A Differential Evolution Approach to Feature Selection and Instance Selection

Jiaheng Wang, Bing Xue(✉), Xiaoying Gao, and Mengjie Zhang

School of Engineering and Computer Science, Victoria University of Wellington,
PO Box 600, Wellington 6140, New Zealand
Bing.Xue@ecs.vuw.ac.nz

Abstract. More and more data is being collected due to constant improvements in storage hardware and data collection techniques. The incoming flow of data is so much that data mining techniques cannot keep up with. The data collected often has redundant or irrelevant features/instances that limit classification performance. Feature selection and instance selection are processes that help reduce this problem by eliminating useless data. This paper develops a set of algorithms using Differential Evolution to achieve feature selection, instance selection, and combined feature and instance selection. The reduction of the data, the classification accuracy and the training time are compared with the original data and existing algorithms. Experiments on ten datasets of varying difficulty show that the newly developed algorithms can successfully reduce the size of the data, and maintain or increase the classification performance in most cases. In addition, the computational time is also substantially reduced. This work is the first time for systematically investigating a series of algorithms on feature and/or instance selection in classification and the findings show that instance selection is a much harder task to solve than feature selection, but with effective methods, it can significantly reduce the size of the data and provide great benefit.

Keywords: Differential evolution · Feature selection · Instance selection · Classification

1 Introduction

As hardware technology improves, more and more data is collected at a rate machine learning and data mining techniques cannot deal with. Often the data collected contains redundant or irrelevant features and instances [7,9,14,22,25], which may slow down and hindering the learning process in many tasks such as classification, reduce the learning performance, and/or learn complex models. A pre-processing step is often needed to remove some of the irrelevant or even noisy data, which can be achieved by *feature selection* (FS) for selecting only a small subset of informative features, *instance selection* (IS) for selecting only a small subset of representative examples/instances, or *FS and IS* for removing useless or redundant features and instances [13,17]. However, FS and/or IS is a

R. Booth and M.-L. Zhang (Eds.): PRICAI 2016, LNAI 9810, pp. 588–602, 2016.
DOI: 10.1007/978-3-319-42911-3_49

challenging problem due to two main reasons. The first is the large search space, which grows exponentially with the total number of features and instances. The second is that there are almost always interactions between features, which leads to a complex search space with many local optima and a good fitness function is often needed to guide the search in order to find a good solution. There have been a large number of works on FS, but not much work on IS, or FS and IS [13].

Different search techniques have been used for FS, but existing algorithms still suffer from the problem of stagnation in local optima. Evolutionary computation techniques are capable of searching large dimensions for solutions. Previous work has shown that various evolutionary computation techniques, such as differential evolution (DE) [15], particle swarm optimization [1,11,19], genetic algorithms [18,26] and others [2,8], achieve better performances than traditional FS and IS approaches [20]. This research will be utilizing the DE approach. DE is a simple but effective approach, which has been used for solving a wide range of complex problems, especially the ones with a large search space [24]. Recent works [3,21] also show its capabilities in solving FS problems, but its potential on IS has not been fully investigated.

Based on the evaluation criteria or fitness functions, feature and/or IS approaches can be grouped into wrapper approaches and filter approaches [23], where wrappers involves training a learning/classification algorithm in each evaluation to use the accuracy to show how good the candidate solution is, and filters are independent from any learning/classification algorithm. Due to the direct link between the learning algorithm and the candidate solution, wrappers can often achieve better accuracy than filters, but are computationally expensive. Filters are often very fast, but may not achieve as high accuracy as wrappers [7].

DE has only been used for wrapper FS recently [4,5,12,21]. Compared with the popularity and promising performance achieved by DE in other areas [6], the potential of DE has not been fully investigated. Although most machine learning tasks require FS and/or IS, classification is the area with the most applications, which could be a good starting point for the investigation.

Goals. The aim of this research is to investigate the use of DE for data preprocessing, which includes FS only, IS only, and FS and IS together. The proposed methods are expected to reduce the size of the data and increase or at least without significantly reducing the classification accuracy. More specifically, the overall goal is broken down into the following objectives:

1. develop a new DE based FS algorithm for selecting a subset of features to reduce the dimensionality and maintain or even increase the classification performance,
2. develop a new DE based IS algorithm for selecting a small subset of representative instances to reduce the size of the data without significantly reducing the classification accuracy,
3. develop a new DE based FS and IS algorithm to achieve FS and IS simultaneously, and

4. investigate the performance increase of the new algorithms compared with existing techniques.

2 Proposed Algorithms

In this section, we will investigate the use of DE for FS, IS, as well as FS and IS. Since feature and/or IS are binary tasks, i.e. either select or not, but DE was originally proposed as a continuous search technique, a binary DE algorithm will be needed. Different from most existing approaches using classification accuracy to evaluate the fitness (i.e. wrapper approaches), we will develop a series of filter algorithms based on the interclass and intraclass (IIC) measures to evaluate each candidate solutions.

