Noname manuscript No.
(will be inserted by the editor)

Mutual Information for Feature Selection: Estimation or Counting?

Hoai Bach Nguyen · Bing Xue · Peter Andreae

the date of receipt and acceptance should be inserted later

Abstract In classification, feature selection is an important pre-processing step to simplify the dataset and improve the data representation quality, which makes classifiers become better, easier to train, and understand. Because of an ability to analyse non-linear interactions between features, mutual information has been widely applied to feature selection. Along with counting approaches, a traditional way to calculate mutual information, many mutual information estimations have been proposed to allow mutual information to directly work in continuous datasets. This work focuses on comparing the effect of counting approach and kernel density estimation (KDE) approach in feature selection using particle swarm optimisation as a search mechanism. The experimental results on 15 different datasets show that KDE can work well on both continuous and discrete datasets. In addition, the feature subsets evolved by KDE achieves similar or better classification performance than the counting approach. Furthermore, the results on artificial datasets with various interactions show that KDE is able to correctly capture the interaction between features, in both relevance and redundancy, which can not be achieved by using the counting approach.

Keywords Mutual information · Feature selection · Classification · Particle Swarm Optimisation

1 Introduction

Nowadays, under the development of technology, many real-world problems have a large number of features, which causes difficulties to machine learning tasks, such as classification. Particularly, there might be some noisy or irrelevant features, which do not provide any useful information to the class label and may also deteriorate the classification accuracy. In addition, some redundant features provide exactly the same information as other features, which results in a longer training time without any improvement in the classification performance. In such cases, feature selection is necessary to reduce any imprecise, misleading and redundant information.

Feature selection is typically a pre-processing step, which selects a feature subset from the original feature set. The selected features are expected to maintain or increase the useful information about the class label over using all features. Additionally, feature selection also aims to reduce the feature set size by removing redundant and irrelevant features. A smaller number of features is useful to avoid "the curse of dimensionality", which leads to improvements in both quality and training speed of classification.

However, feature selection is not an easy task due to its large search space. Suppose that there are n original features, then the total number of possible feature subsets is 2^n. So the search space's size exponentially grows with respect to the number of features. An exhaustive search approach, which considers all possible feature subsets, guarantees to select an optimal feature subset. However, it is too slow to perform in most cases. To improve the efficiency, some greedy feature selection approaches are proposed, for instance sequential forward selection (Whitney, 1971) and sequential backward selection (Marill and Green, 1963). However, these sequential searches usually get stuck at local optima due to the complicated search space of feature selection. Evolutionary computation (EC) algorithms, such as genetic algorithms (GAs), genetic programming (GP) or ant colony optimisation (ACO), particle swarm optimisation (PSO), have been well-known because of their global search ability,

Hoai Bach Nguyen · Bing Xue · Peter Andreae
School of Engineering and Computer Science
E-mail: {Hoai.Bach.Nguyen,Bing.Xue,Peter.Andreae}@ecs.vuw.ac.nz
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

which are suitable mechanisms to cope with large search problems like feature selection. Therefore, recently EC has been widely applied to feature selection,which can be seen in a comprehensive survey about EC-based feature selection algorithms done by Xue et al (2015). Among EC techniques, PSO is preferred because it has a natural representation for feature selection, in which each position entry corresponds to an original feature. In addition, PSO also has fewer parameters and converges more quickly than other EC algorithms. (Eberhart and Shi, 1998) showed that PSO is faster than GAs to achieve the same performance.

Beside the huge search space, feature selection is a challenging task because of the complicated interactions between features. On the one hand, two or more weakly relevant features might become significantly useful when working with each other, which is known as "complementary features". On the other hand, two relevant features might become redundant when working with each other because they provide the same information. The feature interaction is hard to capture because there can be multi-way interactions. A good evaluation criterion of feature subsets needs to be able to handle this difficulty. According to the evaluation criterion, existing feature selection methods can be classified into three main categories: wrapper, filter and embedded approaches (Dash and Liu (1997), Kohavi and John (1997)). In a wrapper approach, a specific classification algorithm is used to evaluate the selected feature subset. In other words, the classification accuracy will reflect the goodness of a feature subset. Meanwhile, filter approaches, which are done in an independent way of learning algorithms, use statistic characteristics of the data to evaluate the feature subset. Therefore, wrapper approaches usually achieve better classification accuracy than filter approaches. However, the filter approach has better generality, which means that its selected features can be applied to different classification algorithms rather than only a wrapped classification algorithm like wrappers. Additionally, filters usually have less expensive computation cost than wrappers because they do not involve any classification process. In embedded approach, the feature subset is selected during the training period for a classification algorithm. An example of the embedded approach is the decision tree classification algorithm, in which all features used in the trained tree are considered an important feature subset.

Filter approaches have been investigated by many researchers, who have proposed a large number of filter measures, such as Fisher score (Duda et al, 2012), Consistency measure (Dash et al, 2000), Correlation measure (Hall, 2000) and Mutual information (Kononenko, 1995). Among these filter measures, mutual information gains more attention because it is able to detect non-linear correlation between features. Further more, mutual information is capable to analyse the interaction between multiple features while other filter measures, like correlation measure, are limited to two-way interaction between features or between a single feature and the class label. However, currently most of MI-based feature selection algorithms count the number of instances in a dataset to derive probability distributions and mutual information. This counting approach can result in an inaccurate mutual information when there are not enough instances. In addition, the counting approach is applicable to only discrete datasets. To overcome these limitations, several estimation methods have been proposed to estimate mutual information (Walters-Williams and Li, 2009). Recently, we (Nguyen et al, 2016) proposed the first work, in which mutual information estimation was worked with PSO to achieve feature selection. The experimental results showed that mutual information estimation could guide PSO to evolve better feature subsets than the sequential search using the counting approach. However, due to the page limit, the comparisons between estimation approach and counting approach using the same search mechanism were not conducted and the feature interaction information was not deeply analysed .

Therefore, this work will provide a detail comparison between estimation and counting approach using PSO as a search technique.

1.1 Goals

The overall goal of this paper is to extend our work in (Nguyen et al, 2016) by inspecting mutual information estimation in more detail. Specifically, we will investigate:

- whether mutual information estimation for feature selection can work well on both discrete and continuous datasets. Note that in this paper, a discrete dataset means that its features are ordinal numeric features,
- whether mutual information estimation can achieve better performance than counting in terms of the classification performance and
- whether mutual information estimation can capture the interactions between features better than the counting approach.

2 Background

2.1 Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation (PSO) is proposed by Kennedy et al (1995), which is inspired from social behaviour such as bird flocking and fish schooling. In PSO, a problem is optimised by using a set of particles, called a swarm. Each particle is a candidate solution, which is represented by a position in the search space. The particle moves around the search space by using a velocity. Both particle's velocity and position are vectors of numbers, which have the same size as the number of dimensionality of the search space. The velocity of a particle is determined by its own best position, called *pbest*, and its neighbours best position, called *gbest*. Each velocity component is limited by a predefined maximum velocity, called v_{max}. The position and velocity of particle i, denoted by *x* and *v*, are updated according to the following equations:

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

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

where t denotes the t^{th} iteration in the search process, d is the d^{th} dimension in the search space, v_{id}^{t} and x_{id}^{t} represents the d^{th} entry of the i^{th}'s velocity and position respectively, w is an inertia weight, c_1 and c_2 are acceleration constants, r_{i1} and r_{i2} are random values uniformly distributed in [0,1], p_{id} and p_{gd} represent the position entry of $pbest$ and $gbest$ in the d^{th} dimension, respectively.

