Evolutionary Sparsity Regularisation-based Feature Selection for Binary Classification

Bach Hoai Nguyen Hoai.Bach.Nguyen@ecs.vuw.ac.nz

Bing Xue Bing.Xue@ecs.vuw.ac.nz

Mengjie Zhang Mengjie.Zhang@ecs.vuw.ac.nz

School of Engineering and Computer Science, Faculty of Engineering, Victoria University of Wellington, Wellington, New Zealand

Abstract

In classification, feature selection is an essential pre-processing step that selects a small subset of features to improve classification performance. Existing feature selection approaches can be divided into three main approaches: wrapper approaches, filter approaches, and embedded approaches. In comparison with two other approaches, embedded approaches usually have better trade-off between classification performance and computation time. One of the most well-known embedded approaches is sparsity regularisation-based feature selection which generates sparse solutions for feature selection. Despite its good performance, sparsity regularisation-based feature selection outputs only a feature ranking which requires the number of selected features to be predefined. More importantly, the ranking mechanism introduces a risk of ignoring feature interactions which leads to the fact that many top-ranked but redundant features are selected. This work addresses the above problems by proposing a new representation that considers the interactions between features and can automatically determine an appropriate number of selected features. The proposed representation is used in a differential evolutionary (DE) algorithm to optimise the feature subset. In addition, a novel initialisation mechanism is proposed to let DE consider various numbers of selected features at the beginning. The proposed algorithm is examined on both synthetic and real-world datasets. The results on the synthetic dataset show that the proposed algorithm can select complementary features while existing sparsity regularisation-based feature selection algorithms are at risk of selecting redundant features. The results on real-world datasets show that the proposed algorithm achieves better classification performance than well-known wrapper, filter, and embedded approaches. The algorithm is also as efficient as filter feature selection approaches.

Keywords

Sparse regularisation, feature selection, classification, differential evolution

1 Introduction

The goal of classification is to determine the class of an instance based on its features. The classification performance relies on the quality of the features. In many modern classification tasks, there are a large number of features available. Unfortunately, with large feature sets, many features are irrelevant to the class label. Using such features may deteriorate the classification performance significantly due to their misleading information (Zhao et al., 2009). Besides irrelevant features, large feature sets may contain redundant features providing the same information as other features. Including redundant features usually does not improve classification performance while causing a

longer training time. Additionally, since the number of possible instances in the data space increases exponentially with respect to the number of features, a large number of features causes the available data to be sparse. Therefore, it requires a huge amount of data to obtain an accurate classifier, which is known as the "curse of dimensionality" (Keogh and Mueen, 2017). To improve the feature quality, feature selection removes the irrelevant and redundant features, which reduces the number of features and improves the classification performance (Guyon and Elisseeff, 2003).

A feature selection algorithm typically has two main components: subset discovery and subset evaluation. The first component, subset discovery, is responsible for generating promising feature subsets. The second component, subset evaluation, is responsible for evaluating the goodness of the feature subsets generated by the subset discovery. It is essential to have an efficient and effective subset discovery method since the total number of possible subsets increases exponentially with respect to the number of features. By considering the feedback from the subset evaluation, the subset discovery is expected to generate more promising feature subsets and search the space more efficiently. Besides the large search space, feature selection is challenging due to the complex interactions between features. For example, individually relevant features may provide the same information about the class label, so selecting them together results in redundancy. On the other hand, combining weakly relevant features may form a significantly relevant feature set. Hence, to achieve a reliable selection performance, the two components need to consider feature interactions.

Based on the subset evaluation, feature selection methods can be divided into three main categories: wrapper approaches, filter approaches, and embedded approaches (Li et al., 2018; Dash and Liu, 1997; Tang et al., 2014). Wrapper approaches use a specific classification algorithm to evaluate feature subsets, thus they achieve high classification performance since the selected features are tailored to the wrapped classifier. However, wrappers are computationally expensive due to having to train classifiers in each evaluation process. On the other hand, filter approaches evaluate feature subsets based on the intrinsic characteristics of a dataset. Since filters do not involve any classification process in their evaluations, they are usually more efficient than wrappers. However, filters usually do not achieve as good classification performance as wrappers (Li et al., 2018). Embedded approaches perform feature selection during the process of training a classifier. For example, building a decision tree can be considered an embedded feature selection approach because it selects features at its internal nodes. Embedded approaches generally provide a better trade-off between selection performance and computation time than the other two approaches (Li et al., 2018). Therefore, this paper focuses on developing an embedded approach for feature selection.

Among existing embedded feature selection approaches, sparsity regularisation-based methods have gained much attention from the feature selection community because of their good performance and interpretability (Liu et al., 2009; Nie et al., 2010; Xiang et al., 2012). The main idea of sparsity regularisation-based methods is to find an optimal weight vector $\boldsymbol{\omega}$ for the features such that the classification loss and the vector's regularisation are minimised. The role of regularisation is to balance between the bias and the variance of the learned model (Bishop, 2006). Since the variance is controlled by the regularisation, sparsity regularisation-based methods can partially avoid overfitting. It has been shown that by using ℓ_p-norm ($0 \leq p \leq 1$) as the regularisation function, the learned weight vector becomes sparse where many vector elements are very small, in some cases, exactly 0 (Keogh and Mueen, 2017; Peng and Fan, 2016). This property is suitable for feature selection since each vector element can be considered a

feature coefficient, and coefficients can be used to select features (Li et al., 2018).

Although existing sparsity regularisation-based methods achieve good feature selection performance, they have two main limitations. Firstly, the values in the sparse vector ω constitute a ranking of features but do not specify how many features to select, thus the number of selected features needs to be predefined. However, this number is problem dependent and usually not known in advance. More importantly, due to the ranking mechanism, some feature interactions might be ignored. For example, selecting two low-ranked features with small coefficients might significantly improve the classification performance if they are complementary and provide information together that is not available from any individual features.

Evolutionary computation (EC) has been widely applied to feature selection because of its potential global search ability (Xue et al., 2016). As one of feature subset selection approaches, EC-based feature selection can evaluate a whole feature subset which considers feature interactions. Among EC methods, differential evolution (DE) has been shown to achieve better performance than other EC methods in many areas (Das and Suganthan, 2010). Also compared with other EC methods, DE is simpler to understand and has fewer parameters to be tuned, which makes it easier to be used by experts from a wide range of fields. Therefore, this work utilises DE to address the above two limitations of sparsity regularisation-based feature selection.

1.1 Goal

The overall goal of this paper is to develop an embedded feature selection method utilising both sparsity regularisation and DE to efficiently select small feature subsets, which achieve similar or better classification performance than using all features on binary classification problems. To achieve this goal, we propose a hybrid representation with two components, which build feature coefficients and feature subsets, respectively. The first component ensures that the selected features are capable of building a reliable classifier; the second component allows DE to automatically determine the number of selected features and to consider feature interactions. The fitness function is similar to the objective function in support vector machine (SVM) where Hinge loss is used to measure the classification loss of the selected features. Thus the proposed algorithm can be considered an SVM-embedded feature selection algorithm. We first focus on binary classification problems since SVM is primarily for binary classification. The proposed algorithm is first evaluated on a synthetic dataset to analyse its capability to detect feature interactions. We then examine our proposed algorithm on 14 real-world datasets of varying difficulties. Specifically, we will investigate:

- whether the proposed algorithm selects a smaller number of features while maintaining or even improving the classification performance over using all features,
- whether the proposed algorithm can select better feature subsets with shorter computational time than wrapper-based feature selection using SVM as the wrapped classification algorithm,
- whether the proposed algorithm can select more complementary features, hence achieving better classification performance than three state-of-the-art sparsity regularisation-based embedded approaches: RFS (Nie et al., 2010), GFS (Peng and Fan, 2017a), and OEC (Bonyadi and Reutens, 2019), and
- whether the proposed algorithm can achieve better performance than three well-known filter feature selection algorithms: mRMR (Peng et al., 2005), CFS (Hall and

Smith, 1999), and reliefF (Robnik-Šikonja and Kononenko, 2003).

1.2 Notations in this paper

We summarise the notations used in this paper. Let $\boldsymbol{X} \in \mathcal{R}^{m \times n}$ be a feature data containing m instances and n features. We can express $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_j, \ldots, \boldsymbol{x}_m)$ where $\boldsymbol{x}_j$ is the j^{th} sample. Alternatively, we also express $\boldsymbol{X} = [\boldsymbol{X}_1; \boldsymbol{X}_2; \ldots; \boldsymbol{X}_i; \ldots; \boldsymbol{X}_n]$ where $\boldsymbol{X}_i$ is all values of the i^{th} feature in the dataset. The label vector is $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ where $y_j \in \{-1, +1\}, j \in \{1, 2, \ldots, m\}$ (binary classification). In the paper, vectors are denoted by bold lower-case letters and matrices are denoted by bold capital letters. The expression "$< \boldsymbol{a}, \boldsymbol{b} >$" means horizontally appending the vector $\boldsymbol{b}$ to the vector $\boldsymbol{a}$ to form a new vector.

2 Background

This section firstly gives an overview of feature selection and related work on feature selection. After that, sparsity regularisation-based feature selection, the main focus of this paper, is described in more details. The final subsection provides a brief description of JADE (Zhang and Sanderson, 2009), an adaptive DE algorithm, which is used as the optimisation technique of the proposed algorithm

2.1 Feature selection for classification

The main goal of feature selection is to select a small feature subset from n original features such that the selected features can maintain or improve the classification performance than using all features. Since the total number of possible feature subsets is 2^n, the search space of feature selection increases exponentially with the number of features. Feature selection can be roughly divided into three main categories: filter approaches, wrapped approaches, and embedded approaches. The main difference between the three categories is the way they evaluate the goodness of a feature subset candidate.

