995, particle swarm optimization (PSO) (Kennedy, 2011) is proposed, which is inspired by social behaviors of bird flocking. The underlying principal of PSO is the knowledge sharing between particles to guide the swarm towards optimal points. Each particle has its own position and velocity in the search space. The velocity of a particle is calculated based on its previous velocity (*momentum*), *pbest*, which is its own best position (*cognitive*) and *gbest*, which is the best position discovered by its neighbors including itself (*social*).

PSO is originally developed to optimize continuous problems. Although it is extended to cope with binary problems (Kennedy and Eberhart, 1997), its performance is still limited in comparison with the continuous one (Xue et al, 2012). Nguyen et al (2017a) propose a binary PSO called sticky binary PSO (SBPSO). In SBPSO, the velocity is a probability vector determining the flipping probability of position entries. The *momentum* is redefined as the tendency to stick with the current position, also known as *stickiness* property (*stk*). The *stk* is linearly decreased until it is 0 or the corresponding entry is flipped. The stickiness property of the d^{th} entry is updated by the following equation:

$$stk_d^{t+1} = \begin{cases} 1, \textbf{ if } \text{the bit is just flipped} \\ \max(stk_d^t - \dfrac{1}{ustkS}, 0), \textbf{ otherwise} \end{cases} \quad (1)$$

where t is the i^{th} iteration and $ustkS$ is a number of iterations to reduce *stk* from 1 to 0.

Based on *stk*, flipping probabilities and position entries of a particles are defined as in Eqs. (2) and (3).

$$p_d = i_s * (1 - stk_d) + i_p * |pbest_d - x_d| + i_g * |gbest_d - x_d| \quad (2)$$

where i_s, i_p and i_g are the importance of *stickiness*, *cognitive* and *social* factor.

$$x_d^{t+1} = \begin{cases} 1 - x_d^t & \textbf{, if } rand() < p_d \\ x_d^t & \textbf{, otherwise} \end{cases} \quad (3)$$

The experimental results show that SBPSO is more efficiently and evolves better solutions than standard BPSO and probability-based BPSO (Xue et al, 2014) on feature selection. Therefore, SBPSO is used as the search mechanism in this work.

2.2 Related work on feature selection

Feature selection is a difficult task because of its large search space and complex feature interactions. Although exhaustive search guarantees the best feature subset, it is infeasible in most cases due to its extremely high computation cost. Several techniques have been developed to reduce the computation cost such as greedy searches (Li et al, 2014), sequential searches (Whitney, 1971; Marill and Green, 1963; Niu, 2017), which consider only one feature each iteration. Therefore, they usually suffer from stagnation in local optima.

EC has been widely applied to feature selection because of its potential global search ability. GAs is the earliest EC algorithm used to achieve feature selection (Siedlecki and Sklansky, 1989; Jiang et al, 2017) because of its natural representation. Besides GAs, genetic programming (GP) can simultaneously perform feature selection and build a classifier (Muni et al, 2006). In addition, GP is suitable for some machine learning tasks, such as regression (Koza, 1999; Chen et al, 2017). Recently, PSO gains more attention from feature selection community (Banka and Dara, 2015; Xue et al, 2016) because of its efficiency and effectiveness. However, since EC is a population-based optimization family, EC algorithms usually require a large number of evaluations. Therefore, EC-based feature selection algorithms are usually computationally intensive. In order to improve the efficiency, many filter measures are used in EC-based feature selection (Neshatian et al, 2012; Nguyen et al, 2016; Chinnaswamy and Srinivasan, 2016).

There is not much attempt to reduce the computation costs of wrapper EC-based feature selection. Nguyen et al (2015) improves the efficiency of PSO-based feature selection by shortening the length of particles. Although the computation time is reduced, the evaluation time mainly remains the same as standard PSO-based feature selection. The improvement is from the updating position process and the upper bound of the number of selected features. Wang and Liang (2016) directly modify the fitness measure by splitting a training set into many subsets. From each subset, a number of features are selected and then all selected features are combined to form the final feature subset. However, it is possible that features selected from different subsets might be redundant. In our previous work (Nguyen et al, 2017b), we use an instance selection algorithm to form a surrogate training set, which has fewer instances than the original training set. Therefore, the computation cost is significantly reduced since the evaluation time is much shorter. In this work, we will investigate more on the surrogate model by improving its quality using a clustering algorithm. In addition, different static surrogate models with various numbers of instances and a proposed dynamic surrogate model are examined to analyze the effect of surrogate training sets.

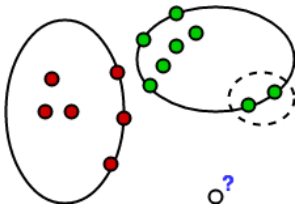

Fig. 1: DROP3 may remove informative instances.

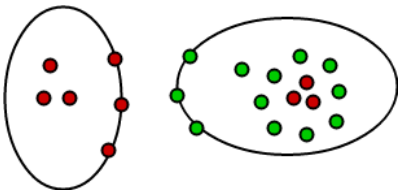

Fig. 2: DROP3 cannot remove noisy instances.