2.1 Binary Differential Evolution

In DE candidate solutions are represented by vectors, with various operators being performed on them at each generation. The operators can range from mathematical functions such as addition, subtraction, or multiplication, to genetic operators such as crossover and mutation. There are various versions of DE in the literature [16]. One of the most promising one is DE/best/1, which is used in this work. A DE/best/1 iteration is defined as such

$$\boldsymbol{v}_{i,G+1} = \boldsymbol{x}_{best,G} + F.(\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}) \tag{1}$$

where i indicates i-th solution in the population, G and $G+1$ indicates the current and next generations. F is a scale factor controlling the size of the particle's movement, and $\boldsymbol{x}_{r_1,G}$ and $\boldsymbol{x}_{r_2,G}$ are other random candidate solutions chosen from the population such that $\boldsymbol{x}_{best,G} \neq \boldsymbol{x}_{r_1,G} \neq \boldsymbol{x}_{r_2,G}$. $\boldsymbol{x}_{best,G}$ is the current global best solution, which is a main feature of DE/best/1 that separates it from other implementations such as DE/rand/1. As seen in the equations, the current global best solution is a main factor, or the bases, of all new solutions. In DE/rand/1, three solutions are chosen at random to generate the new solution.

An initial population of candidate solutions are randomly generated. In each generation, a tentative new candidate solution, $\boldsymbol{v}_{i,G+1}$, is generated with the equation above for each solution i of the population. $\boldsymbol{x}_{i,G+1}$ is updated to $\boldsymbol{v}_{i,G}$ at the $G+1$ generation if the fitness $\boldsymbol{v}_{i,G+1}$ is better than $\boldsymbol{x}_{i,G}$, i.e. an improvement on the old solution. Otherwise $\boldsymbol{x}_{i,G+1}$ is the same as $\boldsymbol{x}_{i,G}$.

Due to similarities between DE and PSO, previous work on binary PSO [10] can be used here. A conversion must be made from the continuous vector representation of the candidate solutions to the binary solutions required for the selection problems. The conversion is given by:

$$output_{i,d} = \begin{cases} 1, \text{ if } rand() < \frac{1}{1+e^{-x_{i,d}}} \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

where $output$ is the d-th bit of the i-th solution, $rand()$ is a random number between 0 and 1, and $\boldsymbol{x}_{i,d}$ is the d-th value of the i-th vector of the candidate

Algorithm 1. Interclass Distance

```
for all classes do
    find all instances belong to this class
    construct mean instance from all instances belong to this class
    find and store a representative set of instances (ReSs) of this class
end for
distance := 0
for all ReSs do
    classDistance := 0
    links := 0
    for all other ReSs do
        for all Instance i1 in ReS do
            for all Instance i2 in other ReS do
                classDistance += distanceBetween(i1,i2)
                links += 1
            end for
        end for
    end for
    distance += classDistance / links
end for
interclass distance := distance / number of classes
```

solution, normalized by the sigmoid function. The values in *output* determine the selection of features or instances.

2.2 Fitness Function

The fitness function is one of the key components in the proposed algorithms, which is based on IIC measures. The IIC measures can be broken down to two parts, which are the interclass distance and the intraclass spread.

Interclass Distance. The interclass distance is a measure of the separability of classes in a dataset. The larger the distance, the further apart and separated the classes are. Therefore, a big distance means that the classes are more distinguished and there is less overlap between classes, which is expected to have a better classification performance. To achieve the goal of performing classification, we propose to build a prototype, which is a mean instance, or centroid, for each class based on a set of representative instances. The reasons for not using all instances here are to avoid outliers and long computational cost.

Algorithm 1 shows how the interclass distance is calculated. For each class, a mean instance, or centroid, is constructed. The mean instance is a feature vector that each value is the mean of all instances belonging to that class. A representative set of instances are the ones nearest to the constructed mean instance, which is found to represent the class. Each representative set has a size of 10 % of the total number of instances belonging to that class plus one, i.e.

Algorithm 2. Intraclass Spread

```
1: {Part 1: Calculating spread of each feature}
2: for all features do
3:    featureSpread := 0
4:    for all class do
5:       featureSpread += σ_{f,c}
6:    end for
7: end for
8: {Part 2: Calculating overall spread}
9: intraclass spread := 0
10: for all featureSpread do
11:    intraclass spread += featureSpread
12: end for
```

the mean instance. The representative sets are used to calculate the Euclidean distance between classes, which can be seen from Lines 1–5 in Algorithm 1. Then, the Euclidean distance between the representative sets are found. The average distance between two classes is defined as the average distance of each instance in one representative set to each instance in the other representative set. The average is taken here due to the different numbers of instances belonging to each class. In Lines 10–17, the average distance between two classes is calculated for between every class. Then the average of the averages between every two classes is used as the distance between all classes as shown in Line 20.