Filter approaches evaluate feature subsets based on the intrinsic characteristics of the dataset. The main idea is to have a measure which can assess the redundancy between features, and/or the relevance between features and the class label. Correlation (CFS) (Hall and Smith, 1999), distance (reliefF) (Robnik-Šikonja and Kononenko, 2003), and information gain (mRMR) (Peng et al., 2005) are three well-known filter measures, which are used as representatives of filter approaches in this work.

Wrapper approaches rely on a classification algorithm to evaluate feature subsets. Some common wrapped classification algorithms are K-nearest neighbours (KNN) (Wang et al., 2015; Gu et al., 2018; Tran et al., 2018; Nguyen et al., 2019), Decision Tree (DT) (Zhang et al., 2014b), and Support Vector Machine (SVM) (Chen and Chen, 2015; Wei et al., 2017; Tuba et al., 2019). Although wrapper approaches usually achieve good classification performance with the wrapped classification algorithm, they are computationally intensive since they repeatedly train the wrapped classification algorithm.

Embedded feature selection is in the middle of filters an wrappers. Embedded approaches usually achieve better classification performance than filters and have a faster selection process than wrappers. The main idea is to perform feature selection during the process of building a classifier which combines the advantage of both filter and wrapper approaches. A good representative of embedded feature selection is sparsity based feature selection (Kim and Kim, 2004; Yan and Yang, 2015) which has gained considerable attention recently due to its good performance (Li et al., 2018). Thus, this work focuses on developing a novel sparsity based feature selection algorithm. The

following subsection discusses sparsity based feature selection in more details.

2.2 Sparsity based feature selection

The main goal of sparsity regularisation-based feature selection is to learn a sparse solution consisting of feature coefficients which minimises the fitting error. The sparse solution can be achieved by a sparse regularisation term which forces many feature coefficients to be close to 0. Given a feature data, $\boldsymbol{X} \in \mathcal{R}^{m \times n}$ where m is the number of samples and n is the number of features, a label vector $\boldsymbol{y} = \{y_1, y_2, \ldots, y_m\}$, sparsity based feature selection can be written as the following optimisation problem:

$$\min_{\mathcal{H}} loss(\mathcal{H}; \boldsymbol{X}, \boldsymbol{y}) + \alpha \times reg(\mathcal{H}) \tag{1}$$

where $\mathcal{H}$ represents the solution, $loss(\cdot)$ is a loss function, $reg(\cdot)$ is a regularisation term of $\mathcal{H}$, and α is a regularisation parameter. The loss function can be least squared loss, Hinge loss, or logistic loss.

For binary classification problems, $\mathcal{H}$ is represented by a weight vector $\boldsymbol{w} \in \mathcal{R}^n$, and the regularisation is an ℓ_p-norm where $0 \leq p \leq 1$ to force a sparse vector $\boldsymbol{w}$. Hence, the objective function for feature selection is:

$$\min_{\boldsymbol{w}} loss(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) + \alpha ||\boldsymbol{w}||_p \tag{2}$$

where $||\boldsymbol{w}||_p = (\sum_{i=1}^{n} |w_i^p|)^{1/p}$. If $p = 0$, the l_0-norm function counts the number of nonzero entries (features) in $\boldsymbol{w}$. However, l_0-norm leads to an integer programming problem which is difficult to solve. It is shown that l_1-norm has a similar effect as l_0-norm since it forces many entries to be much smaller, sometimes close to 0 (Tibshirani, 1996). Thus, l_1 norm has been widely applied to achieve feature selection (Xu et al., 2014; Wei et al., 2016; Hara and Maehara, 2017). The features can be ranked by the absolute values of their coefficients in $\boldsymbol{w}$.

For multi-class classification problems, $\mathcal{H}$ is represented by a weight matrix $\boldsymbol{W} \in \mathcal{R}^{n \times c}$ where c is the number of classes, and the regularisation is an $\ell_{p,q}$-norm. Hence, the objective function for feature selection is:

$$\min_{\boldsymbol{W}} loss(\boldsymbol{W}; \boldsymbol{X}, \boldsymbol{y}) + \alpha ||\boldsymbol{W}||_{p,q} \tag{3}$$

where $||\boldsymbol{W}||_{p,q} = (\sum_{k=1}^{c} (\sum_{i=1}^{n} W_{i,k}^p)^{q/p})^{1/q}$. Most existing works set $p = 2$ and vary q in the range [0, 1] to obtain a sparse matrix $\boldsymbol{W}$ (Peng and Fan, 2016; Zhang et al., 2014a). The features can be ranked according to the value of $||\boldsymbol{W}_i||_2^2$. An early work RFS (Robust Feature Selection) (Nie et al., 2010) is proposed to use $\ell_{2,1}$-norm for both the loss function and the regularisation term, which is more robust to noise. Thus, RFS achieves a good classification performance. Later, Peng and Fan (Peng and Fan, 2017b) show that it is not necessary to use the same norm for both loss function and regularisation term. In their proposed algorithm (GFS), $\ell_{2,r}$-norm ($0 < r \leq 2$) is used for the loss function and $\ell_{2,q}$-norm ($0 < q < 1$) is used for the regularisation term. The results show that the norm flexibility could improve the feature selection performance.

The main drawback of existing sparsity regularisation-based feature selection approaches is their ranking mechanisms where features are ranked according to their coefficients. In many cases, top-ranked features might be redundant features and it is not necessary to select all of them together. It is also possible that the combination of low-ranked features might form a complementary feature set which is significantly relevant

to the class label. Furthermore, most sparsity regularisation-based feature selection algorithms are gradient-based, so they can be trapped at local optima when the optimisation problem is non-convex, for example when p and q are smaller than 1. Thus, it is necessary to develop a global sparsity regularisation-based feature selection algorithm which considers feature interactions. One option is to use EC as an optimisation approach to enhance the algorithm's global search ability. In fact, many EC techniques including genetic algorithms (GA), particle swarm optimisation (PSO), and differential evolution (DE) have been widely applied to achieve features selection (Xue et al., 2016). However, most of them are either filter or wrapper approaches which either achieve low classification performance or have an expensive computation cost. This work proposes an embedded feature selection algorithm which uses EC, more specifically DE, to achieve sparsity regularisation-based feature selection. Since standard EC techniques usually do not work well on optimisation problems with a huge number of decision variables, we first focus only on binary classification problems which have a relatively smaller number of feature coefficients to optimise (Cheng et al., 2016). The following subsection briefly describes DE.

2.3 Differential evolution

Differential evolution (DE) is an evolutionary algorithm proposed by Storn and Price (Storn and Price, 1997). DE maintains a population P of NP individuals, $\{z_1, z_2, \ldots, z_{NP}\}$ where each individual encodes a candidate solution. In DE, each individual z_i is represented by a numeric vector $\{z_{i1}, z_{i2}, \ldots, z_{in}\}$ where i is the individual index ($1 \leq i \leq NP$), n is the problem's dimension, and $z_j^{min} \leq z_{ij} \leq z_j^{max}$ ($1 \leq j \leq n$). Typically, DE starts with a random population and its evolutionary process is a loop of three operations: mutation, crossover, and selection which are described as follows:

- Mutation: for each individual z_i, this operation generates a mutant vector v_i based on some other solutions randomly selected from the current population.

- Crossover: this operation builds a trial vector u_i by applying a binomial crossover between z_i and v_i. Each entry of u_i can be obtained by the following formula:

$$u_{ij} = \begin{cases} v_{ij} & \text{, if rand(0,1)} \leq CR_i \text{ or } j = j_{rand}, \\ z_{ij} & \text{, otherwise} \end{cases}$$

  where rand(0,1) is a random number between 0 and 1, $j_{rand} = randint(1, n)$ is an integer randomly selected between 1 and n, an $CR_i \in [0, 1]$ is used to roughly control the average fraction of vector entries that the trial vector u can inherit from the mutant vector v_i.

- Selection: u_i will replace z_i if it has a better fitness value than z_i. Otherwise, z_i is not changed. If z_i is replaced, it is called a *successful update*.

JADE is an enhanced version of DE where a new mutation scheme called "DE/current-to-$p$best" is implemented and its parameters are adaptively controlled. JADE also records a set of inferior solutions, called A, which are solutions being replaced by the selection operation. The "DE/current-to-$p$best" mutation generates the mutant vector u_i for the individual z_i by the following equation:

$$u_i = z_i + F_i \times (z_{best}^p - z_i) + F_i \times (z_{r1} - \tilde{z}_{r2}) \tag{4}$$

where z_{r1}, is selected from P, z^p_{best} are selected from the top p% solutions of P, and $\tilde{z}_{r2}$ is selected from the union P $\cup$ A. Such mutation mechanism is designed to balance between the convergence rate and the diversity of the population.

For the i^{th} individual, DE has two main parameters CR_i and F_i which are adaptively generated by two normal distributions $\mathcal{N}(\mu_{CR}, 0.1)$ and $\mathcal{N}(\mu_F, 0.1)$, respectively. Both μ_{CR} and μ_F are initialised to 0.5, and updated at the end of each generation as

$$\mu_{CR} = (1-c) \times \mu_{CR} + c \times \text{mean}_A(S_{CR}) \quad (5)$$

$$\mu_F = (1-c) \times \mu_F + c \times \text{mean}_L(S_F) \quad (6)$$

where $c \in (0, 1)$ is a positive constant, S_{CR} and S_F are the sets of crossover probabilities and mutation factors, respectively, which are used in all successful updates of the current generation, $\text{mean}_A(\cdot)$ is the standard mean, and $\text{mean}_L(\cdot)$ is the Lehmer mean. JADE is shown to be better than standard DE and PSO in a wide range of benchmark functions (Zhang and Sanderson, 2009). Therefore, we use JADE as the underlying optimisation technique in this work.

3 Proposed method

The main goal of this paper is to develop an efficient and effective sparsity regularisation-based feature selection algorithm which can directly output a set of complementary features. In the proposed algorithm (named DEEFS), JADE is used as the search mechanism due to its potential global search ability. This is significantly different from existing sparsity regularisation-based feature selection algorithms which are mainly gradient-based. However, JADE is a general optimisation algorithm, and it needs to be tailored to work well on feature selection. There are three main issues to be considered:

- How to represent the sparsity regularisation-based feature selection in JADE?
- How to evaluate each candidate solution?
- How to tailor an initialisation mechanism which works better than random initialisation when applying JADE to achieve sparsity regularisation-based feature selection?