3 Methodology

In this section, we firstly describe DROP3 and its limitations. We then propose how to use a clustering algorithm to address the limitations. We also propose a dynamic surrogate model for feature selection, which is expected to capture characteristics of datasets to evolve better feature subsets.

3.1 DROP3 and its limitations

In our previous work (Nguyen et al, 2017b), DROP3 is used to form surrogate training sets. In the first step, DROP3 removes all instances that are wrongly classified by its K nearest neighbors, which is expected to remove noisy instances. After that, all instances are sorted according to their closest distances to instances from other classes. The instance with a larger distance is considered to be removed earlier since they may be far from its class boundary. An instance is removed if discarding it does not wrongly classify other instances that take the instance as their neighbors. The main idea of DROP3 is to preserve all instances on class boundaries and remove all inner instances, which requires the training set being nicely distributed. Therefore, it is possible that DROP3 removes informative instances or remains noisy instances. Let consider two examples given in Figs. 1 and 2, where there are two class labels (marked by red and green) and a KNN classification algorithm is used with K = 3. In Fig. 1, the two green ones inside the dotted circle have the largest distances to instances from other classes, which means that they are considered to be removed first. It is obvious that removing the two instances does not affect any other green instances since they are too far from the two instances. Therefore, DROP3 will remove them despite

they are on the class boundary. The consequence can be seen in classifying an unseen instance (marked by a question mark). If the two green instances are not removed, it will be classified as a green instance, but removing them changes the class label of the unseen instance. Thus DROP3 removes informative instances. Fig. 2 gives an example where DROP3 is not able to remove noisy instances. It can be seen that in Fig. 2, there are three noisy red instances located inside the region of the green class. The distances from the three red instances to the green class are definitely smaller than any other red instances, so according to DROP3 they are likely to be on the class boundary. In addition, removing one of the three red instances will wrongly classified the other two, so none of them is discarded by DROP3. It can be seen that even DROP3 is designed specifically for KNN, it may result in a poor surrogate training set. In the next section, we will show how a clustering algorithm can address DROP3's problems.

3.2 Clustering-based surrogate model

The problem of DROP3 is that it considers instances from the same class as an instance group despite they may be far from each other as shown in Figs. 1 and 2. On the other hand, a clustering algorithm divides instances from the same class into many clusters. Therefore, in Fig. 1, the two marked green instances are likely to be grouped to a cluster, which ensures that information from the two instances is preserved.

A representative is formed for each cluster, which will contribute as one instance into the surrogate training set. Therefore, the size of surrogate training set is equal to the number of clusters. In this work, the centroid of a cluster, which is the instance closest to the cluster's mean, is selected as the representative. The first reason for selecting the centroid is to ensure a fair comparison with DROP3, which also selects original instances to form the surrogate training set. The second is that using original instances can preserve the relationships/interactions between features while building a new instance from a cluster is more likely to construct new feature interactions, which do not exist in test sets. Note that a cluster may not be pure, which means that it may contain instances from different classes. Therefore, only instances from the majority class, which contributes the largest number of instances in the cluster, are used to select the representative for the cluster. Hence in Fig. 2, the three noisy red instances along with their surrounding green instances are grouped in the same cluster and the noisy instances will be removed since the red class is the minority one in this cluster.

The question is which clustering algorithm should be used. K-means is proposed about 50 years ago and

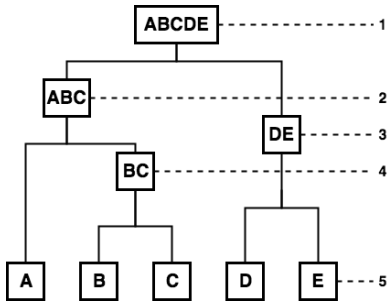

Fig. 3: Example of the agglomerative clustering algorithm.

widely used in clustering (MacQueen et al, 1967), which may be a good option. However, the main task of this work is to analyze how surrogate training sets with different sizes affect performances of the selected feature subset. Therefore, K-means has to be run many times with different numbers of clusters, which is time consuming. Agglomerative clustering (AGG) (Murtagh and Legendre, 2014) is a bottom-up hierarchical clustering algorithm in which each instance starts with its own cluster. When moving up the hierarchy, the two closest clusters are merged into one cluster. An example of the agglomerative clustering algorithm is given in Fig. 3.

Note that although both DROP3 and AGG are deterministic algorithms, they have very different outputs and behaviors. DROP3 directly produces a unique surrogate training set for each dataset. On the other hand, AGG results in a set of possible clustering partitions, whose numbers of clusters ($\#c$) can be from 1 to the total number of instances in the original training set (as shown in Fig. 3). If $\#c$ is decided, AGG produces only one unique clustering partition containing a unique set of clusters. Since only the centroid instance is selected from each cluster, the size of surrogate set formed by AGG is equal to the number of clusters $\#c$.