The following equations provides a mathematical form of this calculation.

$$Distance = \frac{\sum_{C_a, C_b \in C}^{a \neq b} \frac{\sum_{i \in C_a} \sum_{j \in C_b} |i-j|}{|Ca| \times |Cb|}}{|C|}, \tag{3}$$

where C is the set of all representative sets, C_a, C_b are any two different representative sets, and i, j are individual instances.

Intraclass Spread. The intraclass spread is a measure of how spread out a particular class is. The further spread a class is, the more likely it is to overlap with other classes, providing a more cohesive representation of the class. Therefore a smaller spread is preferred. The spread of a class is given by the spread of its features, particularly by all the feature values of the instances in each class. This allows easier calculation, but does not change the total spread of a given dataset due to the associative properties of addition.

The spread of a particular feature is given by the sum of the standard deviation of each class's set of values for that feature. The spread of the set of features is given by the sum of each feature's spread.

$$Spread = \sum_{f \in F'} \sigma_f, \ \ where \ \ \sigma_f = \sum_{c \in C} \sigma_{f,c} \tag{4}$$

where F' is a set of features, c is an instance belonging to the class C, and $\sigma_{f,c}$ is the standard deviation of the values representing feature f in class c.

Fitness Function. To achieve good classification performance, ideally, the intraclass spread should be minimized and the interclass distance should be maximized. Therefore, a (minimization) fitness function is formed and shown by Eq. 5.

$$Fitness = \frac{Spread + \alpha.|F|}{Distance} \tag{5}$$

where $|F|$ is the number of features, and α is a coefficient. The constant $\alpha.|F|$ is added to the spread in the numerator to control the weight ratio between the spread and the distance. A smaller constant would give more weight to the spread, and a larger constant gives more weight to the distance. This also means that the number of features selected in FS can be controlled, as the number of features directly affect the spread and distance, i.e. intraclass spread wants fewer selected features, whereas interclass distance wants more features. Therefore, by adjusting the weights of spread and distance, the number of features can be adjusted.

2.3 New Algorithms

We will investigate the use of DE for FS, IS, as well as FS and IS together. Since the fitness function, Eq. 5, eventually shows how well different classes can be separated, it is used in all the three algorithms, to form IIC-FS, IIC-IS, and IIC-FIS, for FS, IS, and FS and IS, respectively.

The goal of the three algorithms are the same, i.e. minimizing the fitness value. They all follow the basic DE process. The key difference between them is the representation since the candidate solutions are different, i.e. a subset of features, a subset of instances, and a subset of instances with selected features only for IIC-FS, IIC-IS, and IIC-FIS, respectively. In IIC-FS, the representation of each individual in DE is a m-dimensional boolean vector for a dataset with m features, where each dimension determines whether the corresponding feature is selected. 1 means the feature is selected and 0 otherwise. In IIC-IS, the representation is a n-dimensional boolean vector for a dataset with n instances, where each dimension determines whether the corresponding instance is selected. In IIC-FIS, the representation is a $(n+m)$-dimensional boolean vector, where each dimension determines whether the corresponding feature or instance is selected.

In IIC-FS, as the instances do not change, each feature has a particular intraclass spread value associated with it that also does not change. These values only need to be calculated once. Training times are improved since each feature's spread is stored in memory and is simply read for each fitness evaluation, as opposed to recalculating each value every time it is needed. Therefore the first part of Algorithm 2 is only performed once at the beginning. Further evaluations only need to perform the second part. The same cannot be achieved for interclass distances, as changing the dimensions (features) of instances also changes their relative distances. Therefore Algorithm 1 is performed in full for every fitness evaluation for FS. In IIC-IS and IIC-FIS, due to the changing instances, and

thus both the spread and distances of the data, both algorithms' calculations are performed in full for every fitness evaluation.

In addition, since DE has never been used for IS, and FS and IS, we investigate two wrapper based methods using KNN as the classification algorithm to evaluate the classification performance as the fitness function for IS only (KNN-FS), and for FS and IS (KNN-FIS). Both KNN-FS and KNN-FIS are also new to some extent.

3 Experiment Design

The proposed algorithms are run against 10 datasets taken from the UCI machine learning repository shown in Table 1. These datasets are selected to represent a range of feature and instance counts, as well as being widely used datasets such that the new algorithms can be compared against existing ones. Data is normalized as they are loaded, ensuring that distance and standard deviation measures are on the same scale for all features.

Table 1. Experiment datasets

Dataset	NO. of features	NO. of instances	NO. of classes	α
Wine	13	178	3	0.4
Australian	14	690	2	2
Zoo	17	101	7	0.65
Vehicle	18	846	4	0.38
German	24	1000	2	0.16
Wbcd	30	569	2	0.27
Ionosphere	34	351	2	0.2
Lung	56	32	2	0.41
Sonar	60	208	2	0.2
Movementlibras	90	360	15	1.2

For each selection process, 30 runs are conducted for each dataset. The DE has a population of 80 candidate solutions, and is run for 100 generations. Since an optimal solution cannot be easily determined and classification rate is not part of the training process, there is no early stopping criteria. The data is resplit every 10 runs for a total of 3 different splits per dataset. The split is done randomly, with each instance having a 70 % chance of being used for training, and 30 % chance of being used for testing.