The three questions lead to three main contributions of this paper, which will be shown in the three following subsections. The first subsection describes a new representation that learns feature coefficients and selection decision simultaneously. The second subsection describes a fitness function that is similar to the objective function of SVM. The third subsection describes a novel initialisation mechanism which lets DEEFS consider various numbers of features at the beginning. The last subsection gives an overview of a feature selection system using DEEFS.

3.1 Representation

A nice property of existing sparsity based feature selection algorithms is the ability to build a classifier (hyperplane) during the process of selecting features, which does not require an intensive computation time but ensure that the selected features are discriminative. In these algorithms, the weight vectors play two important roles: feature coefficients for the hyperplane and decision to include or exclude features. The selection decision is determined after the feature coefficients have been trained. Particularly,

the features are ranked by their coefficients and a number of top-ranked features are selected. Note that the number of selected features is determined by users. We can show, using the synthetic data (in Section V), that only selecting features with higher weights results in redundancy. On the other hand, several features have smaller coefficients but selecting them together can significantly improve the classification performance. In addition, the weight vector itself cannot determine how many features should be selected. To address the above two issues, we propose a new hybrid representation containing two components: the weight component to evolve the classifier weights and the decision component to decide which features to be selected. The idea of training weights while still having decision components has been used in some neural network frameworks for feature selection. Balın et al. (2019) use a concrete selector layer , which is sampled based on concrete random variables, to select input nodes. Lemhadri et al. (2021) use a feed-forward neural networks with additional input-to-output connections. A feature can have non-zero weight in the hidden layer if the feature's connection is active. Note that the above methods still require to pre-define the number of selected features. Oehmcke and Gieseke (2022) optimise a selection mask that can select an arbitrary amount of features.

The proposed algorithm is inspired by SVM for binary classification, where a hyperplane is built to separate samples from two classes. The hyperplane is represented by a weight vector which is optimised by optimising the weight component. Each element of the weight vector is the weight of an original feature. An additional element is used to optimise the bias b of the hyperplane. Hence, the first component (weight component) is a vector consisting of $n + 1$ elements where n is the number of original features. The second component (decision component) is also a numeric vector with n elements where each vector element represents a decision to include or exclude an original feature. There are two encoding mechanisms for the second component: using a binary value of 0 or 1 to represent the decision or using a continuous value between 0 and 1 along with a fixed threshold value $\theta \in (0, 1)$. Note that for continuous one, the threshold value θ is defined before the selection process and JADE can adapt to θ to automatically determine the number of selected features. Both encoding mechanisms have been widely applied to feature selection. In this work, we adopt the continuous encoding mechanism for the decision component since JADE is designed for continuous optimisation. In summary, the proposed representation is a numeric vector consisting of $(2n+1)$ elements: $<\boldsymbol{w}, b, \boldsymbol{d}> = <w_1, w_2, \ldots, w_n, b, d_1, d_2, \ldots, d_n>$, where w_i and d_i are the weight and decision value of the ith feature ($i \in \{1, 2, \ldots \ldots, n\}$), b is the bias value.

The proposed representation is essentially different from the existing representations for feature selection. Most stochastic (mainly EC-based) feature selection algorithms adopt the decision component as their representations. The selected feature subsets need to be evaluated by a dedicated fitness function which is based on either a specific classification algorithm (wrappers) or a measure of a specific data property (filters). Such a representation usually leads to either a heavy computational cost of wrappers or low classification performance of filters. On the other hand, existing sparsity regularisation-based feature selection algorithms use the weight component as its representation, and select only top-ranked features based on their weights. Therefore, sparsity regularisation-based feature selection is at risk of limiting feature interactions. In contrast, the proposed hybrid representation can not only learn the feature decision but also learn the classification model (weight vectors). Thus, such representation enables to build an evolutionary embedded feature selection approach which is expected

to select discriminative features without a need of dedicated measures as in wrappers and filters. In comparison with sparsity regularisation-based feature selection, the proposed representation considers more feature interactions since it allows to consider various combinations of features regardless of the feature weights. Furthermore, although the weight vector with ℓ_1 regularisation can help to remove irrelevant features, it is still at risk of selecting relevant features that are also redundant. In this case, it is expected that the decision component can remove the such relevant but redundant features by setting their corresponding decision values to 0. Finally, the decision component can automatically determine the number of selected features, which cannot be done by existing sparsity regularisation-based feature selection.

3.2 Fitness function

Given an individual <$\boldsymbol{w}, b, \boldsymbol{d}$>, its fitness is calculated based on the following equation:

$$Fitness(\boldsymbol{w}, b, \boldsymbol{d}) = \sum_{j=1}^{m} h(y_j, \boldsymbol{x_j}, \boldsymbol{w}, b, \boldsymbol{d}) + \alpha \times reg(\boldsymbol{w}, b, \boldsymbol{d}) \tag{7}$$

where m is the number of training instances, $h(y_j, \boldsymbol{x_j}, \boldsymbol{w}, b, \boldsymbol{d})$ is the *loss* function which measures the difference between the desired label y_j and the label of x_j predicted by the hyperplane defined by <$\boldsymbol{w}, b, \boldsymbol{d}$>, and $reg(\cdot)$ and α are the regularisation function and regularisation factor, respectively.

To understand how to calculate the two functions $h(\cdot)$ and $reg(\cdot)$, we firstly show how <$\boldsymbol{w}, b, \boldsymbol{d}$> can be used to build a classifier or a hyperplane. Firstly, based on $\boldsymbol{w}$ and $\boldsymbol{d}$, the final weight vector of the n original features is obtained by

$$\boldsymbol{\omega} = \boldsymbol{w} \cdot I(\boldsymbol{d}) =< w_1 \times I(d_1), w_2 \times I(d_2), \ldots, w_n \times I(d_n) > \tag{8}$$

where

$$I(z) = \begin{cases} 1 & \text{, if } z > \theta \\ 0 & \text{, otherwise} \end{cases} \tag{9}$$

For example, given:

- $\boldsymbol{w}$ = <0.8, 0.01, 0.2, 0.3>,
- $\boldsymbol{d}$ = <0.5, 0.7, 0.1, 0.8>, and θ = 0.6

then, $I(\boldsymbol{d})$ = <0, 1, 0, 1> indicates the 2^{nd} and 4^{th} features are selected. The final weight vector is $\boldsymbol{\omega}$ = <0, 0.01, 0, 0.3>. It can be seen that although the 1^{st} feature has the largest weight, it is still not selected due to the decision vector. The obtained final weight vector $\boldsymbol{\omega}$ can be used to predict the class label of $\boldsymbol{x_j}$ by the following formula:

$$o_j = \begin{cases} 1 & \text{, if } \boldsymbol{\omega} \cdot \boldsymbol{x_j} + b > 0 \\ -1 & \text{, otherwise} \end{cases} \tag{10}$$

Inspired by the Hinge loss as in SVM, we propose to calculate the classification loss as:

$$h(y_j, \boldsymbol{x_j}, \boldsymbol{w}, b, \boldsymbol{d}) = \max(0, 1 - o_j \times y_j) \tag{11}$$

Basically, $h(\cdot)$ outputs 2 if the jth instance is correctly classified, otherwise it outputs 0. Thus, $h(\cdot)$ represents the number of wrongly classified instances that reflects the classification error more accurate than the Hinge loss. Note that $h(\cdot)$ is a discrete function

that cannot be easily addressed by convex optimisation algorithms such as gradient descents

We also adopt ℓ_1 as the regularisation function since it encourages a more sparse weight vector. Thus, $reg(\boldsymbol{w}, b, \boldsymbol{d}) = \|\boldsymbol{\omega}\|_1$.

In short, the fitness value of an individual <$\boldsymbol{w}, b, \boldsymbol{d}$> can be obtained by:

$$Fitness(\boldsymbol{w}, b, \boldsymbol{d}) = \sum_{j=1}^{m} \max\left(0, 1 - o_j \times y_j\right) + \alpha\|\boldsymbol{\omega}\|_1 \tag{12}$$

where $\boldsymbol{\omega}$ and o_j can be obtained by Eq. (8) and Eq. (10), respectively.

3.3 Initialisation

Although the proposed representation has several advantages over existing representations for feature selection, it has a higher dimensionality which results in a larger search space. To cope with such an issue, we propose a novel initialisation mechanism with an expectation that a better starting point would assist JADE to explore the search space better. Since each individual can be seen as a hyperplane in SVM, we utilise SVM to initialise the population. In addition, the boundary values of the weight component are also set in the initialisation phase.

Firstly, based on the population size NP, we define a set of NP selection ratios (a fraction between the number of selected features and the total number of features): $\{1/NP, 2/NP, 3/NP, \ldots, NP/NP\}$. The initialisation goes through each ratio value, r, and perform the following steps:

- Step 1: define the number of selected features n_r based on the ratio value r by $n_r = \lceil r \times n \rceil$;
- Step 2: randomly select n_r features to form a feature subset S_r;
- Step 3: generate *a decision vector* $\boldsymbol{d}$ = <$d_1, d_2, \ldots, d_n$> where $d_i = \begin{cases} 1 & \text{if } i \in S_r \\ 0 & \text{otherwise} \end{cases}$
- Step 4: extract a training data based on the feature subset S_r;
- Step 5: train an SVM classification algorithm on the extracted training data;
- Step 6: use the weight vector and the bias value of the obtained SVM classifier to form *a weight vector* <$\boldsymbol{w}, b$> (note that weights of unselected features are set to 0);
- Step 7: add the vector <$\boldsymbol{w}, b, \boldsymbol{d}$> as a new individual to the population;

The above initialisation has two main advantages. Firstly, it allows JADE to start with various numbers of selected features, which is expected to well cover the search space. Secondly, for each feature subset, an SVM classifier is utilised to generate a more reliable weight vector instead of starting with a random vector. In addition, based on the initial population, we set the boundary values for the weight component as $[w_{min}, w_{max}]$ where w_{max} is the largest absolute weight value of the initialised population, and $w_{min} = -w_{max}$. Pseudocode of the proposed initialisation mechanism can be seen in Algorithm 1.