2.2 Information Theory

Entropy, one of the core concepts in information theory (Jaynes, 1957), is used to measure the uncertainty or the amount of information of a random variable. Given X is a discrete variable, its entropy can be calculated by the following formula:

$$H(X) = -\sum_{x\in X} P(X=x)*\log_2 P(X=x) \quad (3)$$

Entropy can be extended to measure the uncertainty of a joint variable, which consists more than one random variables. The joint entropy can be defined as:

$$H(X_1,\ldots,X_n) = -\sum_{\substack{x_i\in X_i\\ i=1\ldots n}} p(x_1,\ldots,x_n)*\log_2 p(x_1,\ldots,x_n) \quad (4)$$

where $p(x_1,\ldots,x_n) = P(X_1=x_1,\ldots,X_n=x_n)$

Mutual information is another important concept in information theory. Mutual information is used to calculate the common information between two random variables. Mutual information is a symmetric measure, which is defined as the following formula:

$$\begin{aligned} MI(X;Y) &= H(X)+H(Y)-H(X,Y) \\ &= -\sum_{x\in X, y\in Y} p(x,y)*\log_2\frac{p(x,y)}{p(x)p(y)} \end{aligned} \quad (5)$$

where $p(x,y)$ is the joint *probability distribution* function. According to Eq. (5), if X and Y are totally independent, which means $p(x,y) = p(x)*p(y)$, then the mutual information between X and Y become 0. On the other hand, if there is a strong relationship between X and Y then MI(X;Y) will be large. If X and Y are two continuous variables, mutual information is extended by replacing the summation by a definite double integral as below:

$$MI(X;Y) = \int_X\int_Y p(x,y)*\log_2\frac{p(x,y)}{p(x)p(y)}dx\,dy \quad (6)$$

where $p(x), p(y)$ and $p(x,y)$ are *probability density* functions.

Mutual information is also extended in many ways to measure the common information between more than two random variables. Suppose that S is a joint variable, which consists of m single variables. Multi-variate information (MvI) or interaction information is used to measure the common between all variables' information. The interaction information of a joint variable S = $\{s_1,\ldots,s_m\}$ is calculated by the Eq. (7).

$$MvI(S) = -\sum_{U\subseteq S}(-1)^{|S|-|U|}H(U) \quad (7)$$

Meanwhile, there is another extension of MI, called "total correlation information" (TCI) (Alfonso et al, 2010), which measures the common information between any variable subsets of S. Since TCI can capture the interaction between variables, it is more suitable for feature selection. TCI can be computed by using the following equation:

$$TCI(S) = \sum_{s_i\in S} H(s_i) - H(s_1,s_2,\ldots,s_m) \quad (8)$$

where m is the total number of single feature/variable (s_i) in the joint feature/variable (*S*).

2.3 Related Work on Feature Selection

2.3.1 Related Work to Feature Selection Using Non-EC Techniques

A basic version of feature selection is feature ranking (Dash and Liu, 1997), where a score is assigned to each feature according to an evaluation criterion. Feature selection can be achieved by selecting the features with the highest scores. However, this type of algorithms ignores the interaction between features. Additionally, the features with the highest scores are usually similar. Therefore, these algorithms tend to select redundant features.

Sequential search techniques are also applied to solve feature selection problems. In particular, sequential forward selection (SFS) (Whitney, 1971) and sequential backward selection (SBS) (Marill and Green, 1963) are proposed. At each step of selection process, SFS (or SBS) adds (or removes) a feature from an empty (full) feature set. Although these local search techniques achieve better performance than the feature ranking method, they might suffer "nesting" problem, in which once a feature is added (or removed) from the feature set, it cannot be removed (or added) later. In order to avoid nesting effect, Stearns (1976) proposed a "plus-l-take away-r" method in which SFS was applied l times forward and then SBS was applied for r back tracking steps. However, it is challenge to determine the best values of (l,r). This problem is addressed by sequential backward floating selection (SBFS) and sequential forward floating selection (SFFS), proposed by (Pudil et al, 1994). In SBFS ad SFFS, the values (l, r) are dynamically determined rather than being fixed in the "plus-l-take away-r" method.

2.3.2 PSO-based Feature Selection

Many ideas have been proposed to improve the performance of PSO-based feature selection algorithms. These ideas include modifications in the initialisation strategy, representation, fitness function or search mechanisms. Three initialisation strategies, which followed the sequential feature selection procedure, were proposed by Xue et al (2014). The first mechanism initialised the swarm with a small number of features. Meanwhile in the second mechanism, particles were created by using a large number of original features. In the "medium" strategy, the small initialisation was applied to the majority of the swarm and the rest followed the large initialisation. Besides initialisation, Xue et al (2014) also proposed three updating mechanisms for *gbest* and *pbest*. In comparison with the standard PSO and a two-stage PSO algorithms (Xue et al, 2012b, 2013) the proposed mechanisms evolved better feature subsets, which achieved higher classification performance with a smaller number of features. Bharti and Singh (2016) proposed a PSO based feature selection algorithm which applied opposition chaotic method. Firstly, opposition chaotic was used to initialise the swarm by selecting the top feature subsets which were generated on two opposite sides. During the evolutionary phase, opposition chaotic also helped to dynamically update the PSO parameters and mutate *gbest*, which could avoid the stagnation in local optima. The experimental results on 3 text datasets showed that the proposed algorithm could evolve informative feature subsets with in short convergent times.

Along with initialisation, representation also played an important role in PSO. A PSO representation was proposed by Vieira et al (2013) to achieve feature selection and optimise support vector machine kernel parameters at one time. Besides bits for the original feature set, the new representation had additional bits for optimising the kernel parameters. Therefore, the length of this representation was longer than the traditional one. In comparison with other binary PSO and GA based feature selection algorithms (Chuang et al, 2008; Lee et al, 2008; Huang and Wang, 2006), the proposed algorithm evolved better feature subsets, which had a smaller number of features and still achieved higher accuracy. This representation was also applied in continuous encoding (Lin et al, 2008) and a mixture of binary and continuous encoding (Boubezoul and Paris, 2012). Based on statistical clustering, Lane et al (2013) proposed a representation for PSO. Firstly, all similar features were grouped in one feature cluster. For each cluster, a single feature, which was the representative for the cluster, was selected according to the velocity of PSO's particles. Therefore, the number of selected features by the proposed algorithm was equal to the total number of clusters. The idea was extended in (Lane et al, 2014) by applying Gaussian distribution to allow more than one features selected from a cluster. Particularly, a Gaussian distribution was used to firstly determine the number of selected features (m) from each cluster and then m features with the highest velocity in a certain cluster were selected. Later, Nguyen et al (2014b) also applied statistical clustering to proposed a new representation, which had lower dimensionality than the traditional representation. Particularly, a maximum number of selected features from each cluster, which was smaller than the total features from the cluster, was determined. Each bit string belonged to a certain cluster and presented a feature index from the cluster. The experimental results indicated that the proposed algorithm achieved better classification performance and selected a smaller number of features than two other PSO-based algorithms. Although each bit in this representation was a real number, the particle still could not move smoothly in the continuous search space. This problem was addressed by a transformation rule (Nguyen et al, 2015), which based on the Gaussian distribution to form a smoother fitness landscape.

Premature convergence is a typical problem of PSO, in which the swarm is stuck in local optima. In order to avoid this problem, Chuang et al (2008) proposed a *gbest* resetting mechanism, which set all $gbest$ position's elements to zero when the best fitness did not change for a number of iterations. The experimental results showed that the resetting mechanism helped PSO to evolve a smaller set of features with higher classification accuracy than (Yang et al, 2008) in most cases. *Gbest* resetting mechanism was also used with local searches on $pbest$ in (Tran et al, 2014) to further reduce the number of selected features while still improving the classification accuracy. The efficiency of the proposed feature selection algorithm was also improved by considering the changed features only. PSO was also used with other search techniques to achieve feature selection. For instance, in (Ghamisi and Benediktsson, 2015), PSO cooperated with GAs during the evolutionary process to solve feature selection problems. Particularly, in each iteration, the top individuals were selected to be enhanced by both PSO and GAs. Therefore, in the next generation, half of the springs were from PSO and the other half were produced by GA's crossover and mutation operations. The experimental results show that the proposed algorithm could evolve informative feature subsets in acceptable computation times.

2.3.3 Information Theory-based Feature Selection

Freeman et al (2015) did a comprehensive evaluation about the effect of different filter measures on two common classification algorithms, k-nearest neighbour and support vector machine. The experimental results showed that mutual information was able to evolve good feature subsets for both classification algorithms.