3.3 Dynamic clustering-based surrogate model

In SBPSO-based feature selection, the position of each particle is a binary vector, in which each entry corresponds to an original feature. If the entry's value is 1, the corresponding feature is selected. Otherwise, the feature is discarded. Therefore, each particle defines a feature subset which is evaluated according to the following fitness function:

$$fitness = \alpha * Error + (1-\alpha) * \frac{\#selectedFeatures}{\#originalFeatures} \quad (4)$$

where $Error$ is the classification error rate, which can be measured by either the surrogate training set or the original training set, α is used to control the contributions of the two objectives. From now, if $Error$ is measured by the original training set, the fitness value

is called *real fitness*. If it is measured by the surrogate training set, the fitness value is called *surrogate fitness*.

Since the surrogate training set is used to estimate possible good regions, it is important to ensure that the surrogate fitness value should be consistent with the real fitness value. For example, if a feature subset A is better than a feature subset B in terms of the surrogate fitness value, A should also be better than B when they are evaluated by the original training set. However, since feature subsets are changed during the evolutionary process, the consistency between the surrogate training set and the original training set may not be preserved. To address this problem, a dynamic clustering-based surrogate model is proposed. The task can be described as: "Given a pool of surrogate training sets, $P = \{S_1, S_2, ..., S_m\}$, which surrogate training set S_i should be used to evaluate feature subsets."

In the initialization process, each particle is randomly initialized. After evaluating the particles using the original training set S_0, the position x_{best} with the best real fitness value f_0 is recorded to find out the most suitable surrogate training set. Particularly, x_{best} is evaluated on m surrogate training sets, which results in m surrogate fitness values $\{f_1, f_2, ..., f_m\}$. The surrogate training set, which has the smallest difference in comparison with the original training set, i.e. the smallest $|f_i - f_0|$, is used to evaluate feature subsets in the following iterations. Hence, even in the initialization step, the surrogate training set is dynamically determined based on its consistency with the original training set.

In the first I_s iterations, feature subsets are evaluated by the surrogate training set, which is also dynamically updated to preserve the consistency with the original training set. However, updating the surrogate set too frequently makes PSO more difficult to adapt with changes in the fitness landscape. Therefore, the surrogate one is only updated when the real fitness value of *gbest* is not improved for a certain number of iterations ($NIStep$). The process of finding the most suitable surrogate training set is similar to the method used in the initialization process, except for x_{best} is replaced by *gbest*. After the surrogate process, i.e. the first I_s iterations, is finished, the original training set is used to evaluate the candidate solutions. The pseudo-code of the dynamic surrogate model is given in Algorithm 1.

Algorithm 1 Dynamic surrogate model

```
Input: A pool of surrogate training sets, P = {S_1, S_2, ..., S_m}, built by AGG.
randomly initialize the PSO population;
find the best position x_best in the initialized population;
select the most suitable SurrogateSet in P based on x_best;
while maximum number of iterations is not reached do
    if current iteration is smaller than I_s then
        evaluate particles using SurrogateSet;
        update pbest and gbest for each particle;
        evaluate gbest using the original training set;
        if gbest's real fitness is not improved for NIStep iterations then
            select the most suitable SurrogateSet in P based on gbest;
        end if
    else
        evaluate particles using the original training set;
        update pbest and gbest for each particle;
    end if
    apply sampling local search on gbest
    update velocities and positions of particles;
end while
return gbest as the final feature subset;
```

4 Experiment design

4.1 Datasets

The proposed methods are tested on 12 datasets chosen from the UCI machine learning repository (Lichman, 2013). The datasets are selected so that they have different numbers of features (#Fs), classes and instances, which can be seen in Table 1. Each dataset is divided into training and test sets, so that they contain 70% and 30% instances respectively and the class distribution is roughly preserved.

Table 1: Datasets.

Dataset	#Fs	#Classes	#Instances
Australian	14	2	6650
ImageSegmentation	19	7	210
German	24	2	1000
WBCD	30	2	569
Ionosphere	34	2	351
Sonar	60	2	208
Hillvalley	101	2	1213
Arrhythmia	279	16	452
LSVT	310	2	126
Madelon	500	2	4400
Isolet5	617	5	7797
Multiple Features	649	10	2000

In the experiments, the performances of different surrogate training sets are examined. Firstly, the DROP3 algorithm and the agglomerative clustering algorithm are compared. To ensure a fair comparison, the number of clusters in the clustering algorithm is equal to the number of instances selected by DROP3. Therefore, the two algorithms result in two surrogate training sets with the same size, which are called "DROP3" and "AGG", respectively. Since it was shown in our previous work (Nguyen et al, 2017b) that the surrogate model built by DROP3 was already better than an improved version of sequential feature selection search, AGG is not compared with sequential searches due mainly to the page limit.

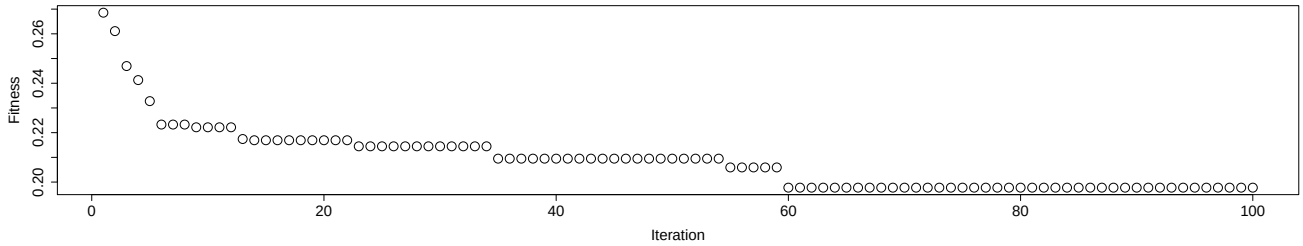

Fig. 4: Evolutionary process of PSO on the Madelon dataset

Table 2: Compare different I_s values against $I_s = 75$.