In ICC-FS, IIC-IS and IIC-FIS, a search was conducted before the experiments for α. The coefficient values for α in Table 1 were found to give a similar number of features to KNNFS and were used for the experiments. IIC-FIS has two specific implementations. The first one, marked with "200", is run with 200

candidate solutions of DE instead of 80. This is to accommodate for the larger search space due to the dimension size being the sum of number of features and instances. The second, marked with "ICC-Half", uses a modified KNN for classification after using IIC-FS to reduce the features. This modified KNN only uses half the instances. For each class, the centroid, or mean instance, is calculated from every instance of that class in the training set. Then half the instances of that class, the half closest to the centroid, are used in the KNN for classification. Although only the features are selected in the training process, this modified KNN selects instances, putting it under FS and IS. In KNN-FS, KNN-IS, and KNN-FIS, the average classification accuracy of a 10-fold validation on the training set is used as the fitness value, where 10-fold validation is used to make sure that no FS bias is involved and the test set is completely unseen for the FS methods.

After the DE generations, the solutions with the best fitness are evaluated for its classification accuracy on the test set, where KNN (K = 5) is used as the classifier. A non-parametric test, the Mann?-Whitney U test, is then used to compare the testing accuracy and number of features/instances selected by the IIC measure against using all features, as well as the standard KNN technique.

4 Results and Discussions

Tables 2, 3, and 4 show the results of the three sets of experiments. Table 2 shows the results of the FS using KNN-FS, and IIC-FS. The first two columns show the dataset name and the methods. The third column shows the average and standard deviation of the number of selected features. The fourth column shows the average, standard deviation, and best accuracy on the test sets. The column "Test 1" shows the statistical significance tests between the method in the corresponding row against All, where "○", "⋆", and "=" means the corresponding method is significantly better than, worse than, and similar to that of All, respectively. The column "Test 2" shows the same information against KNN-FS. Note that "better" means larger for accuracy, but means smaller for the number of features. The last column shows the average training time for a single run, where the number is shown in seconds. Table 3 shows the results of IS, and Table 4 shows the results of FS and IS together, where the meanings of symbols are the same as in Table 2.

4.1 Results of Feature Selection

According to Table 2, it can be seen that comparing IIC-FS with All, the number of features is reduced to around one third of the total number of features. With the reduced feature subsets, IIC-FS achieved better or at least similar classification accuracy than using all the original features on nine out of the ten datasets. The results show that proposed IIC-FS can be successfully used for FS to evolve a small number of features, which can maintain or even increase the classification performance.

Table 2. Experimental results for feature selection

Dataset	Method	NO. of Features	Accuracy		Test 1		Test 2		Average
			Mean (Std)	Best	Acc	Size	Acc	Size	Time
Wine	All	13	0.948 (0.03)	0.979					
	KNN-FS	6.4 (1.13)	0.936 (0.05)	0.98	=	○			566.57
	IIC-FS	5.3 (0.47)	0.959 (0.02)	0.981	=	○	=	○	5.47
Aus.	All	14	0.859 (0.01)	0.867					
	KNN-FS	5.47 (0.94)	0.867 (0.02)	0.903	=	○			8652.8
	IIC-FS	2.9 (0.66)	0.858 (0.01)	0.862	=	○	=	○	95.67
Zoo	All	17	0.909 (0.05)	0.946					
	KNN-FS	9 (1.58)	0.938 (0.04)	1	=	○			264.7
	IIC-FS	8.33 (1.42)	0.904 (0.03)	0.968	=	○	⋆	=	2.27
Vehicle	All	18	0.667 -0	0.667					
	KNN-FS	8.6 (1.54)	0.693 (0.03)	0.751	○	○			20937.43
	IIC-FS	8.63 (1.22)	0.645 (0.04)	0.719	⋆	○	⋆	=	125.27
German	All	24	0.697 (0.01)	0.709					
	KNN-FS	10.4 (1.92)	0.715 (0.03)	0.766	○	○			43018.27
	IIC-FS	8.67 (1.99)	0.718 (0.02)	0.759	○	○	=	○	413.1
WBCD	All	30	0.959 (0.01)	0.969					
	KNN-FS	13.2 (2.16)	0.956 (0.02)	0.982	=	○			18815.8
	IIC-FS	14.13 (1.93)	0.952 (0.01)	0.982	=	○	=	=	189
Ionos.	All	34	0.839 (0.01)	0.843					
	KNN-FS	9.8 (2.50)	0.876 (0.03)	0.933	○	○			8652.6
	IIC-FS	10.57 (2.03)	0.852 (0.02)	0.899	○	○	⋆	=	87.17
Lung	All	56	0.747 (0.04)	0.8					
	KNN-FS	23 (4.34)	0.707 (0.09)	0.923	=	○			129.53
	IIC-FS	24.27 (2.88)	0.719 (0.08)	0.846	=	○	=	=	2.7
Sonar	All	60	0.809 (0.06)	0.895					
	KNN-FS	26.07 (2.80)	0.792 (0.05)	0.895	=	○			9331.53
	IIC-FS	26.33 (3.46)	0.798 (0.06)	0.912	=	○	=	=	148
Movement libras	All	90	0.707 (0.03)	0.745					
	KNN-FS	38.9 (6.15)	0.699 (0.04)	0.764	=	○			51255.67
	IIC-FS	39.37 (4.23)	0.682 (0.05)	0.764	=	○	=	=	154.37