Algorithm 1 : Ratio initialisation for DEEFS

Input: NP (population size), $Train$ (training data)
Output: Pop (initial population), $[w_{min}, w_{max}]$

begin
$Pop = \emptyset$;
$MaxWeight = -1$
for $r = \{1/NP, 2/NP, \ldots, NP/NP\}$ **do**
    $n_r = \lceil r \times n \rceil$;
    randomly select n_r features to form S_r;
    $\boldsymbol{d} = <d_1, d_2, \ldots, d_n>$ where $d_i = \begin{cases} 1 & \text{if } i \in S_r \\ 0 & \text{otherwise} \end{cases}$
    extract a training data $Train_r$ based on S_r;
    $<\boldsymbol{w}, b>$ = train SVM on $Train_r$;
    add $<\boldsymbol{w}, b, \boldsymbol{d}>$ to Pop;
    $MaxWeight = \max(MaxWeight, w_1, w_2, \ldots, w_n)$
end for
w_{max} = $MaxWeight$;
w_{min} = -$MaxWeight$
end

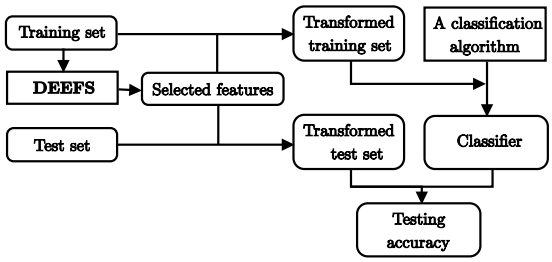


Figure 1: Overall feature selection system where DEEFS selects features based on the training set and the selected features are examined on the test set.

Table 1: Dataset Statistics.

Source	Dataset	#Features	#Instances
UCI	Parkinson	22	195
	German	24	1000
	WBCD	30	569
	Sonar	60	208
	Musk1	166	476
	LSVT	310	126
	Madelon	500	2600
Gene expression	Colon	2000	62
	DLBCL	5469	77
	ALLAML	7129	72
	CNS	7129	60
	Leukemia	7129	72
	Prostate	10509	102
	Ovarian	15154	253

3.4 Overall system

The overall feature selection system is shown in Fig. 1, where DEEFS is the proposed feature selection algorithm using JADE as its search mechanism. Firstly, a dataset is divided into a training set and a test set. Based on the training set, DEEFS outputs a subset of features which are selected by its best candidate solution. The evolved feature subset is then used to transform both training and test sets. Finally, a classification algorithm is trained on the transformed training set. The obtained classifier is applied to the transformed test set to obtain a testing accuracy which is used as the goodness of the selected features. It should be noted that the test set should never be used in the selection process to avoid feature selection bias. This work focuses on binary classification with a relatively small number of feature coefficients to optimise which was discussed in Section II-B.

4 Experiment design

4.1 Benchmark datasets

In this work, the proposed algorithm is examined on both synthetic datasets and real-world datasets. For the synthetic dataset, the optimal feature subset and the feature interactions are known, which is helpful to analyse the behaviour of the proposed algorithm. Details of the synthetic dataset can be seen in Section V. The proposed algorithm is also evaluated on 14 real-world datasets, where seven datasets are selected from UCI repository (Lichman, 2013). The UCI datasets are selected from different real-world areas such as health (Parkinson, WBCD), finance (German), physic/chemistry (Sonar, Musk1). The remaining seven datasets are gene-expression datasets with thousands of features (Han et al., 2015; Shukla et al., 2018). The selected datasets have different numbers of features and instances, which can be seen in Table 1. More comparisons on large-scale datasets are shown in the supplementary material.

4.2 Benchmark techniques

Firstly, we compare our proposed algorithm, DEEFS, with using all features on both SVM with a linear kernel and KNN (K=5). We also compare DEEFS with three state-of-the-art embedded feature selection algorithms and three well-known filter feature

selection algorithms.

The three embedded feature selection algorithms are sparsity regularisation-based algorithms including:

- RFS (Robust Feature Selection) (Nie et al., 2010) uses $\ell_{2,1}$-norm on both loss function and regularisation, which is robust to outliers,
- GFS (General sparsity regularisation Feature Selection) (Peng and Fan, 2017b) uses $\ell_{2,r}$-norm ($0 < r \leq 2$) on loss function and $\ell_{2,p}$-norm ($0 < p \leq 1$) on regularisation. It can be seen that GFS is more generalised than RFS, which is expected to boost the model sparsity better than RFS, and
- OEC (Optimal-margin Evolutionary Classifier) (Bonyadi and Reutens, 2019) applies EC to evolve a hyperplane with a minimal classification risk. Since OEC uses ℓ_1-norm on regularisation, it can output sparse weight vectors. Thus, OEC can be utilised as an embedded feature selection algorithm, as claimed in (Bonyadi and Reutens, 2019).

The three filter feature selection algorithms are:

- CFS (Hall and Smith, 1999) - a representative of correlation based methods,
- mRMR (Peng et al., 2005) - a representative of information based methods, and
- reliefF (Robnik-Šikonja and Kononenko, 2003) - a representative of distance based methods.

We also compare our proposed algorithm with a wrapper feature selection method, where SVM is used as the wrapped classification algorithm. To ensure a relatively fair comparison, we also use JADE as the search mechanism. The representation is exactly the same as the decision component of DEEFS, where each individual is a numeric vector and the vector elements correspond to the original features. The element values indicate whether the corresponding features are selected or not. We also use a standard fitness function for wrapper feature selection, which can be seen as following:

$$Fitness_w(S) = \beta \times Err + (1 - \beta) \times \frac{\#\text{selected features}}{\#\text{original features}} \tag{13}$$

where Err is the balanced classification error rate of the feature subset S using the SVM classifier, β is to control the contribution between the classification error and the selected ratio ($0 \leq \beta \leq 1$). Note that Eq. (13) is used in the benchmark wrapper feature selection algorithm, while the proposed algorithm, DEEFS, uses Eq. (12).

The algorithms are examined by using 5-fold cross validation which allows every instance has chance to be in the training set and the test set while not violating feature selection bias. Such design is to achieve reliable and generalised comparisons, especially on datasets with very few instances such as gene expression datasets. Particularly, each dataset is divided into 5 folds. Each algorithm selects features using 4 folds as a training set. Based on the selected features, a classification algorithm is tested on the remaining fold (as a test set) to obtain the balanced accuracy of the selected features. The final performance of each algorithm is the average accuracy of the five obtained balance accuracies. EC-based algorithms are run 30 independent times on each dataset. Wilcoxon rank sum test and Friedman test are used to compare the benchmark algorithms and the proposed algorithm.

4.3 Parameter settings

The parameters of JADE are set as recommendation in its original paper (Zhang and Sanderson, 2009). The population size and maximum number of iterations are set to 100, α (in Eq. (12)) is set to 0.01 to guide the algorithm focus more on the classification performance. The sensitivity analysis of α will be discussed later. The regularisation parameters of RFS and GFS are tuned using a 3-fold cross validation strategy by searching a candidate set of [10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}]. The number of nearest neighbours in reliefF is tuned using the same cross validation strategy by searching a candidate set of [1, 3, 5, 7, 9]. The parameters of OEC follow its original paper (Bonyadi and Reutens, 2019): its population size is set to $(4+3\times\log(n))$ and its maximum number of iterations is set to $150\times\log(n+1)$, where n is the number of features.

5 Results based on the synthetic data

5.1 Dataset description

To investigate whether the proposed dual-component vector can effectively remove redundant features, we compare DEEFS with GFS —a state-of-the-art sparsity-based feature selection approach— on a synthetic dataset generated in a similar way as GFS's original paper (Peng and Fan, 2017b). Firstly, we generate m samples with features $\boldsymbol{A} \in \mathbb{R}^{m\times n_1}$ according to normal distribution $\mathcal{N}(0,1)$. Secondly, we introduce redundant features $\boldsymbol{B} \in \mathbb{R}^{m\times n_1}$ by adding noise $\boldsymbol{\epsilon}$ to $\boldsymbol{A}$ where $\boldsymbol{\epsilon}$ follows normal distribution $\mathcal{N}(0, 0.01)$. We then generate irrelevant features $\boldsymbol{C} \in \mathbb{R}^{m\times n_2}$ according to uniform distribution $\mathcal{U}(-1,1)$. Thus, we obtain samples with features $\boldsymbol{X} = [\boldsymbol{A};\boldsymbol{B};\boldsymbol{C}] \in \mathbb{R}^{m\times(2n_1+n_2)}$. After that, we generate the label y_i for each sample $\boldsymbol{X_i}$ by

$$y_i = \begin{cases} 1 & \text{if } \boldsymbol{X_i}\boldsymbol{w}^T > 0 \\ -1 & \text{otherwise} \end{cases} \tag{14}$$

where $\boldsymbol{w} = [\boldsymbol{w}_A; \boldsymbol{0}; \boldsymbol{0}] \in \mathbb{R}^{2n_1+n_2}$ and

$$w_{Aj} = \begin{cases} 0.9 & \text{if } j = 0 \\ -0.9 & \text{if } j = 1 \\ 0.1\times(-1)^j & \text{if } 2 \le j \le n_1 \end{cases} \tag{15}$$

We intentionally set large weights for the first two features, and relatively smaller weights for the following $(n_1 - 2)$ features to examine the ability to select weakly relevant but complementary features. Setting $m = 600, n_1 = 15,$ and $n_2 = 70$, we obtain a dataset with 600 instances and 100 features. The generated synthetic dataset has three groups of features:

- The *first group* contains 15 relevant features, ranging from 0 to 14,
- The *second group* contains 15 relevant but redundant feature, ranging from 14 to 29. Note that the features from second group are less relevant than features from the first group due to additional noise.
- The *third group* contains 70 irrelevant features.