Dataset	$I_s = 0$	$I_s = 25$	$I_s = 50$	$I_s = 100$
German	+	=	=	+
Sonar	=	=	=	=
Arrhythmia	-	=	=	+
Isolet5	=	=	=	+

However, DROP3 usually selects a very small number of instances. For example, on Arrhythmia, the surrogate set built by DROP3 contains only 26 instances, which is even 10 times smaller than the total number of features. Therefore, we decide to examine surrogate training sets produced by the clustering algorithm with various numbers of instances. In particular, five surrogate training sets, whose sizes range from 10% to 50% of the original training set, are generated. The lower bound 10% is to ensure that the surrogate training sets contain enough training instances. The upper bound is set to 50% so that the surrogate model still can significantly reduce the computation cost over using the original training set. The five surrogate training sets form a training set pool, from which the dynamic surrogate model picks the most suitable training set during the surrogate process. The percentages i.e "10%", "20%", "30%", "40%" and "50%" are used to name methods with corresponding surrogate training sets, while the dynamic surrogate model is called "Dynamic".

4.2 Parameter settings

The feature subsets are evaluated using a KNN classification algorithm, where K is set to 5 so that it is able to avoid issues caused by noisy instances while still has good efficiency. α in Eq. (4) is set to 0.9 to ensure that the classification performance has higher priority than the number of selected features. For SBPSO, i_m, i_p, i_g, and $ustkS$ are set to 0.1154, 0.4423,0.4423, and 40, respectively, as suggested by results in Nguyen et al (2017a). 5 different values of I_s ranging from 0 to 100 are examined and the results show that 75 is the most suitable setting. Table 2 shows results of statistical significance tests, which compare between the value 75 and the other four different values of I_s on four datasets with different numbers of features. "+"/ "=" / "-" means that 75 is significantly better/ similar or worse than the other values. It can be seen that the value of 75 achieves similar or better performance than the three smaller values (0, 25, 50) while being less computationally intensive. In addition, the value 75 is significantly better than 100 i.e. using only the surrogate training sets. The population size is set to the number of features and limited by 100. The maximum number of iterations is set to 100. $NIStep$ is set to 5 as an indication that the algorithms might be trapped in local optima. An evolutionary process of PSO on the Madelon dataset is shown in Fig. 4. It can be seen that if the *gbest*'s fitness value (vertical axis) is not changed for more than 5 iterations, it is very likely that the fitness value is not changed in the following iterations.

5 Results and discussions

5.1 Effect of applying the clustering algorithm

Table 3 shows the comparison between applying DROP3 and the agglomerative (AGG) clustering algorithm. In the table, "#Features" means the number of selected features, "Training" and "Testing" represent the training and testing accuracies, respectively. Note that both DROP3 and AGG use the same number of instances to build surrogate training sets. The two models are compared using Wilcoxon test, a significance signed rank test with significance level set to 0.05. "↑" or "↓" means that AGG is significantly better or worse than DROP3, while "○" indicates that there is no significant difference between the two algorithms. In terms of training accuracy, AGG is significantly better than DROP3 on 4 datasets while being worse only on German. On the test set, the feature subsets selected by AGG are never worse than the ones using DROP3. On 3 out of the 12 datasets, AGG's accuracies are significantly higher than DROP3's. In addition, the feature subsets selected by AGG are smaller than the ones of DROP3 on 6 datasets. On the other 4 datasets, the two algorithms select the similar number of features. The experimental results show that given the same number of selected instances, AGG can maintain more informative instances to form more consistent surrogate training sets, which results

Table 3: Comparison between DROP3 and Agglomerative Clustering algorithms.

Dataset	#Features		Training		Testing		Time	
	DROP3	AGG	DROP3	AGG	DROP3	AGG	DROP3	AGG
Australian	2.500(↓)	2.800	77.34(○)	81.52	76.14(○)	80.49	0.26	0.29
ImageSegmentation	4.000(↑)	3.400	96.64(○)	96.46	94.89(○)	95.33	0.05	0.05
German	5.300(↑)	3.600	78.36(↓)	75.60	69.16(○)	69.37	0.95	1.00
WBCD	2.000(○)	2.000	94.64(↑)	95.18	93.18(↑)	94.54	0.34	0.38
Ionosphere	3.300(○)	3.500	93.97(○)	93.76	86.31(↑)	87.68	0.16	0.18
Sonar	10.20(○)	11.10	89.77(↑)	91.24	78.84(○)	77.78	0.14	0.15
Hillvalley	22.30(↑)	15.50	74.37(○)	74.65	58.55(○)	59.07	7.06	7.35
Arrhythmia	26.20(↓)	33.40	95.88(○)	95.92	94.94(○)	94.90	0.84	1.04
LSVT	31.20(↑)	27.50	87.31(○)	86.55	66.58(○)	67.63	0.09	0.09
Madelon	195.3(↑)	152.6	88.99(↑)	89.91	79.64(↑)	81.85	62.19	55.65
Isolet5	98.70(↑)	90.40	99.37(○)	99.38	98.74(○)	98.72	21.36	21.31
Multiple Features	94.00(○)	101.4	99.52(↑)	99.55	99.00(○)	99.01	27.28	32.11