Based on the idea of "Max-relevance and min-redundancy" (Peng et al, 2005), mutual information was used to form fitness functions, which aimed to find a feature subset with a minimal redundancy within the subset and a maximal relevance between the subset and the class label. Cervante et al (2012) proposed two new information theory based fitness functions. In the first fitness function, mutual information between two selected features and between a selected feature and the class label (paired evaluation) were used to respectively compute the relevance and redundancy of the feature subsets. These measures were also combined in the second fitness measure. However, in the second measure, instead of using mutual information, information gain (group evaluation) was used to calculate the relevance and redundancy of the feature subset. The results showed that both fitness functions successfully guided PSO to search for small feature subsets, which achieve better classification accuracy than using all features. The subset evolved by the first fitness function is smaller than the one evolved by the second fitness function. However the second algorithm achieved better classification performance.

Multi-objective PSO was also combined with filter measures to form multi-objective feature selection approaches. Xue et al (2012a) proposed two multi-objective PSO-based feature selection algorithms, which simultaneously minimised the number of selected features and maximised the relevance of the selected feature subset. In these algorithms, the relevant measure was calculated by applying either pair-wise mutual information or information gain. The results illustrated that the proposed multi-objective algorithms outperformed single objective algorithms. Mutual information was also applied in hybrid approaches, which took the advantages of both filters and wrappers. For instance, Nguyen et al (2014a) used mutual information as a measure to improve *gbest* by applying a local search. The local search was similar to backward feature selection since it tried to remove selected features from *gbest*. The proposed algorithms selected much smaller number of features while still achieved similar or better performance than other PSO based algorithms.

3 Mutual Information for Feature Selection

Since mutual information is able to detect non-linear interaction between multiple variables, it is widely applied to feature selection. Most of mutual information based feature selection approaches utilise mutual information to measure the redundancy and relevance of a feature subset, using two formulas, Eq. (9) and Eq. (10), respectively.

$$Red = MI(s_1, s_2, \ldots, s_m) \tag{9}$$

$$Rel = MI(S, C) \tag{10}$$

where C is the class label, S is the feature set, which contains m features $s_1, \ldots, s_m$.

The aim of feature selection is to produce an optimal feature subset by removing all redundant and irrelevant features. So the optimal feature subset minimise the quality measure given in Eq. (11).

$$F = -\alpha * Rel + (1 - \alpha) * Red \tag{11}$$

where α is used to control the contribution of relevance and redundancy into the fitness measure.

3.1 Counting approach for mutual information

According to Eq. (3) and Eq. (8), in order to calculate the mutual information between two or more variables/features, it is necessary to know the probability distribution of each variable as well as the joint probability distribution. However, it is not a trivial task in real-world problems. In most current approaches, the probability distribution is achieved by counting the number of instances in the training set. Although this approach is quite efficient, it is hard to apply it to continuous datasets since each continuous variable has an infinite number of values. So in order to be applied to continuous datasets, counting approaches require an efficient and effective way to discretise the datasets. Even with a discrete dataset, counting approaches still can not produce an accurate probability distribution of a joint of variables. Suppose that each feature s_i in the feature set S has n_i possible values, then the total number of possible values of the feature set S is $\prod_{i=1}^{m} n_i$. Therefore, in order to accurately calculate the mutual information of a feature set, it usually requires a huge number of instances in the training set. However this requirement is hardly satisfied in real-world datasets, for instance gene datasets can have up to thousands of features but a small number of samples. To adapt with the privation of samples, the relevance and redundancy measures are estimated by decomposing them into pair-wise mutual information, which can be seen in Eq. (12) and Eq. (13). As a result, a feature subset is also evaluated by using pair-wise mutual information as in Eq. (14).

$$Rel_{pw} = \sum_{i=1}^{m} MI(s_i, C) \tag{12}$$

$$Red_{pw} = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} MI(s_i, s_j) \tag{13}$$

$$F_{pw} = -\alpha * Rel_{pw} + (1 - \alpha) * Red_{pw} \tag{14}$$

In comparison with multi-variate mutual information, the pair-wise mutual information has less expensive computation cost since only the probability distribution of two variables is required. However, pair-wise mutual information can not capture the interaction between features. For example, pair-wise mutual information can not figure out the complementary feature set, in which two or more weakly-relevant features might become highly-relevant when working with each other.

3.2 Estimation approach for mutual information

Pair-wise mutual information only considers two-way interaction between features. In addition, counting approach is only applicable to discrete datasets. To overcome these limitations, mutual information estimations have been developed. The oldest and simplest estimator is the "basic histogram" (Sturges, 1926), in which each dimension corresponding to one variable is divided into many non-overlapping bins with fixed size. The probability distribution of each "bin" is calculated as a ratio between the number of observations falling into the bin and the total number of observations. Therefore, each bin is considered a possible value of a single variable or a joint variable. The entropy of each single/joint variable can be calculated by applying the discretised version given in Eq. (3) and then the mutual information can be acquired according to the formula Eq. (5). In this approach, there are two most important parameters, which are the number of bins and the bin's size.

The basic histogram is sensitive to the parameter selections. In addition, histogram approaches have sharp boundaries, which mean that two similar instances on different sides of boundary are considered different values. To avoid this discontinuity, Parzen (1962) proposed kernel density estimation (KDE). This approach estimated the probability density of each instance with a kernel function Θ, which is shown in the Eq. (15).

$$\hat{p}(S_j) = \frac{1}{N} * \sum_{j'=1}^{N} \Theta\left(|S_j - S_{j'}| - r\right) \tag{15}$$

where Θ is the kernel function and r is the kernel width, $|.|$ is a norm and *N* is the total number of instances.

The kernel function Θ measures the similarity between two instances of feature set S, S_j and $S_{j'}$. Normally, the Θ is a step function, which means that $\Theta(X > 0) = 0$ and $\Theta(X \leq 0) = 1$. The norm $|.|$ is the maximum norm. Therefore, the probability estimated by Eq. (15) is the proportion of the N instances, whose distances to the instance S_i are less than r. The entropy of the joint variable or feature subset S is then achieved by averaging the local entropy of all instances, which can be seen in the Eq. (16). The calculated entropies are plugged in Eq. (5) to derive the mutual information estimation.

$$\hat{H}(S) = \frac{1}{N} * \sum_{i=1}^{N} -\hat{p}(S_i) * \log \hat{p}(S_i) \tag{16}$$

Beside KDE, recently Kraskov et al (2004) proposed another estimation approach, called Nearest Neighbour estimation (NNE). Similar to KDE, NNE also works on each instance. The main idea of NNE is if neighbours of an instance on two dimensions X and Y are similar, then there must be a strong relationship between X and Y. Particularly, for each instance, K nearest neighbours of an instance are found to derive the distance ϵ, which is then used as a boundary to define the neighbours of the instance on each dimension (feature). The mutual information is acquired by plugging the number of neighbours on each dimension to Eq. (17)

$$\hat{MI}(S) = \psi(k) - \frac{m-1}{k} + (m-1) * \psi(N) - \frac{1}{N} * \sum_{i=1}^{N} \sum_{j=1}^{m} n_{ij} \tag{17}$$

where m is the number of single variables (features) in the variable (feature) set S, n_{ij} is the number of neighbours whose distance from the i^{th} instance S_i in the space specified by dimension (feature) s_j is not greater than $0.5 * \epsilon(i) = 0.5 * \max(\epsilon_{X_1}(i), \ldots, \epsilon_{X_m}(i))$.

Therefore, NNE can be seen as an improvement of KDE, where the boundary *r* is dynamically determined by the number of nearest neighbours *K*. Both estimators are implemented in Java Information Dynamics Toolkit (JIDT), an information-theoretic toolkit developed by Lizier (2014). In terms of computation cost, NNE is more expensive than KDE. Particularly, NNE's computation cost is *O(KN^2)*, where N is the total number of instances. Although JIDT implements k-d tree algorithm to faster search for nearest neighbours, its cost is still *O(KNlog(N))*, which is more expensive than KDE, whose time-complexity is only *O(N)* with box-assisted methods.