Comparing IIC-FS with KNN-FS, the number of features and the classification performance are similar in most of the cases, with three cases of IIC-FS selecting a smaller feature subsets and KNN-FS achieving better classification accuracy. KNN-FS is expected to achieve better accuracy since it is a wrapper approach while IIC-FS is a filter approach.

In terms of the training time, there is a huge difference between IIC-FS with KNN-FS, where IIC-FS always used a substantial shorter time (48 to 167 times faster) than KNN-FS, with the Vehicle dataset having the biggest difference.

In summary, the proposed ICC-FS methods can be successfully used for FS. As a filter approach, IIC-FS is able to achieve similar FS performance to the wrapper method, KNN-FS, but the computational time is much shorter.

Table 3. Experimental results for instance selection

Dataset	Method	NO. of instances	Accuracy		Test 1		Test 2		Average
			Mean(Std)	Best	Acc	Size	Acc	Size	Time
Wine	All	128 (3.61)	0.948 (0.031)	0.979					
	KNN-IS	50.9 (5.82)	0.943 (0.031)	1	=	○			255.33
	IIC-IS	47.57 (6.02)	0.943 (0.02)	0.98	=	○	=	=	3.2
Australian	All	487.67 (10.97)	0.859 (0.009)	0.867					
	KNN-IS	209.47 (20)	0.862 (0.015)	0.888	=	○			4327.37
	IIC-IS	209.9 (20.52)	0.862 (0.021)	0.898	=	○	=	=	40.17
Zoo	All	69.67 (5.51)	0.909 (0.046)	0.946					
	KNN-IS	31.83 (6.79)	0.856 (0.043)	0.968	⋆	○			120.2
	IIC-IS	26.13 (4.03)	0.817 (0.079)	0.968	⋆	○	=	○	0.03
Vehicle	All	596 (10.54)	0.667 (0)	0.667					
	KNN-IS	277 (25.63)	0.62 (0.032)	0.699	⋆	○			11258.87
	IIC-IS	255.13 (22.4)	0.629 (0.037)	0.707	⋆	○	=	○	35.07
German	All	704 (14.53)	0.697 (0.009)	0.709					
	KNN-IS	311.77 (25.42)	0.711 (0.019)	0.756	○	○			26422.23
	IIC-IS	300.47 (25.19)	0.7 (0.022)	0.745	=	○	=	=	275.43
WBCD	All	400 (10.39)	0.959 (0.011)	0.969					
	KNN-IS	185.2 (20.76)	0.957 (0.013)	0.975	=	○			11477.57
	IIC-IS	168.67 (14.59)	0.946 (0.013)	0.969	⋆	○	⋆	○	149.47
Ionosphere	All	241.33 (8.62)	0.839 (0.005)	0.843					
	KNN-IS	104.5 (13.01)	0.86 (0.017)	0.892	○	○			5608.13
	IIC-IS	99.77 (8.6)	0.707 (0.057)	0.866	⋆	○	⋆	=	59.07
Lung	All	21.67 (2.52)	0.747 (0.045)	0.8					
	KNN-IS	6.77 (2.39)	0.647 (0.038)	0.7	⋆	○			32.77
	IIC-IS	2.5 (0.9)	0.693 (0.124)	0.9	=	○	○	○	1.63
Sonar	All	147.67 (3.51)	0.809 (0.063)	0.895					
	KNN-IS	63.4 (9.05)	0.678 (0.054)	0.817	⋆	○			4899.07
	IIC-IS	57.23 (6.88)	0.668 (0.058)	0.767	⋆	○	=	○	68.33
Movement libras	All	247 (9.85)	0.707 (0.033)	0.745					
	KNN-IS	122.7 (9.62)	0.511 (0.057)	0.6	⋆	○			36993.93
	IIC-IS	87.37 (8.68)	0.42 (0.045)	0.482	⋆	○	⋆	○	143.6

4.2 Results of Instance Selection

Table 3 shows the results of IS, where both KNN-IS and IIC-FS are newly investigated in this paper. The results show that both KNN-IS and IIC-IS selected only a much smaller number of instances compared with the total number of instances on all the datasets. Although the number of instances to be selected by IIC-IS was not controlled, the number of instances selected by IIC-FS is significantly smaller than that of KNN-IS on six out of the ten datasets, and similar on the other four datasets. Compared to using all instances, both KNN-IS and IIC-IS performed significantly better or similar in around half of the cases, but in general the difference is not too big, and the best accuracy of KNN-IS and IIC-IS is often better than using all instances. This is different from the good performance of their corresponding FS methods, as shown in Table 2. This is not too surprised given that IS could change the original pattern and distribution

of the data, which is probably why there has been much more work on FS than IS, although IS can benefit classification in many ways as FS. We will further investigate effective IS methods in the future.