Thus, the first 15 features form the optimal feature subset. The dataset is split into 70% as the training set and 30% as the test set.

Table 2: Comparisons between DEEFS and GFS on the synthetic data. The average SVM accuracy of each algorithm is beside the algorithm's name. The three following rows show the features selected by each algorithm. The first row is the actual selected features. The second row converts features from the second group to the features from the first group.

GFS (86.99%)
{0, 26, 11, 1, 23, 27, 12, 15, 9, 17, 16, 5, 8, 24, 29}
↓
{0, 11, 11, 1, 8, 12, 12, 0, 9, 2, 1, 5, 8, 9, 14}
DEEFS (94.67%)
{0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 13, 15, 16, 17, 19}
↓
{0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 13, 0, 1, 2, 4}

5.2 Results

The results on the synthetic data are shown in Table 2. As can be seen from the table, DEEFS achieves better classification performance than GFS. However, on the synthetic dataset, we are more interested in analysing which features are selected. For each dataset, the first row shows the actual features selected by each algorithm. Note that the features from the second group can be used interchangeably with the features from the first group since they provide almost the same information about the class label. Thus, in the second line, we convert the second group's features to the corresponding ones from the first group to show redundancy in the selected feature subsets. The redundant features are underlined.

Among the top 15 features, GFS selects six redundant features, while DEEFS selects only three redundant features. In addition, GFS ignores more than half of weakly relevant features from feature 3 to the feature 14, even though they complement feature 0 and feature 1. In contrast, DEEFS still selects these weakly relevant but complementary features. Furthermore, DEEFS selects most features from the first group, while GFS selects more than half features from the second group which is less relevant than the first group. As a result, DEEFS achieves about 8% higher accuracy than GFS.

The results on the synthetic dataset show that DEEFS can consider the interaction between features, and thus avoiding selecting redundant and irrelevant features. In addition, DEEFS can detect weakly relevant features that complement the strongly relevant ones to boost the classification performance.

6 Results based on the real-world datasets

In this section, DEEFS is examined on 14 real-world datasets. We first compare DEEFS with using all features. We then further compare DEEFS with well-known and state-of-the-art wrapper, filter, and embedded feature selection approaches.

6.1 Comparisons with using all features

The comparisons between DEEFS and using all features can be seen in Table 3, where "Full" indicates the results of using all features. Wilcoxon signed rank test with a significance level of 0.05 is used for the comparisons. In the table, $\uparrow$ / $\downarrow$ /$\circ$ indicate that DEEFS is significantly better/worse or similar to using all features.

In terms of the SVM classification algorithm, DEEFS can select features which achieve similar or even significantly better accuracy than using all features on 11 out of the 14 datasets. The most significant improvement is on the LSVT and CNS

Table 3: Comparisons with using all features (Wilcoxon rank-sum test).

Dataset	SVM Acc		KNN Acc		#Features	
	Full	DEEFS	Full	DEEFS	Full	DEEFS
Parkinson	77.60 ○	77.50	88.75 ↓	87.44	22	11.07
German	67.14 ↑	67.77	61.57 ↓	61.16	24	22.35
WBCD	95.95 ↑	96.05	95.68 ↓	94.92	30	15.53
Sonar	78.67 ↑	80.80	82.65 ↑	83.56	60	38.61
Musk1	86.09 ↓	84.48	85.05 ↓	82.59	166	73.59
LSVT	76.08 ↑	79.16	78.06 ↑	80.87	310	37.61
Madelon	55.19 ○	55.09	55.62 ○	55.75	500	465.65
Colon	78.50 ○	78.37	79.50 ↓	77.64	2000	25.27
DLBCL	95.76 ↓	92.47	86.44 ↑	89.93	5469	26.65
ALLAML	98.00 ↓	96.47	78.89 ↑	96.13	7129	25.17
CNS	53.46 ↑	56.42	61.71 ↓	57.35	7129	39.40
Leukemia	98.00 ○	98.16	77.78 ↑	97.56	7129	25.14
Prostate	90.27 ○	90.17	83.18 ↑	90.72	10509	36.53
Ovarian	100.00 ○	100.00	90.74 ↑	99.88	15154	20.99
W/D/L	5/6/3	-	7/1/6	-	-	-

datasets, where DEEFS can improve 3% accuracy over using all features. On most of the first 7 UCI datasets (Parkinson to Madelon) with small and medium numbers of features, DEEFS can significantly improve the classification performance while selecting around 50% features, except for the German and Madelon datasets. On gene-expression datasets, DEEFS mostly maintains the classification accuracy which can be considered good results since SVM usually achieves very high classification performance on such datasets. The most notable pattern is that DEEFS significantly reduces the number of features on these datasets. For example, on Ovarian, DEEFS selects only 21 features from the 15154 original features, which means DEEFS reduces 99.9% features while still achieving 100% accuracy as using all features.

To examine the generalisation of DEEFS, we evaluate the performance of DEEFS using the KNN classification algorithm. Although on most of the first 7 UCI datasets, DEEFS does not significantly improve the classification performance over using all features, the largest difference between them is only around 2%. In contrast, on most of the 7 gene-expression datasets, DEEFS achieves significantly better classification performance. The improvement can be up to 20% as on the Leukemia dataset. It has been known that the performance of KNN deteriorates when there is a large number of features. Thus, the significance of DEEFS is more visible on gene-expression datasets. The results show that although DEEFS is an embedded feature selection approach based on SVM, its selected features are likely to maintain or improve the classification of KNN, which shows that DEEFS has a good generalisation ability.

In general, the experimental results show that DEEFS can successfully reduce at least 50% of the features while mostly maintaining or even improving the classification performance over using all features. It should be noted that on the very high-dimensional datasets such as gene-expression datasets, DEEFS is able to reduce from 90% to 99.9% features, which results in a very small number of relevant features being selected while achieving high classification accuracy.

Table 4: Comparisons with wrappers in terms of **accuracies** (Friedman test).

Dataset	SVM Acc				KNN Acc			
	$W_{0.9}$	$W_{0.98}$	$W_{1.0}$	DEEFS	$W_{0.9}$	$W_{0.98}$	$W_{1.0}$	DEEFS
Parkinson	77.39	**80.85**	79.18	77.50	77.56	79.84	79.23	**87.44**
German	61.91	65.98	65.75	**67.77**	60.93	**62.44**	61.46	61.16
WBCD	95.47	95.94	**96.61**	96.05	94.64	95.01	**95.10**	94.92
Sonar	71.73	72.64	72.17	**80.80**	77.74	78.67	78.65	**83.56**
Musk1	80.46	81.18	81.48	**84.48**	82.91	83.49	**83.65**	82.59
LSVT	79.08	**80.06**	79.20	79.16	77.05	77.32	77.35	**80.87**
Madelon	**55.99**	55.55	55.37	55.09	55.39	55.27	55.23	**55.75**
Colon	76.40	74.99	76.17	**78.37**	73.09	72.27	75.22	**77.64**
DLBCL	89.72	91.02	**94.14**	92.47	77.94	77.55	83.97	**89.93**
ALLAML	92.77	92.67	**97.77**	96.47	80.30	80.00	79.60	**96.13**
CNS	**59.61**	58.43	58.85	56.42	**62.54**	61.64	62.05	57.35
Leukemia	91.48	91.48	97.67	**98.16**	79.77	79.77	78.83	**97.56**
Prostate	89.89	90.22	**91.08**	90.17	84.15	84.12	84.09	**90.72**
Ovarian	99.78	99.80	**100.0**	**100.0**	90.85	91.26	91.11	**99.88**
Ranking	3.00	2.64	2.20	**2.10**	2.97	2.68	2.61	**1.74**
W/D/L	9/2/3	9/3/2	5/6/3	-	11/1/2	9/3/2	9/2/3	-

6.2 Comparisons with wrapper approaches

In this subsection, we compare DEEFS with three wrapper algorithms which use SVM as the wrapped classifier and JADE as the search mechanism. In this work, we use three different values of β in Eq. (13): 0.9, 0.98, and 1.0, which forms three wrapper algorithms, named $W_{0.9}$, $W_{0.98}$, and $W_{1.0}$. The three values are selected so we can have more general comparisons where the wrapper algorithms pay some attention to the number of selected features (β = 0.9) or focus only on the classification performance (β = 1.0). The classification accuracies and numbers of selected features are shown in Tables 4 and 5, respectively. The best classification performance for each classification algorithm is marked in bold. A Friedman test is used to perform a statistical comparison between the four algorithms. The ranking of each algorithm is showed in the "Ranking" row of Table 4. The algorithms are then pair-wise compared by a Holm post-hoc test with a significance level of 0.05. "W/D/L" in Table 4 shows how many times DEEFS is significantly better/similar/worse than the benchmark algorithms.

In terms of the SVM classification algorithm, among the three wrapper algorithms, $W_{1.0}$ seems to achieve the highest classification accuracy since it focuses only on improving the classification performance. In comparison with $W_{1.0}$, DEEFS is significantly better on 4 datasets while achieving the same performance on 6 other datasets. DEEFS has a slightly higher rank than $W_{1.0}$ where DEEFS achieves the best classification performance on 6 datasets. These results indicate that although DEEFS is an embedded approach, it still can achieve comparative or even better results in comparison with wrapper approaches.

The results of the KNN classification algorithm show how generalised the selected features are. It can be seen that DEEFS completely dominates all other wrapper algorithms by achieving the best classification performance on 10 out of the 14 datasets. Overall, DEEFS is ranked as the top algorithm when KNN is used. The results suggest that DEEFS is more generalised than the three wrapper approaches.