This work will compare between two ways to compute mutual information, including the counting approach and

Table 1: Datasets

	Dataset	Type	#Fs	#Cs	#Is
Real-world datasets	Wine	Con	13	3	178
	Vehicle	Dis	18	4	946
	German	Dis	24	2	1000
	WBCD	Con	30	2	569
	Ionosphere	Con	34	2	351
	Sonar	Con	60	2	208
	Musk 1	Dis	166	2	476
	Arrhythmia	Dis	279	16	452
Artificial datasets	Binary 1	Dis	3	2	8
	Binary 2	Dis	3	2	8
	Monk 1	Dis	6	2	432
	Monk 2	Dis	6	2	432
	Monk 3	Dis	6	2	432
	2-way linear	Con	4	2	200
	3-way linear	Con	4	2	200

the estimation approach. The KDE is chosen as the representative of estimation approaches because it is simpler, easier to understand and faster than NNE, which was used in (Nguyen et al, 2016). PSO is chosen as a feature subset generation. Each feature subset is evaluated using the pairwise fitness measure shown in Eq. (14), where both counting approach and KDE can be applied.

3.3 PSO representation for feature selection

The representation of a particle in PSO is a vector of n real numbers, where n is the total number of features. Each position entry x_{id} falls in the range [0,1] and corresponds to the d^{th} feature in the original feature set. A threshold θ is used to determine whether or not a feature is selected: if $x_{id} > \theta$ then the d^{th} feature is selected, otherwise the d^{th} feature is not selected.

4 Design of Experiments

4.1 Datasets

In this work, KDE and counting approach will be compared in both artificial and real-world datasets. All datasets can be seen in the Table 1, where “Con” and “Dis” mean respectively continuous and discrete datasets, #Fs means the total number of features, #Cs means the total number of class values and #Is is the total number of available instances. There are 8 real-world datasets, which are original from UCI repository (Asuncion and Newman, 2007). These datasets contain different number of features and instances. The continuous datasets are discretised so that the counting approach can be applied.

There are 7 different artificial datasets, which have different relationships between features and between features and the class labels. The first two artificial datasets have three binary features. In Binary 1, an instance belongs to class 1 if exactly two features have value 1, otherwise the instance is in class 0. In Binary 2, if all instances' features have the same value then it is in class 1, otherwise it belongs to class 0. So in these two datasets, there is no redundancy and all three features are relevant to the class label. Feature selection on these datasets should select all three features.

Three other artificial datasets are Monk datasets (Asuncion and Newman, 2007), which have 6 discrete features and one binary class label. The 3^{rd} and 6^{th} features are binary variables, which can be either 1 or 2. The 5^{th} feature has four possible values from 1 to 4. The other features have three values, which range from 1 to 3. In Monk 1 dataset, the class label is 1 if either $f_0 = f_1$ or $f_4 = 1$. So the optimal feature set of Monk 1 is $\{f_0, f_1, f_4\}$. Meanwhile, in Monk 2, the class label is 1 if there are exactly two features taking value 1. In this case, all features are important in Monk 2 dataset. The last Monk dataset is a bit more complicated, where the class label is 1 if ($f_3 = 1$ and $f_4 = 3$) or ($f_4 \neq 4$ and $f_1 \neq 3$). So in Monk 3 datasets, the most important feature subset is $\{f_1, f_3, f_4\}$. Notice that there is no redundancy in the Monk datasets.

2-way linear and 3-way linear also have 4 continuous features. In 2-way linear, the last two features are copies of the first two features ($f_0 = f_2, f_1 = f_3$). The class label is set to 1 if the average of the first two features is greater than 0.5. Therefore, the optimal feature subset for this dataset is one of 4 feature subsets, $\{f_0, f_1\}$, $\{f_0, f_3\}$,$\{f_1, f_2\}$ or $\{f_2, f_3\}$. In 3-way linear dataset, the first two features are two random variables, which fall in [0,1]. The 3^{rd} feature is the average of the first two features, $f_2 = \dfrac{f_0 + f_1}{2}$. The 4^{th} feature (f_3) is just a copy of the first feature. So in this dataset, there is redundancy in any feature subsets that contains f_0 along with f_3 or (f_1 and f_2). The class label is determined by feature f_2. Particularly, the class label is set to 1 if $f_2 > 0.5$. So the optimal feature subset for this dataset is $\{f_2\}$.

4.2 Parameter setting

Each dataset is divided into 10 folds. Each fold will be selected as a test set and the other folds are used as a training set to select features. This process is run 30 independent times. So there will be 300 evolved feature subsets. Since each dataset has continuous and discrete versions, the selected feature subsets are tested on both version using three classification algorithms K-nearest neighbour (K=5) (KNN), Decision Tree (DT) and Naive Bayes (NB).

The kernel width *r* needs to satisfy the condition $K_r \leq N/(3/r)^m$, where K_r is the number of neighbours fall in the range r and m is the number of dimensions or the number of features and N is the total number of instances. In this case, since only pair-wise mutual information is used, the number of dimensions m is 2. Lungarella et al (2005) proposed that K_r should be at least equal to 3 to avoid undersampling effects. Therefore, in this work K_r is set to 3. From the above conditions, the kernel width r is specified by $\dfrac{3}{\log_2 N/3}$.

The weight α in the pair-wise fitness measure (Eq. (14)) has three different values: 0.6, 0.8 and 1.0 to evaluate the effect of different relevance and redundancy's contributions.

For PSO algorithm, the fully connected topology is used. The parameters are set as follows (Van Den Bergh, 2006): $w = 0.7298, c_1 = c_2 = 1.49618, v_{max} = 6.0$. The population size is 30 and the maximum number of iterations is 100. The threshold θ is set as 0.6.

5 Results and Discussion

Experimental results on real-world and artificial datasets are shown in Tables 2 and 3, respectively. Each table is the results of PSO using counting and KDE approach on a dataset. The prefix "Con-" and "Dis-" correspond to the results on the continuous and discrete versions of each dataset. The significant test between KDE and counting approach is shown in the brackets, beside KDE's accuracies. "+", "=" or "-" mean that KDE approach is respectively significantly better, similar or significantly worse than counting approach. Table 4 shows which features are selected by either KDE or counting approaches on artificial datasets.

5.1 Real-world datasets

The results on the 8 real-world datasets are shown in Table 2.

5.1.1 Consistency of KDE and Counting Approach

As can be seen from the results, in most datasets the order of classification accuracies is preserved after the feature selection process. For example, in Vehicle dataset, the highest classification accuracy belongs to DT classifier and KNN is the second best classifier. After performing feature selection using either KDE or counting approach, the best classifier is still DT, which is followed by KNN. This consistency is an evident that mutual information is not bias to any classification algorithm among the three classification algorithms. Mutual information is able to extract a general feature subset, which is meaningful to all the three classification algorithms.

5.1.2 KDE vs Counting approach on Real-world Datasets

In terms of the classification accuracy, KDE is significantly better than counting approach in the continuous version of most of datasets. For example, in the Wine dataset, the classification accuracy of KDE is about 10% better than counting approach on both DT and NB classification algorithms. In addition, in the Ionosphere and Sonar datasets, by applying to KNN classification algorithm, the feature subsets generated by KDE achieve up to 10% better than counting approach regardless the similar number of selected features. In WBCD, KDE is significantly better than counting approach in all the three classification algorithms when α is set to 0.6 and 0.8. In summary, on the continuous version of datasets, in almost all cases KDE achieves similar or better performance than counting approach in the three classification algorithms.

On the discrete version of each dataset, KDE also achieves similar or better performance than counting approach. In most cases, KDE outperforms counting approach when α is set to 0.8. For example, in Vehicle dataset (Table (2b)), the improvements of KDE over counting method on KNN, DT and NB are 4.5%, 4% and 7% respectively. Despite of selecting the same number of features, with $\alpha = 0.8$, KDE's accuracies on all the three classification algorithms are up to 1% higher than the results of counting approach. The experimental results show that KDE is not only able to cope with both continuous and discrete datasets but also similar or better than the counting approach, which only works well with discrete datasets.