Regarding the training times, both KNN-IS and IIC-IS have a faster training time than their respective FS counterparts as shown in Table 2. This is due to the highly reduced number of instances in each fitness evaluation, resulting in fewer calculations of the distances between instances. Once again the IIC technique is much faster than the KNN technique. The speed increase ranges from 20–321 times faster. The Zoo dataset is a special case, most of the training times where recorded as 0 (seconds) since the entire training process took less than one second. This results in an extremely low average training time, which was 4000 times faster than KNN-IS.

In summary, the two IS methods cannot in most cases maintain or increase the classification performance, although it can substantially reduce the size of the data. The speed of the algorithms is very fast, much faster than the FS methods. How to maintain the speed and simultaneously increase the classification performance is an interesting direction for future work.

4.3 Results of Feature and Instance Selection

Table 4 shows the results for the FS and IS experiments, where "200" is used to represent the implementation of IIC-FIS with 200 generations, and "IIC-Half" is used to represent the version of IIC-FIS with the KNN implementation. All the three methods on this set of experiments are new in this work.

The results from Table 4 show that both the number of features and the number of instances have been significantly reduced, but the price is the lower classification performance, especially on the large datasets. IIC-200 selected significantly more features than KNN-FIS on every dataset, and they are similar in the number of instances on most datasets. IIC-Half has a similar number of features selected on seven of the ten datasets as KNN-FIS, and a similar classification performance. Compared to using all feature and instances, KNN-FIS achieves a better classification accuracy on three datasets, and worse on six. Both IIC-200 and IIC-Half achieves similar results on three (Wine and German for both, then Australian for 200 and Zoo for IIC-Half) datasets. Neither achieves a significantly better result than using all features and instances.

For the training time, the IIC methods have a much faster time than the KNN based method. Although the improvement here is not as high as in FS and IS, with the range of reduction at 4–80 times faster.

4.4 Analysis on the Computational Time

The ICC methods are orders of magnitude faster than KNN-FIS in terms of the training time while still achieving similar results. According to Tables 2, 3 and 4, the average training speed is roughly 4–400 times faster (on average 120 times faster) using IIC than the KNN technique. This is due to the number of distance

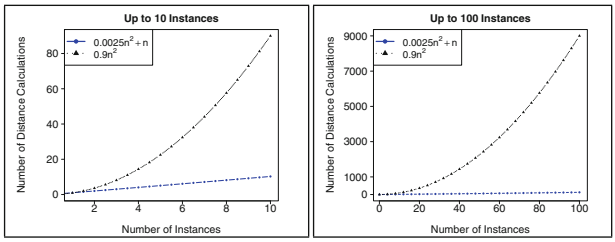
Table 4. Experimental results for feature and instance selection