In terms of the number of selected features, DEEFS tends to select larger numbers

Table 5: Comparisons with wrappers in terms of **numbers selected features**.

Dataset	$W_{0.9}$	$W_{0.98}$	$W_{1.0}$	DEEFS
Parkinson	3.17	4.20	5.54	11.07
German	6.78	12.77	14.21	22.35
WBCD	4.83	7.06	11.73	15.53
Sonar	8.36	10.43	11.53	38.61
Musk1	32.51	42.32	44.24	73.59
LSVT	45.27	53.43	74.49	37.61
Madelon	55.72	89.13	97.66	465.65
Colon	106.28	109.14	520.57	25.27
DLBCL	146.32	146.85	1829.92	26.65
ALLAML	187.58	187.69	2514.35	25.17
CNS	826.62	843.74	1357.03	39.40
Leukemia	164.39	164.39	2581.52	25.14
Prostate	509.28	511.70	2890.29	36.53
Ovarian	381.18	387.85	5558.99	20.99

of features than the three wrapper algorithms on the 7 UCI datasets to achieve better classification performance. It seems that DEEFS focuses more on the classification performance on the UCI datasets. In contrast, on the genes expression datasets, DEEFS significantly reduces more features than the wrapper approaches. Among the three wrapper algorithms, $W_{0.9}$ selects the smallest number of features. However, DEEFS selects up to 20 times smaller feature subsets than $W_{0.9}$ while achieving similar or better classification performance. The results suggest that DEEFS has a better trade-off between the number of selected features and the classification performance. The main reason is due to its initialisation which will be further analysed later.

In general, in comparison with wrapper approaches, DEEFS can achieve comparative SVM classification performance with a better trade-off between the feature subset size and the classification performance. As an embedded approach, DEEFS also selects more generalised features.

6.3 Comparisons with embedded approaches

In this subsection, we compare the performance of three benchmark embedded feature selection algorithms: RFS (Nie et al., 2010), GFS (Peng and Fan, 2017b), and OEC (Bonyadi and Reutens, 2019) with DEEFS. Note that the three algorithms cannot automatically determine the number of features. Thus, to ensure a relatively fair comparison, we use the number of features selected by DEEFS as the predefined number of selected features for the three benchmark algorithms. The classification performance of the four algorithms is shown in Table 6. On the Leukemia, Prostate, and Ovarian datasets, since RFS and GFS have not returned results after 3 weeks (including the process of tuning parameters), we are not able to report their accuracies.

GFS extends RFS by using different norms for the loss function and the regularisation function which are able to avoid outliners and further boost the model sparsity. Thus, GFS usually achieves better performance than RFS. However, both GFS and RFS are based on a gradient to search for the weights, which makes them easily to be stuck at local optima. In contrast, OEC utilises covariance matrix adaptation evolution strategy (CMA-ES) (Hansen et al., 2003) to evolve the weights. As a population-based optimisation approach, CMA-ES may assist OEC to avoid local optima, which results in a higher rank of OEC over GFS and RFS. However, OEC selects features by picking the

Table 6: Comparisons with embedded approaches (Friedman test).

Dataset	SVM Acc				KNN Acc			
	RFS	GFS	OEC	DEEFS	RFS	GFS	OEC	DEEFS
Parkinson	76.25	76.14	77.29	**77.50**	83.08	84.42	84.92	**87.44**
German	67.31	65.55	67.14	**67.77**	60.93	**61.79**	61.47	61.16
WBCD	94.80	95.64	**96.34**	96.05	94.73	**95.07**	95.64	94.92
Sonar	75.17	78.58	77.66	**80.80**	**84.70**	81.89	83.77	83.56
Musk1	77.96	79.44	82.83	**84.48**	82.31	81.58	**83.57**	82.59
LSVT	76.65	**85.89**	79.57	79.16	73.98	75.76	77.65	**80.87**
Madelon	54.19	**55.38**	55.08	55.09	57.08	**57.81**	57.03	55.75
Colon	75.00	70.00	70.97	**78.37**	74.25	70.00	66.58	**77.64**
DLBCL	79.92	79.02	79.34	**92.47**	74.62	78.03	66.73	**89.93**
ALLAML	81.44	89.89	75.08	**96.47**	75.44	90.89	66.87	**96.13**
CNS	**56.89**	54.39	53.63	56.42	**60.64**	42.61	53.76	57.35
Leukemia	N/A	N/A	74.52	**98.16**	N/A	N/A	67.60	**97.56**
Prostate	N/A	N/A	74.15	**90.17**	75.36	N/A	72.51	**90.72**
Ovarian	N/A	N/A	94.08	**100.0**	N/A	N/A	82.93	**99.88**
Ranking	2.46	2.18	2.03	**1.18**	2.20	2.10	2.20	**1.56**
W/D/L	10/0/1	9/0/2	8/2/1	-	8/0/3	8/0/3	6/1/4	-

top-ranked features with higher weights. Such a selection mechanism is at risk of selecting redundant features and ignoring complementary features. On the contrary, DEEFS considers not only the weights of features but also the interaction between features. Thus, DEEFS achieves the best classification performance on 10 out of the 14 datasets, which leads to its highest rank (1.18). In terms of the KNN classification algorithm, DEEFS also achieves the best classification performance on 8 out of the 14 datasets. The interesting pattern is on most datasets where DEEFS achieves the best SVM performance, it also achieves the best KNN performance, which shows that DEEFS selects more generalisable features than the three benchmark algorithms.

In summary, the experimental results show that DEEFS achieves better classification performance than the three benchmark embedded approaches due to its potential global search ability from JADE and its consideration of feature interactions.

6.4 Comparisons with filter approaches

We also compare DEEFS with three well-known filter approaches: CFS (Hall and Smith, 1999), mRMR (Peng et al., 2005), and reliefF (Robnik-Šikonja and Kononenko, 2003), which are representatives for correlation-based feature selection, information-based feature selection, and distance-based feature selection, respectively. The classification performance and the Friedman comparisons are shown in Tables 7. As can be seen from the table, DEEFS achieves significantly better SVM accuracies than the three filter approaches on at least 11 out of the 14 datasets. Such domination is because DEEFS is based on SVM. To examine the generalisation of DEEFS, we compare the four algorithms using the KNN classification algorithm. Table 7 shows that DEEFS achieves the best KNN performance on 9 out of the 14 datasets. Especially, on all the 7 gene-expression datasets, DEEFS achieves significantly better KNN accuracy than the three filter approaches, where DEEFS's accuracy is at least 10% higher than that of the filter approaches.

In summary, the results show that DEEFS generates even more generalisable feature subsets than the three filter approaches. The generated feature subsets have a good

Table 7: Comparisons with filter approaches (Friedman test).

Dataset	SVM Acc				KNN Acc			
	CFS	mRMR	reliefF	DEEFS	CFS	mRMR	reliefF	DEEFS
Parkinson	74.24	**77.92**	74.79	77.50	84.48	82.30	**87.74**	87.44
German	66.98	66.60	67.10	**67.77**	61.67	**61.81**	60.57	61.16
WBCD	95.06	**96.09**	94.44	96.05	94.09	94.82	92.58	**94.92**
Sonar	76.09	75.77	82.42	**80.80**	**84.56**	84.18	82.69	83.56
Musk1	80.83	82.96	82.30	**84.48**	**83.19**	80.00	83.08	82.59
LSVT	70.16	57.69	64.99	**79.16**	64.62	64.44	64.48	**80.87**
Madelon	**55.38**	54.54	55.23	55.09	**58.15**	57.00	56.85	55.75
Colon	**86.00**	75.50	70.75	78.37	64.25	67.25	69.25	**77.64**
DLBCL	80.68	77.20	91.67	**92.47**	62.27	59.02	76.67	**89.93**
ALLAML	68.33	80.33	77.56	**96.47**	61.78	60.67	77.78	**96.13**
CNS	**64.21**	44.79	51.64	56.42	54.29	50.07	46.46	**57.35**
Leukemia	72.44	73.22	80.67	**98.16**	62.67	62.78	82.67	**97.56**
Prostate	67.64	72.55	75.36	**90.17**	67.64	68.45	80.45	**90.72**
Ovarian	95.52	82.58	85.34	**100.0**	84.50	78.17	73.90	**99.88**
Ranking	2.80	3.10	2.63	**1.47**	2.54	2.00	2.80	**1.65**
W/D/L	11/0/3	12/0/2	11/1/2	-	10/0/4	11/0/3	11/1/2	-

performance on both SVM and KNN classification algorithms.

6.5 Computation time

Table 8 shows the computation times of DEEFS and the benchmark algorithms. Note that $W_{0.9}$ is used as a representative of the three wrapper approaches since it selects the smallest number of features which results in its lowest computation time. As can be seen from the table, DEEFS is usually about 50 to 100 times faster than $W_{0.9}$. The difference is due to the fact that an SVM classifier is built in each individual of DEEFS, thus DEEFS does not need to train any SVM classifier. In contrast, $W_{0.9}$ needs to train an SVM classifier to evaluate each individual, thus $W_{0.9}$ is much more time-consuming than DEEFS.

In comparison with the three embedded approaches, DEEFS is always faster than at least one of the embedded approach. Especially, on the 7 gene-expression with thousands of features, DEEFS is always the most efficient one. The main reason is that RFS and GFS involve many matrix operations, which are computationally intensive when there is a large number of features. OEC does not involve any matrix operation, thus it can output feature subsets on the last three gene-expression datasets (Leukemia, Prostate, and Ovarian), which cannot be done by RFS and GFS within 3 weeks. However, OEC utilised CMA-ES to optimise the weight vector. CMA-ES learns dependencies between the variables, which results in its $O(n^2)$ computation complexity where n is the number of decision variables (features) (Loshchilov, 2014). Thus, OEC does not scale well with datasets having large numbers of features. In contrast, DEEFS only requires basic vector operators to optimise the weights. Therefore, DEEFS can be up to 600 times faster than OEC.