In terms of the number of selected features, when α increases, which means the contribution of redundancy into the fitness function decreases, the number of selected features of both approaches also increases. The extreme case is when redundancy is ignored (α = 1.0), in the datasets with small number of features, almost all original features are selected. Meanwhile, when the number of original features is larger, the proportion of selected features is smaller. The reason might be a dataset with a large number of features might contains many irrelevant features. However, the smaller number of selected features with respect to lower contribution of redundancy does not mean that redundancy measure Red_{pw} works well in this case. The reason is that Red_{pw}, which is shown in Eq. (13), is a monotonic function. Regardless of which features are selected, according to Eq. (13) adding any feature into the feature subset will results in additional MI, which might increases Red_{pw} because mutual information is non-negative. So in this case, it only can be confirmed that PSO does find out optimal or near-optimal feature subsets when $\alpha = 1.0$. It would be hard to analyse the effect of Rel_{pw} and Red_{pw} in the real datasets since the optimal feature subset is unknown. Therefore, a deep analysis on the artificial datasets is provided in the next section.

Table 2: Test accuracies on real-world datasets.

(a) Wine

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	13	94.38	80.94	86.86	13	93.25	97.76	97.78
0.6	Counting	1.0	82.52	80.48	67.72	2.24	92.58	93.06	92.7
	KDE	2.56	92.84(+)	81.69(=)	75.71(+)	2.24	92.42(=)	93.01(=)	92.63(=)
0.8	Counting	1.0	83.12	81.17	67.9	4.98	93.72	96.91	96.3
	KDE	4.98	95.61(+)	80.98(=)	81.15(+)	4.98	93.74(=)	96.79(=)	96.31(=)
1.0	Counting	11.92	94.48	80.81	85.84	11.99	93.45	97.08	97.39
	KDE	11.95	94.51(=)	80.76(=)	86.04(=)	12.01	93.46(=)	97.06(=)	97.33(=)

(b) Vehicle

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	18	85.93	83.04	81.32	18	85.93	83.04	81.32
0.6	Counting	1.02	75.8	75.01	71.12	1.02	75.8	75.01	71.12
	KDE	1.97	75.17(-)	74.41(=)	74.39(+)	1.97	75.17(-)	74.41(=)	74.39(+)
0.8	Counting	1.18	76.96	76.07	71.73	1.18	76.96	76.07	71.73
	KDE	3.83	81.43(+)	80.1(+)	78.69(+)	3.83	81.43(+)	80.1(+)	78.69(+)
1.0	Counting	15.96	85.49	82.47	81.18	15.96	85.49	82.47	81.18
	KDE	16.25	85.43(=)	82.46(=)	81.33(=)	16.25	85.43(=)	82.46(=)	81.33(=)

(c) German

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	24	74.2	68.2	73.5	24	74.2	68.2	73.5
0.6	Counting	3.2	70.11	67.53	70.66	3.2	70.11	67.53	70.66
	KDE	3.02	70.88(+)	68.36(=)	71.33(+)	3.02	70.88(+)	68.36(=)	71.33(+)
0.8	Counting	4.97	71.81	70.09	72.39	4.97	71.81	70.09	72.39
	KDE	5.03	72.4(+)	71.0(+)	72.97(+)	5.03	72.4(+)	71.0(+)	72.97(+)
1.0	Counting	19.76	73.95	68.69	73.18	19.76	73.95	68.69	73.18
	KDE	19.98	74.21(=)	68.95(=)	73.58(=)	19.98	74.21(=)	68.95(=)	73.58(=)

(d) WBCD

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	30	94.73	93.32	88.57	30	91.91	96.49	94.38
0.6	Counting	1.37	88.21	87.32	75.93	2.07	92.09	91.74	92.73
	KDE	2.14	91.81(+)	90.31(+)	84.76(+)	2.07	92.07(=)	91.75(=)	92.73(=)
0.8	Counting	1.9	90.25	89.93	80.72	4.21	93.0	94.39	94.87
	KDE	3.79	93.49(+)	90.9(+)	89.26(+)	4.2	93.02(=)	94.41(=)	94.85(=)
1.0	Counting	24.96	94.14	92.94	88.49	25.01	92.31	96.06	94.19
	KDE	24.83	94.21(=)	93.04(=)	88.79(=)	25.0	92.35(=)	96.07(=)	94.18(=)

(e) Ionosphere

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	34	89.17	84.33	35.9	34	90.87	85.19	90.58
0.6	Counting	2.4	81.52	79.7	81.17	2.31	84.89	84.49	84.65
	KDE	2.37	84.1(+)	84.71(+)	83.3(+)	2.25	84.75(=)	84.42(=)	84.54(=)
0.8	Counting	2.65	80.01	78.01	81.68	4.03	88.93	89.11	89.22
	KDE	4.08	87.75(+)	88.12(+)	80.86(=)	4.0	88.89(=)	89.17(=)	89.24(=)
1.0	Counting	27.87	88.53	83.73	35.9	27.55	90.59	84.89	90.55
	KDE	27.75	89.03(=)	84.14(+)	35.9(=)	27.59	90.65(=)	84.89(=)	90.56(=)

(f) Sonar

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	60	74.0	80.17	50.38	60	72.57	85.07	75.98
0.6	Counting	1.57	57.48	56.88	51.86	2.11	63.87	62.06	64.29
	KDE	2.18	61.7(+)	62.03(+)	52.32(=)	2.13	63.41(=)	61.64(=)	63.83(=)
0.8	Counting	1.58	57.27	57.87	52.09	2.69	68.3	67.11	68.38
	KDE	2.67	67.21(+)	67.05(+)	50.64(=)	2.69	68.46(=)	67.05(=)	68.58(=)
1.0	Counting	46.29	72.96	80.22	51.11	45.99	73.13	83.75	75.19
	KDE	45.79	72.81(=)	80.54(=)	50.02(-)	45.91	73.35(+)	83.66(=)	75.15(=)

(g) Musk1

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	166	74.59	86.97	65.36	166	64.59	86.97	65.36
0.6	Counting	11.21	73.13	76.46	68.91	11.2	73.11	76.51	68.98
	KDE	11.81	73.67(=)	76.91(=)	68.03(=)	11.81	73.67(=)	76.91(=)	68.03(=)
0.8	Counting	11.19	73.05	76.31	69.23	11.24	73.17	76.4	69.28
	KDE	12.06	73.65(=)	76.92(=)	68.28(=)	12.06	73.65(=)	76.92(=)	68.28(=)
1.0	Counting	113.27	75.59	85.9	74.89	113.27	75.59	85.93	74.89
	KDE	113.7	75.1(=)	86.01(=)	74.92(=)	113.7	75.1(=)	86.01(=)	74.92(=)

(h) Arrhythmia

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	278	94.86	93.57	94.96	278	94.86	93.57	94.96
0.6	Counting	41.85	93.71	93.3	93.75	41.85	93.71	93.3	93.75
	KDE	41.79	93.71(=)	93.29(=)	93.75(=)	41.79	93.71(=)	93.29(=)	93.75(=)
0.8	Counting	42.42	93.91	93.48	93.85	42.42	93.91	93.48	93.85
	KDE	42.24	93.89(=)	93.5(+)	93.84(=)	42.24	93.89(=)	93.5(+)	93.84(=)
1.0	Counting	174.63	94.67	93.75	95.01	174.63	94.67	93.75	95.01
	KDE	174.67	94.67(=)	93.75(=)	95.01(=)	174.67	94.67(=)	93.75(=)	95.01(=)

5.2 Artificial datasets

Tables 3 and 4 show respectively the test accuracies and the feature subsets selected by applying Counting and KDE algorithms on 7 artificial datasets. The results of two datasets Binary 1 and Binary 2 are not shown because DT, KNN and NB are not able to classify the problems (0% accuracy). In terms of classification accuracy, as can be seen from table 3, KDE achieves similar of significantly better results than counting approach. The largest difference between the two approaches is in Monk 1 dataset, where KDE's accuracies are about 25% better than counting's accuracies.

The more important factor to be considered in the artificial datasets is the feature subsets evolved. For each α value, feature selection algorithms are run 30 times on each dataset. Each independent run uses 10-folds approach. Therefore there will be 300 (30×10) feature subsets generated for each α value and each dataset. The feature subsets selected by KDE and counting approaches are shown in Table 4. In the table all indexes of selected features are in the curly brackets, which follows by the number of times the feature subset is selected. For example, $\langle\{0,1,2,3\} : 300\rangle$ means that the feature subset $\{0,1,2,3\}$ are selected *300* times.