Dataset	Method	Features used	Instances used	Accuracy	Test 1			Test 2			Average
				mean(Std)	Acc	Ins	Feas	Acc	Ins	Feas	Time
Wine	All	13	128(3.61)	0.948(0.03)							
	KNN-FIS	7.17(2.13)	52(5.62)	0.935(0.04)	=	○	○				84.9
	IIC-200	10.93(1.17)	54.93(7.98)	0.94(0.04)	=	○	○	=	⋆	=	5
	IIC-Half	6.97(1.13)	66.67(1.27)	0.941(0.03)	=	○	○	=	=	⋆	4.07
Aus.	All	14	487.67(10.97)	0.859(0.01)							
	KNN-FIS	7.03(1.77)	207.57(21.50)	0.866(0.02)	○	○	○				1563.4
	IIC-200	11.1(1.18)	223.17(23.03)	0.858(0.02)	=	○	○	=	⋆	⋆	63.57
	IIC-Half	4.7(1.29)	245(4.32)	0.813(0.03)	⋆	○	○	⋆	○	⋆	86.23
Zoo	All	17	69.67(5.51)	0.909(0.05)							
	KNN-FIS	8.43(1.85)	31.17(6.18)	0.835(0.04)	⋆	○	○				38.1
	IIC-200	12.57(1.74)	30.77(4.16)	0.836(0.08)	⋆	○	○	=	⋆	=	1.13
	IIC-Half	8(1.26)	40.67(2.54)	0.916(0.04)	=	○	○	○	=	⋆	1.33
Veh.	All	18	596(10.54)	0.667 -0							
	KNN-FIS	9.5(1.85)	283.03(19.67)	0.652(0.04)	⋆	○	○				4082
	IIC-200	15(1.84)	287.9(24.69)	0.627(0.03)	⋆	○	○	⋆	⋆	=	78.2
	IIC-Half	9.57(1.14)	301.33(4.18)	0.572(0.03)	⋆	○	○	⋆	=	⋆	104.17
Germ.	All	24	704(14.53)	0.697(0.01)							
	KNN-FIS	9.97(2.24)	316.5(28.43)	0.712(0.03)	○	○	○				7683.53
	IIC-200	15.93(2.08)	326.63(21.46)	0.696(0.02)	=	○	○	⋆	⋆	=	273.7
	IIC-Half	10.47(2.21)	353.67(6.23)	0.687(0.04)	=	○	○	⋆	=	⋆	397.47
WBCD	All	30	400(10.39)	0.959(0.01)							
	KNN-FIS	14.23(2.99)	186.13(17.47)	0.947(0.02)	⋆	○	○				4215.1
	IIC-200	21.13(2.87)	185.33(16.04)	0.95(0.01)	⋆	○	○	=	⋆	=	178.67
	IIC-Half	14.83(2.52)	201.33(4.57)	0.941(0.01)	⋆	○	○	=	=	⋆	215.87
Ionos.	All	34	241.33(8.62)	0.839(0.01)							
	KNN-FIS	14.53(2.96)	106.97(12.08)	0.864(0.03)	○	○	○				1633.83
	IIC-200	19.63(2.24)	105.33(11.30)	0.763(0.04)	⋆	○	○	⋆	⋆	=	61.27
	IIC-Half	11.93(2.42)	122.33(3.36)	0.708(0.04)	⋆	○	○	⋆	○	⋆	116.47
Lung	All	56	21.67(2.52)	0.747(0.04)							
	KNN-FIS	23.87(3.66)	6.23(2.06)	0.647(0.04)	⋆	○	○				10.37
	IIC-200	27.73(4.70)	2.43(1.07)	0.704(0.12)	⋆	○	○	○	⋆	○	1.87
	IIC-Half	24.6(3.63)	12.67(1.27)	0.715(0.09)	⋆	○	○	○	=	⋆	2.27
Sonar	All	60	147.67(3.51)	0.809(0.06)							
	KNN-FIS	26.57(4.32)	64.07(9.66)	0.722(0.05)	⋆	○	○				1648
	IIC-200	30.53(4.07)	61.43(6.37)	0.679(0.05)	⋆	○	○	⋆	⋆	=	62.03
	IIC-Half	26.3(4.46)	75.67(1.27)	0.707(0.04)	⋆	○	○	=	=	⋆	86
Move. libras	All	90	247(9.85)	0.707(0.03)							
	KNN-FIS	43.77(5.77)	125.67(9.83)	0.497(0.06)	⋆	○	○				11808.07
	IIC-200	54.17(5.11)	108.67(9.77)	0.46(0.05)	⋆	○	○	⋆	⋆	○	190.83
	IIC-Half	40.3(4.02)	135.33(4.18)	0.553(0.04)	⋆	○	○	○	○	⋆	142.63

calculations between instances, a costly operation, is much lower in IIC than KNN.

Assuming instances are equally distributed between classes, for each fitness evaluation the number of calculations between values in IIC can be roughly calculated by the following equation:

$$n + 0.01 \left(\frac{n}{c}\right)^2 . \frac{c(c-1)}{2}, \tag{6}$$



(a) Up to 10 instances (b) Up to 100 instances

Fig. 1. Number of distance calculations in KNN and IIC

where $c \in [1, n]$ is the number of classes and n is the number of instances. The initial n at the beginning is to find the distance from each instance to its mean instance to identify the representative set. The second part is the distance calculations between representative set. A fully connected graph (where each vertex is a class representative set) has $\frac{c(c-1)}{2}$ edges, with each edge consisting of $0.1\frac{n}{c} \times 0.1\frac{n}{c}$ calculations. This is largest at $c = 2$ for all $n > 2$, giving $n+0.0025n^2$ calculations.

For the KNN based techniques, each evaluation requires $10 \times 0.9n \times 0.1n$ $(0.9n^2)$ distance calculations. This is from the 10-fold cross validation, where in each fold 10 % of instances are compared with the other 90 %. So the rough number of calculations is

$$n + 0.0025n^2 > 0.9n^2 \text{ for all } n > 1, \tag{7}$$

Figure 1 shows the number of distance calculations in KNN and IIC when the number of instances is 10 and 100. One can see that even at only 10 instances, KNN has greatly separated from IIC. When there is 100 instances, IIC is negligible compared to KNN. Note that are more than 100 instances in the training set in all but two datasets.

5 Conclusions and Future Work

The goal of this paper was to investigate the use of DE for feature and/or IS in classification, which a new binary DE algorithm and a new fitness function. The experiments and comparisons on ten datasets show that the proposed DE based FS algorithm is successful in terms of the number of features, the classification accuracy and the training time. However, when the IS task involved, the algorithms are good at reducing the size of the data, but the classification accuracy may suffer, which is a critical problem. The reason for this is due to the large search space, which is also probably why there has been much more work on FS than IS.