in better classification accuracies. Although AGG and DROP3 use surrogate training sets with the same size, AGG usually selects a smaller number of features. The possible reason is that AGG can remove the outliers, so it selects only necessary features to distinguish instances from different classes. DROP3 may select some noisy instances, which possibly requires additional features to correctly classify them.

5.2 Results of clustering-based surrogate models

The results of clustering-based surrogate models with different sizes of the surrogate set are shown in Table 4. The best classification accuracies on both training and test sets are marked in bold. In comparison with other clustering-based models, AGG can achieve the best performance on only 1 out of the 24 cases (including both training and testing accuracies). In terms of the number of selected features, AGG usually selects a smaller number of features than the other methods. The reason for this pattern is that AGG uses the smallest number of instances, so it does not need to select as many features as the other methods. This is an example of underfitting, where AGG does not have enough instances to select a sufficient number of informative features which are necessary for classifying unseen instances.

As can be seen in Table 4, depending on characteristics of the datasets, the best accuracies are achieved by different sizes of surrogate training sets. Mostly the surrogate training sets ranging from 30% to 50% produce the best accuracies since these training sets are more similar to the original ones. However, on 3 datasets, Australian, ImageSegmentation and WBCD, 10% and 20% achieve the best performance, which may be an indication that the 3 datasets have noisy instances and small size surrogate training sets help to eliminate these instances. An important pattern shown in Table 4 is the consistency between training and testing performance. Specifically, on 6 out of the 12 datasets, both best training and testing accuracies are achieved by the same method. On the other datasets, although the exact consistency does not happen, the method with the best testing accuracy usually has the second best training performance. This pattern shows that to some extent, using surrogate models can help to avoid overfitting.

In order to analyze the condition for a surrogate model to locate good search regions, for each surrogate model (10%-50%), the evolutionary process of the best run is shown in Fig. 5. The horizontal axis is iterations and the vertical axis shows the **real** fitness function of *gbest* on each iteration. Due to the space limitation, only 6 out of the 12 datasets are shown. The evolutionary processes on the other 6 datasets have similar patterns. Note that in the first 75 iterations, particles are evaluated by the surrogate set, which means that in terms of the surrogate fitness, the later *gbest* is always not worse than the earlier *gbest*. However, on the figure, the *gbest* is re-evaluated by using the original training set, which does not guarantee that the later *gbest* has better **real** fitness value. Therefore, the less fluctuating evolutionary process shows that the corresponding surrogate model is more consistent with the original training set. By collating between Table 4 and Fig. 5, it can be seen that usually the method with the least fluctuating evolutionary process yields the best classification accuracy. For example, on the Arrhythmia dataset, the best training and testing accuracies are achieved by the 50% surrogate model, which has the least fluctuating evolutionary process.

5.3 Results of the dynamic surrogate model

As illustrated in Section 5.2, in order to achieve good classification performance, it is important to maintain the consistency between the surrogate and the original training sets. However, which surrogate model should be selected heavily depends on datasets. Therefore, the dynamic surrogate model is designed with an expectation of selecting the most suitable surrogate training set during the evolutionary process. The results of the dynamic one are shown by the "Dyn" column in Table 4. A

Table 4: Results of clustering-based surrogate models.

Dataset	Training accuracy							Time						
	AGG	10%	20%	30%	40%	50%	Dyn	AGG	10%	20%	30%	40%	50%	Dyn
Australian	81.52(○)	**85.42**(○)	82.13(○)	80.78(○)	81.82(○)	79.22(○)	80.15	0.29	0.29	0.29	0.35	0.43	0.47	0.46
ImageSeg	96.46(↑)	96.69(○)	**97.11**(○)	96.95(○)	96.90(○)	97.02(○)	96.94	0.05	0.04	0.04	0.05	0.06	0.07	0.06
German	75.60(↑)	76.77(○)	80.41(↓)	**82.55**(↓)	81.21(↓)	80.97(○)	78.56	1.00	0.95	1.16	1.38	1.69	2.05	1.35
WBCD	95.18(○)	**95.92**(↓)	95.19(○)	94.74(↑)	95.06(○)	95.31(↓)	95.14	0.38	0.37	0.42	0.49	0.59	0.73	0.67
Ionosphere	93.76(○)	93.44(○)	93.57(○)	93.75(○)	**93.89**(○)	**93.89**(○)	93.66	0.18	0.17	0.19	0.23	0.28	0.34	0.30
Sonar	91.24(○)	90.28(↑)	91.13(○)	91.08(○)	**91.38**(○)	91.19(○)	91.08	0.15	0.11	0.12	0.14	0.17	0.21	0.17
Hillvalley	74.65(○)	74.60(○)	74.38(○)	**74.92**(○)	74.48(○)	74.82(○)	74.85	7.35	6.43	7.10	8.35	10.08	12.74	10.78
Arrhythmia	95.92(↑)	95.74(↑)	95.73(↑)	96.06(↑)	96.13(○)	**96.18**(○)	96.17	1.04	22.77	1.28	1.45	1.74	2.15	2.14
LSVT	86.55(↑)	86.55(↑)	85.11(↑)	89.20(○)	88.14(↑)	**89.62**(○)	89.43	0.09	0.08	0.10	0.11	0.14	0.17	0.15
Madelon	89.91(↑)	89.64(↑)	89.82(↑)	90.09(○)	90.47(○)	**90.61**(○)	90.41	55.65	52.65	60.48	66.62	81.18	99.94	91.34
Isolet5	99.38(○)	99.34(↑)	99.35(↑)	99.38(○)	99.38(○)	**99.40**(○)	**99.40**	21.31	16.75	17.68	20.91	25.92	31.72	32.97
MultipleFs	99.55(○)	99.54(○)	99.55(○)	**99.57**(○)	99.55(○)	99.55(○)	99.56	32.11	30.23	50.40	41.05	46.99	54.76	58.74