As expected, the three filter approaches are very efficient where reliefF is the most efficient one. Since CFS and mRMR need to compute the pair-wise interaction between features, they do not scale with the number of features as well as reliefF which computes a weight for each individual feature. In comparison with the three filter approaches, DEEFS has a comparative computation time. Particularly, DEEFS is usually

Table 8: Computation time (in seconds).

Dataset	Filter			Embedded			$W_{0.9}$	DEEFS
	CFS	mRMR	reliefF	RFS	GFS	OEC		
Parkinson	0.96	0.11	0.03	0.20	0.03	0.39	87.62	1.45
German	4.19	0.49	0.29	84.41	0.30	0.68	468.97	2.36
WBCD	2.34	0.57	0.18	9.82	0.12	0.54	100.96	2.40
Sonar	3.21	1.01	0.02	0.47	0.07	1.16	126.82	2.36
Musk1	38.24	11.35	0.11	10.06	1.26	8.01	904.33	7.50
LSVT	68.84	5.00	0.02	0.62	1.57	17.09	467.45	5.45
Madelon	217.11	703.36	3.06	2866.17	55.23	93.51	8450.27	117.31
Colon	102.89	10.70	0.25	42.05	269.41	1645.73	939.71	32.26
DLBCL	223.16	41.46	2.12	812.96	5696.37	14593.10	643.24	68.43
ALLAML	163.61	42.47	3.12	1903.34	12647.75	26589.32	569.70	85.03
CNS	321.94	71.75	3.11	1499.07	13194.40	25675.51	1439.78	151.03
Leukemia	196.66	54.01	3.14	N/A	N/A	26627.74	545.19	88.04
Prostate	365.11	105.45	6.79	N/A	N/A	54419.07	5003.51	166.47
Ovarian	4935.66	210.02	14.30	N/A	N/A	125653.83	4867.88	210.81

faster than CFS. Especially, on the gene-expression datasets, DEEFS can be up to 20 times faster. DEEFS is also at most 2 times slower than mRMR.

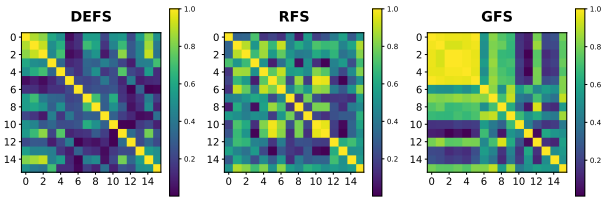
In summary, DEEFS has been shown to be an efficient feature selection approach, which scales well with the number of features and still selects feature subsets with high classification accuracies.

7 Further discussions

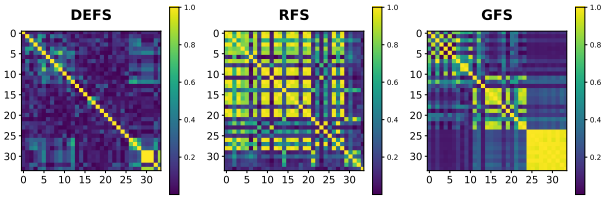
7.1 An Analysis of Selected Features

An advantage of DEEFS is the ability to consider feature interactions which helps DEEFS to avoid selecting redundant features. We further examine the claim by evaluating the correlation between features selected by DEEFS, RFS, and GFS. The correlation between features is calculated by Pearson's correlation which has been widely applied to feature selection (Pan et al., 2021).

The comparisons between DEEFS, RFS, and GFS are showed in Fig. 2. In the figure, each heatmap represents a correlation matrix of the selected features, i.e. the element at the ith row and jth column is the correlation between the ith selected feature and the jth selected feature. The brighter heatmap cells indicate stronger correlations between selected features, i.e. more redundancy between the selected features. As can be seen from the figure, RFS and GFS usually have brighter heatmaps than DEEFS, which illustrates that both RFS and GFS tend to select redundant features. A typical pattern of RFS and GFS is that their neighbouring selected features are very likely to have a strong correlation, which can be seen on the top-left region of GFS on WBCD, the bottom-right region of GFS on LSVT, and the middle region of RFS on LSVT (all regions are highly bright). The main reason is that RFS and GFS rank features based on the weight values of the features, and then top-ranked features are selected. Thus, the neighbouring selected features usually have similar weight values and provide the same information, which causes significant redundancy between them. The analysis of the selected features indicates that DEEFS can determine the number of selected features and avoid redundant features, which cannot be achieved by RFS and GFS.



(a) WBCD



(b) LSVT

Figure 2: Correlation between the features selected by DEEFS, RFS, and GFS. The brighter the figure, the more strongly correlated (redundant) features.

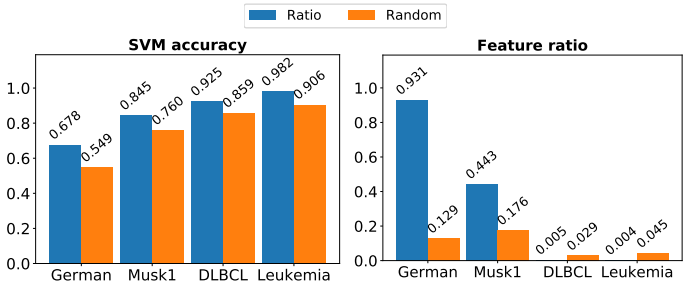


Figure 3: Comparison between ratio and random initialisations.

7.2 Effect of ratio initialisation

Instead of using a random initialisation, DEEFS uses a ratio initialisation to have a better starting point. This subsection analyses the effect of the ratio initialisation. The analysis is shown on four datasets: German, Musk1, DLBCL, and Leukemia, which are representatives for UCI and gene-expression datasets. The patterns are similar on the other datasets. For presentation convenience, we denote DEEFS with ratio and random initialisations as DEEFS$_{ratio}$ and DEEFS$_{rnd}$, respectively.

Fig. 3 shows a further comparison between DEEFS$_{ratio}$ and DEEFS$_{rnd}$ in terms of the SVM accuracy and the selected feature ration. It can be seen that DEEFS$_{ratio}$ achieves significantly better SVM accuracies on all the four datasets. The accuracy differences range from 7% to 13%. In terms of the feature ratio, DEEFS$_{ratio}$ selects a larger number of features on the UCI datasets. However, on the gene-expression datasets, DEEFS$_{ratio}$ selects a smaller number of features. The main reason is the number of instances. Since UCI datasets have a large number of instances, DEEFS$_{ratio}$ tends to select more features to help SVM separate the instances. In contrast, gene-expression datasets have a small number of instances which are easier to be separated, thus DEEFS$_{ratio}$ selects only a small number of features (around 0.5%). It can be con-

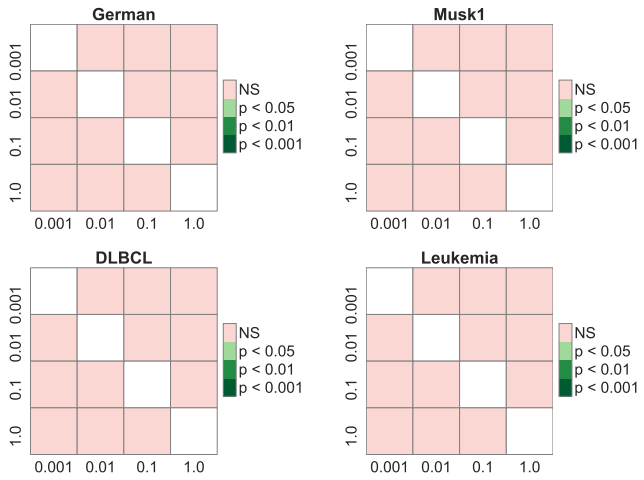


Figure 4: Friedman-Nemenyi test for different α values in terms of **accuracy**. "NS" means there is no significant difference (best view in colour).

cluded that DEEFS$_{ratio}$ is able to adapt to each dataset to achieve better selection performance. Such characteristic is a result of the ratio initialisation which allows DEEFS to consider various numbers of features ranging from 1 to all features. In contrast, the random initialisation uses the threshold 0.6 to initialise, thus on average DEEFS$_{rnd}$ starts with 60% of features. In addition, DEEFS$_{ratio}$ also utilises SVM to initialise its weight vectors, which ensures that the built-in SVM classifier is good enough to illustrate the goodness of the selected features. Thus, the ratio initialisation allows JADE to focus on promising regions in the large search space of the proposed representation.

In summary, the results show that the ratio initialisation can assist DEEFS to achieve better performance by considering more various numbers of selected features and ensuring that the weight vectors suit the corresponding selected features.

7.3 Effect of the regularisation parameter α

To examine the effect of α in Eq. (12), DEEFS is run with 4 different α values: 0.001, 0.01, 0.1, and 1.0. The obtained accuracies are compared using the Friedman test with Nemenyi post-hoc analysis. The significance level is set to 0.05. Fig. 4 shows the comparisons between the results of the 4 different α values on 4 datasets: German, Musk1, DLBCL, and Leukemia. Similar patterns are obtained on the other datasets. As can be seen from the figure, the SVM accuracies obtained by the four α values are not significantly different. Thus, the accuracy of DEEFS is not sensitive to the value of α, which suggests that DEEFS can be applied to a wide range of applications.

We also compare the number of selected features obtained by different α values. The results can be seen in Fig. 5. On most cases, the numbers of selected features are not significantly different across different α values. However, setting α to 0.001 results in much larger numbers of selected features than setting α to larger values such as 0.01, 0.1, and 1.0. Thus, although the number of selected features is not too sensitive to α, it is not recommended to set α smaller than 0.01.