In two Binary datasets, the optimal set is the original feature set. According to the experimental results, regardless of the values of α, the original feature set is selected by KDE in more than 98% of the 300 times. Because there is no redundancy in these datasets, the α values should not affect on the evolved feature subsets. Therefore, the redundancy measured by KDE works well in this case. For counting approach, the proportion of the original feature set to all feature subset ranges from 20% to 100% when α increases from 0.6 to 1.0. When redundancy contributes to the fitness function, counting approach still results in a smaller set than the optimal set regardless of the fact that redundancy should be 0. Therefore, it can be seen that redundancy measure by counting approach does not work well in Binary datasets. Particularly, redundancy between two independent feature, measured by counting approach is greater than 0.

In Monk 1 dataset, the optimal feature subset is $\{f_0, f_1, f_4\}$ and there is no redundancy in this dataset. Three features f_2, f_3 and f_5 are irrelevant to the class label. Once more, since the redundancy in this dataset is 0, the α values should not affect on the selected feature subsets. This fact is completely reflected by the KDE approach, which selects the optimal subset $\{f_0, f_1, f_4\}$ all the 300 times. Meanwhile, counting approach selects very different feature subsets even within the same α values. In all α values, f_2 and f_3 appears frequently in the feature subsets, which indicates that the relevance measure by counting approach still gives some score to these irrelevant features. An obvious evidence is that the counting approach selects all features when $\alpha = 1$, which means the irrelevant features are selected. For Monk 2 datasets, it is important to select all original features. According to the experimental results, for all values of α, KDE always selects no less than 5 features, in which all features are selected more than 280 times out of the 300 times. Meanwhile, the size of feature subset selected by the counting approach ranges from 3 to 6 features. In Monk 3 dataset, the most complicated Monk dataset, the optimal feature subset is $\{f_1, f_3, f_4\}$, which is also selected by KDE in all cases regardless of the α values. Meanwhile, the counting approach still selects irrelevant features like f_0, f_2 and f_5 very frequently. So with the Monk datasets, it can be seen that the counting approach is not able to detect irrelevant features, but this is done very well by the KDE approach.

In the rest two artificial datasets, 2-way and 3-way linear datasets, there is no irrelevant feature but there are redundant

Table 3: Test accuracies on artificial datasets.

(a) Monk 1

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	6	85.87	94.21	75.0	6	85.87	94.21	75.0
0.6	Counting	1.59	75.0	63.22	75.0	1.59	75.0	63.22	75.0
	KDE	3.0	99.77(+)	100.0(+)	75.0(=)	3.0	99.77(+)	100.0(+)	75.0(=)
0.8	Counting	2.79	75.0	66.79	75.0	2.79	75.0	66.79	75.0
	KDE	3.0	99.77(+)	100.0(+)	75.0(=)	3.0	99.77(+)	100.0(+)	75.0(=)
1.0	Counting	5.94	85.88	93.2	75.0	5.94	85.88	93.2	75.0
	KDE	3.0	99.77(+)	100.0(+)	75.0(=)	3.0	99.77(+)	100.0(+)	75.0(=)

(b) Monk 2

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	6	79.63	69.46	66.45	6	79.63	69.46	66.45
0.6	Counting	4.67	65.96	57.58	66.26	4.67	65.96	57.58	66.26
	KDE	5.94	78.92(+)	68.7(+)	66.48(+)	5.94	78.92(+)	68.7(+)	66.48(+)
0.8	Counting	5.24	69.83	62.04	66.24	5.24	69.83	62.04	66.24
	KDE	5.94	78.92(+)	68.7(+)	66.48(+)	5.94	78.92(+)	68.7(+)	66.48(+)
1.0	Counting	5.95	78.84	68.74	66.46	5.95	78.84	68.74	66.46
	KDE	5.94	78.92(=)	68.7(=)	66.48(=)	5.94	78.92(=)	68.7(=)	66.48(=)

(c) Monk 3

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	6	100.0	99.54	97.23	6	100.0	99.54	97.23
0.6	Counting	3.29	100.0	100.0	97.23	3.29	100.0	100.0	97.23
	KDE	3.0	100.0(=)	100.0(=)	97.23(=)	3.0	100.0(=)	100.0(=)	97.23(=)
0.8	Counting	3.97	100.0	100.0	97.23	3.97	100.0	100.0	97.23
	KDE	3.0	100.0(=)	100.0(=)	97.23(=)	3.0	100.0(=)	100.0(=)	97.23(=)
1.0	Counting	5.97	100.0	99.53	97.23	5.97	100.0	99.53	97.23
	KDE	3.0	100.0(=)	100.0(+)	97.23(=)	3.0	100.0(=)	100.0(+)	97.23(=)

(d) 2-way linear

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	4	92.5	94.5	46.0	4	92.5	94.5	46.0
0.6	Counting	1.0	69.5	69.3	54.0	1.0	69.5	69.3	54.0
	KDE	2.0	92.5(+)	94.5(+)	54.0(=)	2.0	92.5(+)	94.5(+)	54.0(=)
0.8	Counting	1.0	69.5	69.3	54.0	1.0	69.5	69.3	54.0
	KDE	2.0	92.5(+)	94.5(+)	54.0(=)	2.0	92.5(+)	94.5(+)	54.0(=)
1.0	Counting	4.0	92.5	94.5	46.0	4.0	92.5	94.5	46.0
	KDE	2.0	92.5(=)	94.5(=)	54.0(+)	2.0	92.5(=)	94.5(=)	54.0(+)

(e) 3-way linear

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	4	99.5	95.5	52.0	4	99.5	95.5	52.0
0.6	Counting	1.0	80.4	76.13	48.0	1.0	80.4	76.13	48.0
	KDE	2.8	99.5(+)	95.0(+)	48.0(=)	2.8	99.5(+)	95.0(+)	48.0(=)
0.8	Counting	1.0	80.4	76.13	48.0	1.0	80.4	76.13	48.0
	KDE	2.8	99.5(+)	95.0(+)	48.0(=)	2.8	99.5(+)	95.0(+)	48.0(=)
1.0	Counting	4.0	99.5	95.5	52.0	4.0	99.5	95.5	52.0
	KDE	2.8	99.5(=)	95.0(-)	48.0(-)	2.8	99.5(=)	95.0(-)	48.0(-)

features. In 2-way linear dataset, the class label can be determined by one of the following feature subsets $\{f_0, f_1\}$, $\{f_0, f_3\}$, $\{f_1, f_2\}$ and $\{f_2, f_3\}$, which are also the only 4 feature subsets selected by KDE. On the other hand, the counting approach always select a single feature when α is set to 0.6 or 0.8. Once more the result shows that the redundancy between two independent features is not correctly calculated by the counting approach. In addition, KDE approach is able to detect the complementary feature subsets, although it is a hard problem when pair-wise fitness function is used. In the 3-way linear dataset, once more the counting approach always select a single feature when α is less than 1.0. On the other hand, KDE selects only 4 feature subsets, which are $\{f_1,f_2,f_3\}$, $\{f_0,f_1,f_2\}$, $\{f_0,f_1\}$ and $\{f_1, f_3\}$. As

Table 4: Feature sets selected by KDE and Counting approaches.