This paper investigates a series of different feature and/or IS methods, which have not been done before. Although it is only a preliminary work, the findings

are very useful, especially when both feature selection and instance selection are becoming increasingly important for big data tasks. There is still a lot of work should be done in this filed. For example, a novel representation of solutions is needed, which can effectively reduce the search space and also form a more smooth landscape to be more easily searched. A computationally cheap fitness measure is also of key component, especially on datasets with a large number of features and instances. We will focus on these directions in the future.

References

1. Ahmad, S.S.S.: Feature and instances selection for nearest neighbor classification via cooperative PSO. In: 2014 Fourth World Congress on Information and Communication Technologies (WICT), pp. 45–50. IEEE (2014)
2. Ahmed, S., Zhang, M., Peng, L., Xue, B.: Multiple feature construction for effective biomarker identification and classification using genetic programming. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 249–256. ACM (2014)
3. Al-Ani, A., Alsukker, A., Khushaba, R.N.: Feature subset selection using differential evolution and a wheel based search strategy. Swarm Evol. Comput. **9**, 15–26 (2013)
4. Bharathi, P.T., Subashini, P.: Differential evolution and genetic algorithm based feature subset selection for recognition of river ice type. J. Theor. Appl. Inf. Technology **7**(1), 254–262 (2014)
5. Bharathi, P.T., Subashini, P.: Optimal feature subset selection using differential evolution and extreme learning machine. Int. J. Sci. Res. (IJSR) **3**, 1898–1905 (2014)
6. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
7. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
8. Hancer, E., Xue, B., Karaboga, D., Zhang, M.: A binary ABC algorithm based on advanced similarity scheme for feature selection. Appl. Soft Comput. **36**, 334–348 (2015)
9. John, G.H., Kohavi, R., Pfleger, K., et al.: Irrelevant features and the subset selection problem. In: Machine Learning: Proceedings of the Eleventh International Conference, pp. 121–129 (1994)
10. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108 (1997)
11. Lane, M.C., Xue, B., Liu, I., Zhang, M.: Gaussian based particle swarm optimisation and statistical clustering for feature selection. In: Blum, C., Ochoa, G. (eds.) EvoCOP 2014. LNCS, vol. 8600, pp. 133–144. Springer, Heidelberg (2014)
12. Li, Z., Shang, Z., Qu, B., Liang, J.: Feature selection based on manifold-learning with dynamic constraint handling differential evolution. In: IEEE Congress on Evolutionary Computation (CEC), pp. 332–337 (2014)
13. Liu, H., Motoda, H.: Instance Selection and Construction for Data Mining, vol. 608. Springer Science & Business Media, US (2013)
14. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. Knowl. Data Eng. **17**(4), 491–502 (2005)

15. Qin, A., Huang, V., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Knowl. Data Eng. **13**(2), 398–417 (2009)
16. Storn, R.: On the usage of differential evolution for function optimization. In: 1996 Biennial Conference of the North American Fuzzy Information Processing Society, pp. 519–523. IEEE (1996)
17. Tsai, C.F., Chen, Z.Y.: Towards high dimensional instance selection: an evolutionary approach. Decision Support Syst. **61**, 79–92 (2014)
18. Tsai, C.F., Eberle, W., Chu, C.Y.: Genetic algorithms in feature and instance selection. Knowl. Based Syst. **39**, 240–247 (2013)
19. Unler, A., Murat, A.: A discrete particle swarm optimization method for feature selection in binary classification problems. Eur. J. Oper. Res. **206**(3), 528–539 (2010)
20. Xue, B., Zhang, M., Browne, W., Yao, X.: A survey on evolutionary computation approaches to feature selection. IEEE Trans. Evol. Comput. PP(99) (2015). doi:10.1109/TEVC.2015.2504420
21. Xue, B., Fu, W., Zhang, M.: Multi-objective feature selection in classification: a differential evolution approach. In: Dick, G., Browne, W.N., Whigham, P., Zhang, M., Bui, L.T., Ishibuchi, H., Jin, Y., Li, X., Shi, Y., Singh, P., Tan, K.C., Tang, K. (eds.) SEAL 2014. LNCS, vol. 8886, pp. 516–528. Springer, Heidelberg (2014)
22. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans. Cybern. **43**(6), 1656–1671 (2013)
23. Xue, B., Zhang, M., Browne, W.N.: A comprehensive comparison on evolutionary feature selection approaches to classification. Int. J. Comput. Intell. Appl. **14**(02), 1550008 (2015)
24. Yang, Z., Tang, K., Yao, X.: Scalability of generalized adaptive differential evolution for large-scale continuous optimization. Soft Comput. **15**, 2141–2155 (2011)
25. Zhu, P., Zuo, W., Zhang, L., Hu, Q., Shiu, S.C.: Unsupervised feature selection by regularized self-representation. Pattern Recogn. **48**(2), 438–446 (2015)
26. Zhu, Z., Ong, Y.S., Dash, M.: Wrapper-filter feature selection algorithm using a memetic framework. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **37**(1), 70–76 (2007)