Dataset	Testing accuracy							#Features						
	AGG	10%	20%	30%	40%	50%	Dyn	AGG	10%	20%	30%	40%	50%	Dyn
Australian	80.49(○)	**82.50**(○)	81.02(○)	79.36(○)	80.45(○)	78.02(○)	78.88	2.8	3.0	2.8	2.8	2.8	2.6	2.6
ImageSeg	95.33(○)	95.09(↑)	**95.97**(○)	95.82(○)	95.86(○)	95.74(○)	95.75	3.4	3.9	3.9	3.8	3.9	3.9	4.0
German	69.37(○)	68.59(○)	68.99(○)	**69.50**(↓)	69.22(○)	69.45(↓)	68.56	3.6	3.7	5.7	6.1	6.0	5.8	4.9
WBCD	94.54(○)	93.39(↑)	93.88(○)	92.71(↑)	93.53(↑)	**94.62**(↓)	94.13	2.0	2.6	2.2	2.4	2.4	2.1	2.2
Ionosphere	87.68(○)	88.09(○)	88.00(○)	**88.25**(○)	87.52(○)	87.65(○)	87.52	3.5	3.6	3.4	3.4	3.6	3.7	3.4
Sonar	77.78(○)	77.88(○)	78.63(○)	**79.74**(○)	79.15(○)	79.52(○)	79.42	11.1	11.6	10.7	12.3	12.8	11.9	13.1
Hillvalley	59.07(○)	58.98(○)	58.44(○)	58.86(○)	**59.14**(○)	58.37(○)	58.88	15.5	20.5	19.2	16.6	16.1	18.4	19.3
Arrhythmia	94.90(○)	94.81(↑)	94.74(↑)	94.94(↑)	95.09(○)	**95.16**(○)	**95.16**	33.4	44.9	43.3	37.7	33.5	28.2	28.3
LSVT	67.63(↑)	67.63(↑)	67.28(↑)	71.84(○)	69.21(↑)	**75.97**(○)	74.74	27.5	27.5	30.6	30.1	30.6	36.0	32.9
Madelon	81.85(↑)	80.80(↑)	81.91(○)	82.65(○)	**83.34**(↓)	82.89(○)	82.64	152.6	169.3	154.2	144.4	124.8	132.9	138.4
Isolet5	**98.72**(○)	98.66(○)	98.67(○)	98.65(○)	98.68(○)	98.70(○)	98.70	90.4	108.0	92.1	90.7	87.5	84.2	85.0
MultipleFs	99.01(○)	99.04(○)	99.04(○)	**99.07**(○)	99.04(○)	99.05(○)	99.06	101.4	112.3	109.5	95.2	92.0	88.6	88.5