To further illustrate the importance of the regularisation part in the fitness function, we compare the results of DEEFS when α is set to 0.01 and 0.0, where the value of 0.0

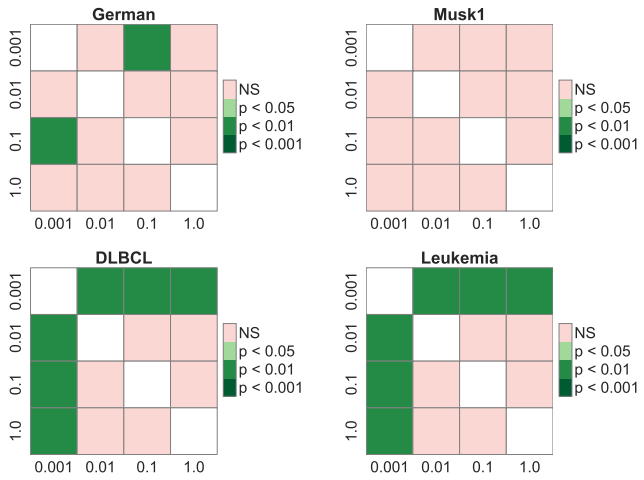


Figure 5: Friedman-Nemenyi test for different α values in terms of **the number of selected features**.

means that we remove the regularisation part in the fitness function. The comparison is showed in Table 9. From the table, we can see that the value of 0.01 is slightly better than the value of 0.0 in terms of the SVM and KNN classification accuracies. However, in terms of the number of selected features, setting α to 0.01 can select up to 3 times smaller numbers of features than setting α to 0.0, which can be seen clearly on the last seven gene expression datasets. Based on the results, we can conclude that the regularisation part plays a significant role in DEEFS, which can not only improve the classification performance but also significantly reduce the number of features.

8 Conclusions and future work

The goal of this study was to develop a sparsity regularisation-based feature selection that considered the feature interactions and could determine the number of selected features. The goal was achieved by proposing a hybrid representation to simultaneously build weight vectors and decide which features to be selected. Experimental results on a synthetic dataset showed that the proposed algorithm, called DEEFS, could select complementary features with different weight values while existing sparsity regularisation-based feature selection algorithms tended to select redundant features. DEEFS was also compared with well-known and state-of-the-art wrapper, filter, and embedded approaches. The results showed that DEEFS achieved significantly better classification performance than the filter and embedded approaches. In comparison with wrapper approaches, DEEFS mostly achieved better classification performance or at least comparative performance and used significantly smaller computation times. The feature subsets selected by DEEFS were also more generalisable than that selected by other benchmark algorithms.

Although DEEFS is an efficient and effective feature selection algorithm, it is currently only applicable for binary classification. The main issue is a large number of weights to be optimised for multi-class classification which is difficult for EC to adapt. In the future, we will investigate extending DEEFS for multi-class classification. Besides, the proposed representation essentially enlarges the search space. We will also

Table 9: Comparison between $\alpha = 0.01$ and $\alpha = 0.0$.

Dataset	SVM Acc		KNN Acc		Number of features	
	$\alpha = 0$	$\alpha = 0.01$	$\alpha = 0$	$\alpha = 0.01$	$\alpha = 0$	$\alpha = 0.01$
Parkinson	77.55 ↑	77.50	87.11 ○	87.44	11.07	11.07
German	67.72 ○	67.77	60.69 ↑	61.16	22.04	22.35
WBCD	96.07 ○	96.05	94.74 ○	94.92	15.31	15.53
Sonar	81.02 ○	80.80	83.97 ○	83.56	38.89	38.61
Musk1	84.48 ○	84.48	82.38 ○	82.59	73.27	73.59
LSVT	79.13 ○	79.16	81.03 ○	80.87	39.23	37.61
Madelon	55.09 ○	55.09	55.72 ○	55.75	465.30	465.65
Colon	77.03 ↑	78.37	79.43 ↓	77.64	33.91	25.27
DLBCL	92.89 ○	92.47	87.81 ↑	89.93	64.13	26.65
ALLAML	97.26 ↓	96.47	96.06 ○	96.13	73.21	25.17
CNS	57.72 ↓	56.42	59.28 ○	57.35	52.20	39.40
Leukemia	97.17 ↑	98.16	96.17 ↑	97.56	76.40	25.14
Prostate	90.46 ○	90.17	90.10 ○	90.72	57.69	36.53
Ovarian	100.00 ○	100.00	99.68 ↑	99.88	53.65	20.99

investigate a more comprehensive representation which encodes the same information but using a smaller number of bits. Another potential future direction is to use non-linear kernels in DEEFS to consider more complex feature to label interactions.

References

Balın, M. F., Abid, A., and Zou, J. (2019). Concrete autoencoders: Differentiable feature selection and reconstruction. In *International conference on machine learning*, pages 444–453. PMLR.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Science+ Business Media.

Bonyadi, M. R. and Reutens, D. C. (2019). Optimal-margin evolutionary classifier. *IEEE Transactions on Evolutionary Computation*, 23(5):885–898.

Chen, G. and Chen, J. (2015). A novel wrapper method for feature selection and its applications. *Neurocomputing*, 159:219–226.

Cheng, S., Liu, B., Shi, Y., Jin, Y., and Li, B. (2016). Evolutionary computation and big data: key challenges and future directions. In *International Conference on Data Mining and Big Data*, pages 3–14. Springer.

Das, S. and Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31.

Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156.

Gu, S., Cheng, R., and Jin, Y. (2018). Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing*, 22(3):811–822.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182.

Hall, M. A. and Smith, L. A. (1999). Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In *FLAIRS conference*, volume 1999, pages 235–239.

Han, F., Yang, C., Wu, Y.-Q., Zhu, J.-S., Ling, Q.-H., Song, Y.-Q., and Huang, D.-S. (2015). A gene selection method for microarray data based on binary PSO encoding gene-to-class sensitivity information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(1):85–96.

Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18.

Hara, S. and Maehara, T. (2017). Enumerate lasso solutions for feature selection. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Keogh, E. and Mueen, A. (2017). *Curse of Dimensionality*, pages 314–315. Springer US, Boston, MA.

Kim, Y. and Kim, J. (2004). Gradient LASSO for feature selection. In *Proceedings of the Twenty-First International Conference on Machine learning*, page 60. ACM.

Lemhadri, I., Ruan, F., and Tibshirani, R. (2021). Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pages 10–18. PMLR.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2018). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94.

Lichman, M. (2013). UCI machine learning repository Irvine, CA: University of California, School of Information and Computer Sciences.

Liu, J., Ji, S., and Ye, J. (2009). Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 339–348. AUAI Press.

Loshchilov, I. (2014). A computationally efficient limited memory cma-es for large scale optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 397–404. ACM.

Nguyen, B. H., Xue, B., Andreae, P., Ishibuchi, H., and Zhang, M. (2019). Multiple reference points based decomposition for multi-objective feature selection in classification: Static and dynamic mechanisms. *IEEE Transactions on Evolutionary Computation*.

Nie, F., Huang, H., Cai, X., and Ding, C. H. (2010). Efficient and robust feature selection via joint l2, 1-norms minimization. In *Advances in Neural Information Processing Systems*, pages 1813–1821.

Oehmcke, S. and Gieseke, F. (2022). Input selection for bandwidth-limited neural network inference. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 280–288. SIAM.

Pan, H., You, X., Liu, S., and Zhang, D. (2021). Pearson correlation coefficient-based pheromone refactoring mechanism for multi-colony ant colony optimization. *Applied Intelligence*, 51(2):752–774.

Peng, H. and Fan, Y. (2016). Direct sparsity optimization based feature selection for multi-class classification. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1918–1924.

Peng, H. and Fan, Y. (2017a). A general framework for sparsity regularized feature selection via iteratively reweighted least square minimization. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Peng, H. and Fan, Y. (2017b). A general framework for sparsity regularized feature selection via iteratively reweighted least square minimization. In *Thirty-First AAAI Conference on Artificial Intelligence*.

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

Robnik-Šikonja, M. and Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53(1):23–69.

Shukla, A. K., Singh, P., and Vardhan, M. (2018). A two-stage gene selection method for biomarker discovery from microarray data for cancer classification. *Chemometrics and Intelligent Laboratory Systems*, 183:47–58.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Tang, J., Alelyani, S., and Liu, H. (2014). Feature selection for classification: A review. *Data classification: Algorithms and applications*, page 37.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tran, B., Xue, B., and Zhang, M. (2018). Variable-length particle swarm optimization for feature selection on high-dimensional classification. *IEEE Transactions on Evolutionary Computation*, 23(3):473–487.

Tuba, E., Strumberger, I., Bacanin, N., Jovanovic, R., and Tuba, M. (2019). Bare bones fireworks algorithm for feature selection and svm optimization. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2207–2214.

Wang, A., An, N., Chen, G., Li, L., and Alterovitz, G. (2015). Accelerating wrapper-based feature selection with k-nearest-neighbor. *Knowledge-Based Systems*, 83:81–91.

Wei, J., Zhang, R., Yu, Z., Hu, R., Tang, J., Gui, C., and Yuan, Y. (2017). A bpso-svm algorithm based on memory renewal and enhanced mutation mechanisms for feature selection. *Applied Soft Computing*, 58:176–192.

Wei, X., Cao, B., and Yu, P. S. (2016). Nonlinear joint unsupervised feature selection. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 414–422.

Xiang, S., Nie, F., Meng, G., Pan, C., and Zhang, C. (2012). Discriminative least squares regression for multiclass classification and feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 23(11):1738–1754.

Xu, Z., Huang, G., Weinberger, K. Q., and Zheng, A. X. (2014). Gradient boosted feature selection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 522–531.

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

Yan, H. and Yang, J. (2015). Sparse discriminative feature selection. *Pattern Recognition*, 48(5):1827–1835.

Zhang, J. and Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958.

Zhang, M., Ding, C., Zhang, Y., and Nie, F. (2014a). Feature selection at the discrete limit. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Zhang, Y., Wang, S., Phillips, P., and Ji, G. (2014b). Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. *Knowledge-Based Systems*, 64:22–31.

Zhao, H., Sinha, A. P., and Ge, W. (2009). Effects of feature construction on classification performance: An empirical study in bank failure prediction. *Expert Systems with Applications*, 36(2):2633–2644.