(a) Binary 1

	Counting	KDE
α =0.6	$\langle\{0,1,2\}:90\rangle$, $\langle\{0,1\}:73\rangle$, $\langle\{1,2\}:46\rangle$, $\langle\{2\}:30\rangle$, $\langle\{0\}:30\rangle$, $\langle\{1\}:30\rangle$, $\langle\{0,2\}:1\rangle$,	$\langle\{0,1,2\}:293\rangle$, $\langle\{0\}:3\rangle$, $\langle\{1\}:3\rangle$, $\langle\{2\}:1\rangle$,
α =0.8	$\langle\{0,1,2\}:210\rangle$, $\langle\{0,1\}:42\rangle$, $\langle\{0,2\}:39\rangle$, $\langle\{1,2\}:9\rangle$,	$\langle\{0,1,2\}:297\rangle$, $\langle\{1\}:2\rangle$, $\langle\{2\}:1\rangle$,
α =1.0	$\langle\{0,1,2\}:300\rangle$,	$\langle\{0,1,2\}:297\rangle$, $\langle\{2\}:1\rangle$, $\langle\{0\}:1\rangle$, $\langle\{1\}:1\rangle$,

(b) Binary 2

	Counting	KDE
α =0.6	$\langle\{0\}:100\rangle$, $\langle\{2\}:85\rangle$, $\langle\{0,1,2\}:60\rangle$, $\langle\{1\}:35\rangle$, $\langle\{1,2\}:10\rangle$, $\langle\{0,1\}:8\rangle$, $\langle\{0,2\}:2\rangle$,	$\langle\{0,1,2\}:300\rangle$,
α =0.8	$\langle\{0,2\}:75\rangle$, $\langle\{0,1\}:68\rangle$, $\langle\{0,1,2\}:61\rangle$, $\langle\{1,2\}:58\rangle$, $\langle\{2\}:17\rangle$, $\langle\{0\}:15\rangle$, $\langle\{1\}:6\rangle$,	$\langle\{0,1,2\}:300\rangle$,
α =1.0	$\langle\{0,1,2\}:240\rangle$, $\langle\{2\}:18\rangle$, $\langle\{0\}:15\rangle$, $\langle\{1,2\}:9\rangle$, $\langle\{0,1\}:8\rangle$, $\langle\{1\}:8\rangle$, $\langle\{0,2\}:2\rangle$,	$\langle\{0,1,2\}:300\rangle$,

(c) Monk 1

	Counting	KDE
α =0.6	$\langle\{4\}:122\rangle$, $\langle\{3,4\}:118\rangle$, $\langle\{1,4\}:57\rangle$, $\langle\{2,4\}:3\rangle$,	$\langle\{0,1,4\}:300\rangle$,
α =0.8	$\langle\{0,3,4\}:60\rangle$, $\langle\{1,2,4\}:39\rangle$, $\langle\{4\}:30\rangle$, $\langle\{1,4\}:30\rangle$, $\langle\{3,4\}:30\rangle$, $\langle\{0,2,4,5\}:30\rangle$, $\langle\{3,4,5\}:30\rangle$, $\langle\{1,2,3,4\}:28\rangle$, $\langle\{2,3,4\}:19\rangle$, $\langle\{2,4,5\}:2\rangle$, $\langle\{1,3,4\}:2\rangle$,	$\langle\{0,1,4\}:300\rangle$,
α =1.0	$\langle\{0,1,2,3,4,5\}:281\rangle$, $\langle\{0,1,2,3,4\}:6\rangle$, $\langle\{1,2,3,4,5\}:4\rangle$, $\langle\{0,2,3,4,5\}:4\rangle$, $\langle\{0,1,3,4,5\}:4\rangle$, $\langle\{0,1,2,4,5\}:1\rangle$,	$\langle\{0,1,4\}:300\rangle$,

(d) Monk 2

	Counting	KDE
α =0.6	$\langle\{0,1,3,4\}:98\rangle$, $\langle\{0,1,2,3,4\}:59\rangle$, $\langle\{0,1,2,3,4,5\}:59\rangle$, $\langle\{0,1,3,4,5\}:54\rangle$, $\langle\{0,1,3\}:28\rangle$, $\langle\{0,1,4\}:2\rangle$,	$\langle\{0,1,2,3,4,5\}:282\rangle$, $\langle\{0,2,3,4,5\}:7\rangle$, $\langle\{0,1,2,3,5\}:6\rangle$, $\langle\{0,1,2,4,5\}:4\rangle$, $\langle\{0,1,2,3,4\}:1\rangle$,
α =0.8	$\langle\{0,1,2,3,4,5\}:114\rangle$, $\langle\{0,1,3,4,5\}:81\rangle$, $\langle\{0,1,2,3,4\}:62\rangle$, $\langle\{0,1,3,4\}:42\rangle$, $\langle\{0,1,3,5\}:1\rangle$,	$\langle\{0,1,2,3,4,5\}:282\rangle$, $\langle\{0,2,3,4,5\}:7\rangle$, $\langle\{0,1,2,3,5\}:6\rangle$, $\langle\{0,1,2,4,5\}:4\rangle$, $\langle\{0,1,2,3,4\}:1\rangle$,
α =1.0	$\langle\{0,1,2,3,4,5\}:285\rangle$, $\langle\{0,1,2,3,4\}:11\rangle$, $\langle\{0,1,3,4,5\}:4\rangle$,	$\langle\{0,1,2,3,4,5\}:282\rangle$, $\langle\{0,2,3,4,5\}:7\rangle$, $\langle\{0,1,2,3,5\}:6\rangle$, $\langle\{0,1,2,4,5\}:4\rangle$, $\langle\{0,1,2,3,4\}:1\rangle$,

(e) Monk 3

	Counting	KDE
α =0.6	$\langle\{1,3,4\}:214\rangle$, $\langle\{1,3,4,5\}:57\rangle$, $\langle\{1,2,3,4\}:29\rangle$,	$\langle\{1,3,4\}:300\rangle$,
α =0.8	$\langle\{1,2,3,4\}:88\rangle$, $\langle\{1,3,4,5\}:87\rangle$, $\langle\{1,3,4\}:66\rangle$, $\langle\{0,1,2,3,4\}:28\rangle$, $\langle\{0,1,3,4,5\}:28\rangle$, $\langle\{0,1,3,4\}:3\rangle$,	$\langle\{1,3,4\}:300\rangle$,
α =1.0	$\langle\{0,1,2,3,4,5\}:290\rangle$, $\langle\{0,1,2,3,4\}:6\rangle$, $\langle\{1,2,3,4,5\}:4\rangle$,	$\langle\{1,3,4\}:300\rangle$,

(f) 2-way linear

	Counting	KDE
α =0.6	$\langle\{0\}:110\rangle$, $\langle\{3\}:80\rangle$, $\langle\{2\}:70\rangle$, $\langle\{1\}:40\rangle$,	$\langle\{2,3\}:110\rangle$, $\langle\{1,2\}:81\rangle$, $\langle\{0,1\}:60\rangle$, $\langle\{0,3\}:49\rangle$,
α =0.8	$\langle\{0\}:110\rangle$, $\langle\{3\}:80\rangle$, $\langle\{2\}:70\rangle$, $\langle\{1\}:40\rangle$,	$\langle\{2,3\}:110\rangle$, $\langle\{1,2\}:80\rangle$, $\langle\{0,1\}:60\rangle$, $\langle\{0,3\}:50\rangle$,
α =1.0	$\langle\{0,1,2,3\}:300\rangle$,	$\langle\{2,3\}:110\rangle$, $\langle\{1,2\}:80\rangle$, $\langle\{0,1\}:60\rangle$, $\langle\{0,3\}:50\rangle$,

(g) 3-way linear

	Counting	KDE
α =0.6	$\langle\{0\} : 110\rangle$, $\langle\{3\} : 80\rangle$, $\langle\{2\} : 70\rangle$, $\langle\{1\} : 40\rangle$,	$\langle\{1,2,3\} : 144\rangle$, $\langle\{0,1,2\} : 96\rangle$, $\langle\{0,1\} : 32\rangle$, $\langle\{1,3\} : 28\rangle$,
α =0.8	$\langle\{0\} : 110\rangle$, $\langle\{3\} : 80\rangle$, $\langle\{2\} : 70\rangle$, $\langle\{1\} : 40\rangle$,	$\langle\{1,2,3\} : 144\rangle$, $\langle\{0,1,2\} : 96\rangle$, $\langle\{0,1\} : 32\rangle$, $\langle\{1,3\} : 28\rangle$,
α =1.0	$\langle\{0,1,2,3\} : 300\rangle$,	$\langle\{1,2,3\} : 144\rangle$, $\langle\{0,1,2\} : 96\rangle$, $\langle\{0,1\} : 32\rangle$, $\langle\{1,3\} : 28\rangle$,

Table 5: Computation time on real-world datasets

Datset	KDE time (ms)	Counting time (ms)
Wine	344.49	38.74
Vehicle	244.7	1.64
German	145.45	2.19
Wbcd	6288.96	88.69
Ionosphere	4941.29	98.97
Sonar	6819.77	187.77
Musk1	253546.83	424.43
Arrhythmia	4020.61	36.22
Binary 1	0.77	0.5
Binary 2	0.75	0.46
Monk 1	39.49	0.55
Monk 2	69.78	0.64
Monk 3	43.83	0.55
2-way linear	8.01	0.45
3-way linear	11.11	0.57

can be seen KDE never selects f_0 and f_3 together because they are redundant. According to the linear datasets, KDE is able to detect the complementary feature subset and remove the redundant features, which can not be done by the counting approach.