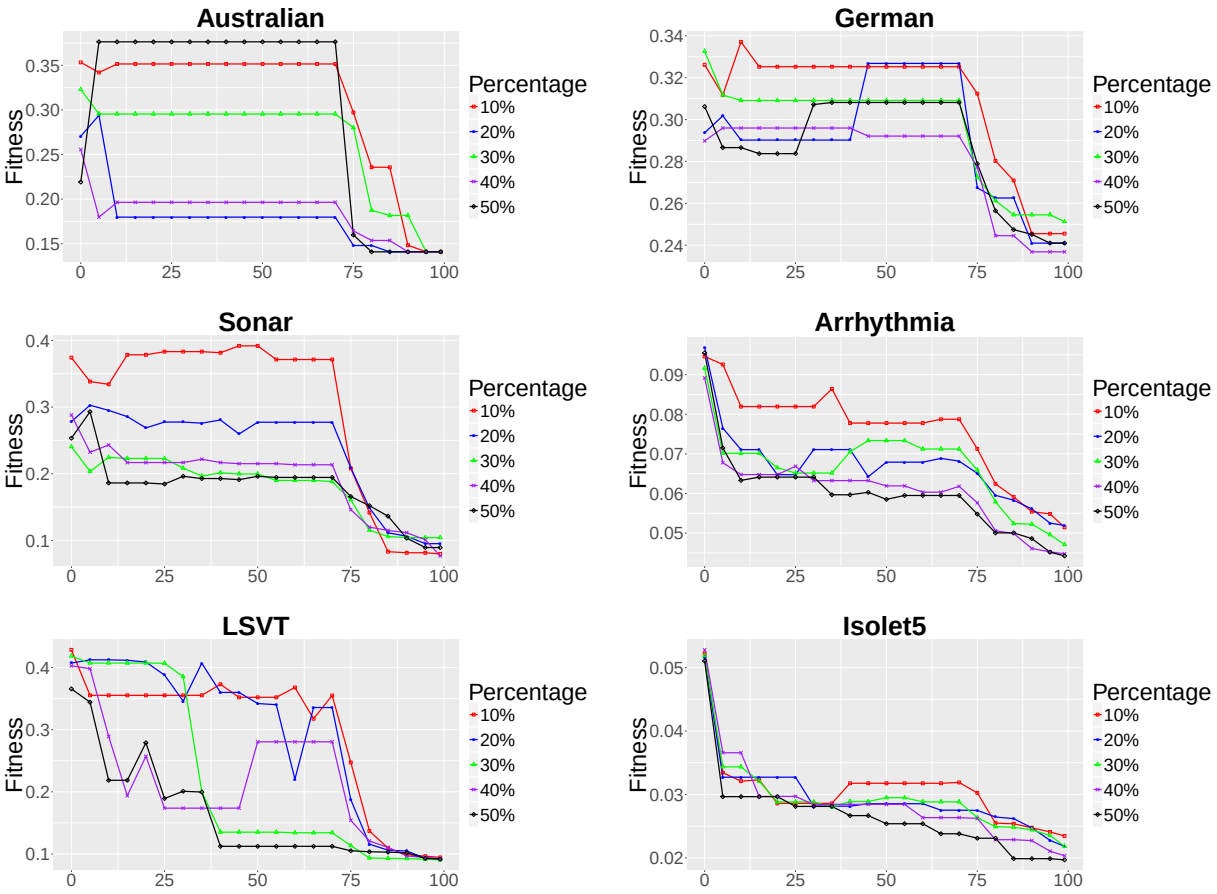

Fig. 5: Real evolutionary processes on different sizes of surrogate training sets.

Table 5: Average ranks of clustering-based surrogate models.

Term	AGG	10%	20%	30%	40%	50%	Dyn
Training	4.58	5.50	4.75	3.59	3.66	2.33	3.57
Testing	4.25	5.17	4.58	3.42	3.50	3.17	3.92

Wilcoxon signed rank test is used to compare between the dynamic and other fixed-size surrogate models. "↑" / "↓" or "○" shows that the dynamic surrogate model is significantly better, worse or no significant difference in comparison with the others.

On training sets, except for German, the dynamic model (Dyn) achieves similar or better performance than the other methods. Specifically, Dyn is significantly better than AGG on 5 datasets. From 10% to 40%, the number of datasets on which Dyn is superior ranges from 5 to 1. Similarly, on the test sets, except for the 50% surrogate, Dyn is never worse than the other models on most datasets. On each dataset, the 7 models are sorted according to their accuracies and their average ranks on all datasets are shown in Table 5. The smaller the rank, the better the method. It can be seen that on the training sets, Dyn is the second best and it is only worse than 50%, which is understandable since the 50% surrogate training set is the most similar to the original training set. However, on the test sets, Dyn is only ranked at the 4^{th} position, which is an indication of overfitting. The possible reason is at the step finding the best suitable surrogate training set (line 10 in Algorithm 1), which can be considered as a local search. In terms of computation time, 50% is the only model which is worse than Dyn. However, in comparison with AGG, Dyn is at most 2 times slower, particularly less than 1.5 times on 9 out of the 12 datasets. It was shown in our previous work (Nguyen et al, 2017b) that the surrogate model built by DROP3 was already 3-4 times less computationally expensive than using the original training set. Given that AGG and DROP3 have the same computation cost, one can say that both static (AGG) and dynamic (DYN) can reduce the computation cost over using the original training set.

The experimental results show that the dynamic model can adapt with different datasets to select the suitable surrogate training set, which results in similar or better performance than other algorithms on most datasets. However, the dynamic model may suffer from the overfitting problem, which results in less general feature subsets than the other algorithms.

6 Conclusions and future work

This work investigates the effect of surrogate models on feature selection. Firstly, the quality of the surrogate training set is improved by a clustering algorithm, which divides the original training set into many clusters. The surrogate training set is formed by selecting a centroid instance as a representative of each cluster. The experimental results show that when selecting the same number of instances, the clustering-based surrogate model can maintain or improve the classification performance while selecting fewer features than the DROP3-based surrogate model on most datasets. In addition, various clustering-based surrogate models with different numbers of instances are examined. The results highlight the importance of selecting enough informative instances to avoid underfitting. It is also shown that to some extent using the surrogate models can improve the generalization of evolved feature subsets. In addition, it is also necessary to maintain the consistency between the surrogate and the original training sets. To ensure the consistency, a dynamic surrogate model is proposed which automatically selects the most suitable surrogate training set during the evolutionary process. The dynamic model can adapt with different datasets to consistently achieve similar or better training accuracies than other static surrogate models.

Although the dynamic model achieves good results, there are issues which we will investigate in the future. For example, the dynamic model may suffer the overfitting problem, which makes its testing accuracies are not as good as its training accuracies. In addition, the clustering-based models have the same size on all datasets. It would be better if the pool of surrogate training sets is designed with respect to the characteristics of datasets. However, it is not an easy task since it requires a deep understanding of each dataset. In the future we will further investigate and develop surrogate models on larger datasets in terms of both number of features and instances.

References

Banka H, Dara S (2015) A hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation. Pattern Recognition Letters 52:94–100

Chen Q, Zhang M, Xue B (2017) Feature selection to improve generalisation of genetic programming for high-dimensional symbolic regression. IEEE Transactions on Evolutionary Computation

Chinnaswamy A, Srinivasan R (2016) Hybrid feature selection using correlation coefficient and particle swarm optimization on microarray gene expression data. In: Innovations in Bio-Inspired Computing and Applications, Springer, pp 229–239

Eberhart RC, Shi Y (1998) Comparison between genetic algorithms and particle swarm optimization. In:

International conference on evolutionary programming, Springer, pp 611–616

Friedman J, Hastie T, Tibshirani R (2001) The elements of statistical learning, vol 1. Springer series in statistics Springer, Berlin

Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. Journal of machine learning research 3(Mar):1157–1182

Jiang S, Chin KS, Wang L, Qu G, Tsui KL (2017) Modified genetic algorithm-based feature selection combined with pre-trained deep neural network for demand forecasting in outpatient department. Expert Systems with Applications 82:216–230

Kennedy J (2011) Particle swarm optimization. In: Encyclopedia of machine learning, Springer, pp 760–766

Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, IEEE, vol 5, pp 4104–4108

Koza JR (1999) Genetic programming III: Darwinian invention and problem solving, vol 3. Morgan Kaufmann

Li Z, Liu J, Yang Y, Zhou X, Lu H (2014) Clustering-guided sparse structural learning for unsupervised feature selection. IEEE Transactions on Knowledge and Data Engineering 26(9):2138–2150

Lichman M (2013) UCI machine learning repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Sciences

MacQueen J, et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA., vol 1, pp 281–297

Marill T, Green DM (1963) On the effectiveness of receptors in recognition systems. IEEE Transactions on Information Theory 9(1):11–17

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

Murtagh F, Legendre P (2014) Wards hierarchical agglomerative clustering method: which algorithms implement wards criterion? Journal of Classification 31(3):274–295

Neshatian K, Zhang M, Andreae P (2012) A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. IEEE Transactions on Evolutionary Computation 16(5):645–661

Nguyen BH, Xue B, Andreae P (2017a) A novel binary particle swarm optimization algorithm and its applications on knapsack and feature selection problems. In: Proceeding of the 20th Asia Pacific Symposium on Intelligent and Evolutionary Systems, Springer, pp 319–332

Nguyen HB, Xue B, Liu I, Andreae P, Zhang M (2015) Gaussian transformation based representation in particle swarm optimisation for feature selection. In: European Conference on the Applications of Evolutionary Computation, Springer, pp 541–553

Nguyen HB, Xue B, Andreae P (2016) Mutual information for feature selection: estimation or counting? Evolutionary Intelligence 9(3):95–110

Nguyen HB, Xue B, Andreae P (2017b) Surrogate-Model Based Particle Swarm Optimisation with Local Search for Feature Selection in Classification, vol 10199, Springer International Publishing, pp 487–505

Niu G (2017) Feature Selection Optimization, Springer Singapore, pp 139–171

Olvera-López JA, Carrasco-Ochoa JA, Martínez-Trinidad JF, Kittler J (2010) A review of instance selection methods. Artificial Intelligence Review 34(2):133–143

Siedlecki W, Sklansky J (1989) A note on genetic algorithms for large-scale feature selection. Pattern recognition letters 10(5):335–347

Tang J, Alelyani S, Liu H (2014) Feature selection for classification: A review. Data Classification: Algorithms and Applications p 37

Wang F, Liang J (2016) An efficient feature selection algorithm for hybrid data. Neurocomputing 193:33–41

Whitney AW (1971) A direct method of nonparametric measurement selection. IEEE Transactions on Computers 100(9):1100–1103

Xue B, Zhang M, Browne WN (2012) Multi-objective particle swarm optimisation (pso) for feature selection. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation, ACM, pp 81–88

Xue B, Nguyen S, Zhang M (2014) A new binary particle swarm optimisation algorithm for feature selection. In: European Conference on the Applications of Evolutionary Computation, Springer, pp 501–513

Xue B, Zhang M, Browne WN, Yao X (2016) A survey on evolutionary computation approaches to feature selection. IEEE Transactions on Evolutionary Computation 20(4):606–626