The experimental results suggests that KDE for mutual information works well on both continuous and discrete datasets. The feature subsets generated by KDE achieve similar or better performance than the counting approach. The main reason is the counting approach can not correctly calculate the redundancy measure and detect the complementary interaction between features, which can be achieved by using KDE.

5.3 Computation Cost

The computation costs of KDE and counting approach are shown in Table 5. As can be seen from the table, KDE is more expensive than the counting approach. The reason is that in order to calculate the mutual information, KDE needs to calculate the distance from each instance to all other instances to find out the number of neighbours of an instance, which is about N times slower than the counting approach (N is the total number of available instances).

6 Conclusions and Future Work

Although mutual information has been widely applied to feature selection, it is limited to discrete datasets, which requires discretising continuous datasets. Mutual information estimation has been developed to allow mutual information to directly work on continuous datasets without any pre-processing step. The goal of this paper is to compare between estimation and counting approach in cooperation with PSO to achieve feature selection. The experimental results show that mutual information estimation is able to capture the interaction between features to evolve optimal feature subsets. In addition, mutual information estimation also works well in both continuous and discrete versions of datasets. Meanwhile the counting approach provides good accuracy only in the discrete datasets and it is fail to measure the redundancy between features.

However, in terms of efficiency, mutual information estimation is still slower than the counting approach. In order to improve mutual information estimation's efficiency, it is important to develop instance selection algorithms along with feature selection algorithms, which is left for our future work.

References

Alfonso L, Lobbrecht A, Price R (2010) Optimization of water level monitoring network in polder systems using information theory. Water Resources Research 46(12)

Asuncion A, Newman D (2007) Uci machine learning repository

Bharti KK, Singh PK (2016) Opposition chaotic fitness mutation based adaptive inertia weight bpso for feature selection in text clustering. Applied Soft Computing 43:20–34

Boubezoul A, Paris S (2012) Application of global optimization methods to model and feature selection. Pattern Recognition 45(10):3676–3686

Cervante L, Xue B, Zhang M, Shang L (2012) Binary particle swarm optimisation for feature selection: A filter based approach. In: Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE

Chuang LY, Chang HW, Tu CJ, Yang CH (2008) Improved binary pso for feature selection using gene expression data. Computational Biology and Chemistry 32(1):29–38

Dash M, Liu H (1997) Feature selection for classification. Intelligent data analysis 1(3):131–156

Dash M, Liu H, Motoda H (2000) Consistency based feature selection. In: Knowledge Discovery and Data Mining. Current Issues and New Applications, Springer, pp 98–109

Duda RO, Hart PE, Stork DG (2012) Pattern classification. John Wiley & Sons

Eberhart RC, Shi Y (1998) Comparison between genetic algorithms and particle swarm optimization. In: Evolutionary Programming VII, Springer, pp 611–616

Freeman C, Kulić D, Basir O (2015) An evaluation of classifier-specific filter measure performance for feature selection. Pattern Recognition 48(5):1812–1826

Ghamisi P, Benediktsson JA (2015) Feature selection based on hybridization of genetic algorithm and particle swarm optimization. Geoscience and Remote Sensing Letters, IEEE 12(2):309–313

Hall M (2000) Correlation-based feature selection for discrete and numeric class machine learning, proceedings of 7th intentional conference on machine learning, stanford university

Huang CL, Wang CJ (2006) A ga-based feature selection and parameters optimizationfor support vector machines. Expert Systems with applications 31(2):231–240

Jaynes ET (1957) Information theory and statistical mechanics. Physical review 106(4):620

Kennedy J, Eberhart R, et al (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Perth, Australia, vol 4, pp 1942–1948

Kohavi R, John GH (1997) Wrappers for feature subset selection. Artificial intelligence 97(1):273–324

Kononenko I (1995) On biases in estimating multi-valued attributes. In: IJCAI, vol 95, pp 1034–1040

Kraskov A, Stögbauer H, Grassberger P (2004) Estimating mutual information. Physical review E 69(6):066,138

Lane MC, Xue B, Liu I, Zhang M (2013) Particle swarm optimisation and statistical clustering for feature selection. In: AI 2013: Advances in Artificial Intelligence, Springer, pp 214–220

Lane MC, Xue B, Liu I, Zhang M (2014) Gaussian based particle swarm optimisation and statistical clustering for feature selection. In: Evolutionary Computation in Combinatorial Optimisation, Springer, pp 133–144

Lee S, Soak S, Oh S, Pedrycz W, Jeon M (2008) Modified binary particle swarm optimization. Progress in Natural Science 18(9):1161–1166

Lin SW, Ying KC, Chen SC, Lee ZJ (2008) Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert systems with applications 35(4):1817–1824

Lizier JT (2014) Jidt: An information-theoretic toolkit for studying the dynamics of complex systems. arXiv preprint arXiv:14083270

Lungarella M, Pegors T, Bulwinkle D, Sporns O (2005) Methods for quantifying the informational structure of sensory and motor data. Neuroinformatics 3(3):243–262

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

Nguyen H, Xue B, Liu I, Zhang M (2014a) Filter based backward elimination in wrapper based pso for feature selection in classification. In: Evolutionary Computation (CEC), 2014 IEEE Congress on, pp 3111–3118

Nguyen HB, Xue B, Liu I, Zhang M (2014b) Pso and statistical clustering for feature selection: A new representation. In: Simulated Evolution and Learning, Springer, pp 569–581

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

Nguyen HB, Xue B, Andreae P (2016) Mutual information estimation for filter based feature selection using particle swarm optimization. In: Applications of Evolutionary Computation, Springer, pp 719–736

Parzen E (1962) On estimation of a probability density function and mode. The annals of mathematical statistics pp 1065–1076

Peng H, Long F, Ding C (2005) Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. Pattern Analysis and Machine Intelligence, IEEE Transactions on 27(8):1226–1238

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

Stearns SD (1976) On selecting features for pattern classifiers. In: Proceedings of the 3rd International Conference on Pattern Recognition (ICPR 1976), Coronado, CA, pp 71–75

Sturges HA (1926) The choice of a class interval. Journal of the American Statistical Association 21(153):65–66

Tran B, Xue B, Zhang M (2014) Improved pso for feature selection on high-dimensional datasets. In: Simulated Evolution and Learning, Springer, pp 503–515

Van Den Bergh F (2006) An analysis of particle swarm optimizers. PhD thesis, University of Pretoria

Vieira SM, Mendonça LF, Farinha GJ, Sousa JM (2013) Modified binary pso for feature selection using svm applied to mortality prediction of septic patients. Applied Soft Computing 13(8):3494–3504

Walters-Williams J, Li Y (2009) Estimation of mutual information: A survey. In: Rough Sets and Knowledge Technology, Springer, pp 389–396

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

Xue B, Cervante L, Shang L, Browne WN, Zhang M (2012a) A multi-objective particle swarm optimisation for filter-based feature selection in classification problems. Connection Science 24(2-3):91–116

Xue B, Zhang M, Browne WN (2012b) New fitness functions in binary particle swarm optimisation for feature selection. In: Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE, pp 1–8

Xue B, Zhang M, Browne WN (2013) Novel initialisation and updating mechanisms in PSO for feature selection in classification. Springer

Xue B, Zhang M, Browne WN (2014) Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. Applied Soft Computing 18:261–276

Xue B, Zhang M, Browne W, Yao X (2015) A survey on evolutionary computation approaches to feature selection. Evolutionary Computation, IEEE Transactions on PP(99):1–1

Yang CS, Chuang LY, Ke CH, Yang CH (2008) Boolean binary particle swarm optimization for feature selection. In: IEEE Congress on Evolutionary Computation (CEC), pp 2093–2098