Knowledge-Based Systems 204 (2020) 106202

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Supervised feature selection through Deep Neural Networks with pairwise connected structure

Yingkun Huang *, Weidong Jin, Zhibin Yu, Bing Li

College of Electrical Engineering, Southwest Jiao tong University, Chengdu, PC 610031, China

ARTICLE INFO

Article history:
Received 17 January 2020
Received in revised form 21 May 2020
Accepted 28 June 2020
Available online 2 July 2020

Keywords:
Feature selection
Sparse learning-based
DNN
Fully-connected network
Pairwise connected structure

ABSTRACT

Feature selection is an important data preprocessing strategy, has been proven empirically that it contributes to reducing the dimensionality of feature and enhancing the performance of learning algorithms in practice. Typical sparse learning-based models select the features by removing ones that the feature scores are zero. However, linear models puzzle to build the non-linear relations between features and responses. The Deep Neural Network (DNN) has a strong capability to mode the non-linear relations and has been employed to select features. In this paper, we introduce a novel deep Neural network-based Feature Selection (NeuralFS) method to identify features. The new model comprises of a fully-connection network, a decision network, and connect them through a pairwise connected structure. In NeuralFS, the fully-connected network is the crucial structure in NeuralFS that transforms the features into their corresponding scores, and the decision network is the final structure that performs classification or regression. The pairwise connected can be regarded as a "bridge" to connect the two networks, and its weights are fixed as the normalized input as well as it is un-trainable during model training. After optimizing, the feature scores can be obtained by calculating the output of the fully-connected network. NeuralFS takes advantage of the deep network to model the non-linearity, and also make features scores sparse without the sparse regularization technology. We apply the proposed method to both synthetic datasets and benchmark datasets to prove its effectiveness.
© 2020 Elsevier B.V. All rights reserved.

1. Introduction

In many real-world machine learning and data mining tasks, they usually take three steps to build generalization models, including feature engineering, ensemble algorithm, and parameters optimization. The key step is the feature engineering, which includes feature extraction and feature selection. Feature extraction builds a set of new features through the transformation of feature space, while feature selection chooses a feature subset by various selection criteria. Both feature extraction and feature selection can address the known *curse of dimensionality issue* [1,2], which causes the performance degradation of the learning algorithm, especially if the dimensionality of features is larger than the size of the dataset. Feature selection maintains physical meanings of original features that make models better comprehensibility and readability compared to feature extraction [1], so it is widely used in many scientific areas with high dimensional features.

Feature ranking is a major research direction of feature selection because of its simplicity, comprehensibility, low computational cost, and empirical success [2]. Its mechanism is very simple, that is, it assigns the scores for features based on artificial evaluation functions and then select the top first features with high scores to be the subset. The design of evaluation functions is the critical process for feature ranking. Many methods have been used to create the functions, such as the similarity-based [3–5], the information theoretical-based [6–8], and the statistical-based methods [9–11]. The similarity-based methods are needed to calculate the similarity matrix of data so that the time and space complexity will be very high for high-dimensional features; The information theoretical-based are only applicable to discrete data such that, the discretization is necessary for continuous data. Most statistical-based methods analyze features individually so that some redundancy features may be selected. Feature ranking have three stubborn problems should be solved: (1) The bias of selected features is often larger because they are independent of the learning algorithms; (2) How many features should be chosen, which is a difficult question to answer; (3) For classification, or regression tasks, very high features scores does not mean provide a significant performance improvement [2].

Feature subset selection is the main direction that has been taken to solve the above problems of features ranking. It contains wrapper methods and embedded methods. Wrappers rely on the performance of the learning algorithm to score subsets of feature, but they are not usually used in big data scenarios as

* Corresponding author.
E-mail addresses: hykun@live.com (Y. Huang), wdjin@home.swjtu.edu.cn (W. Jin), zbyu@swjtu.cn (Z. Yu), bl@swjtu.edu.cn (B. Li).

https://doi.org/10.1016/j.knosys.2020.106202
0950-7051/© 2020 Elsevier B.V. All rights reserved.

a result of the larger computational cost. The search strategy is critical for wrapper methods. Common methods include best-first search [12], genetic [13], and branch-and-bound search methods [14]. Also, swarm optimization-based wrapper has some research results recently. For example, hybrid particle swarm optimization algorithm-based method [15], multi-verse optimization algorithm-based method [16], enhanced chaotic multi-Swarm whale optimization-based method [17], and improved ant colony optimization-based method [18,19]. Embedded methods have the advantage of computational efficiency and the low bias of selected features. The sparse learning-based method is an important direction of embedded methods. It gets sparse feature scores, and then removing the features with zero scores and combining the features with non-zeros feature score to be a subset. To make the feature weights sparse, embedding a sparse regularization term into the learning model is a good ideal. Tibshirani first used a linear model with an l_1-regularization term to score features, the model named Least Absolute Shrinkage and Selection Operator (LASSO) [20]. Hoai et al. introduced a new sparse $l_2 - l_0$ regularization term and implement it in Support Vector Machine (SVM)-based feature selection [21]. Han et al. developed a novel optimization algorithm to solve the Structural Sparse Less Square Regression (SSLSR) with l_0-regularization [22], the relevant features are selected in the forward step of the algorithm. However, the linear models unable to work for non-linear problems. Thus, some researches integrate the ideal of deep learning and sparse regularization term to identify features [23,24], which have been applied to complex classification or regression tasks. Nevertheless, these methods use the complex sparse regularization term (such as elastic-net [25], $l_2 - l_0$ regularization, and sparse group LASSO [26]), which not only add the parameter into the model but also increase the difficulty of model optimization.

How to get the sparse feature scores based on a learning algorithm without any regularization terms? Return to feature ranking, we can find that the feature scores can be obtained by creating a nonlinear transformation among features and scores. And it is a simple but effective operation for feature selection. It seems that an inspiring way to solve the above problem of the learning model with a sparse regularization term. However, the nonlinear transformations function is designed artificially, and also, the feature scores are hard to get zero values. If we introduce a fully-connected network to build the nonlinear transformations function, the above problem will be easy to solve. A fully-connected network can be constructed by deep learning techniques. Also, we can introduce a sparse activation function to make the outputs of fully-connected network sparse.

In this paper, we integrate the idea of feature ranking into the deep model to select features. We propose a new deep model architecture named deep Neural network-based Feature Selection (NeuralFS). In contrast to the prior sparse learning-based model with regularization terms, NeuralFS get the feature scores by modeling the mapping relation between input features and their corresponding scores. The architecture of NeuralFS means that it can easy to make feature scores sparse by choosing a suitable sparse activation function. Specifically, the main contributions of our method are as follow.

1. A novel way of feature selection is introduced in this paper. It transforms the input features into their corresponding scores. Our method takes advantage of DNN to model the non-linearity between the feature and its score, therefore, it can get the correct feature scores.
2. A deep model-based feature selection model is built, and it consists of a fully-connected network, pairwise connected layer, and a decision network is built in the paper. The fully-connected is used to transform the input into corresponding scores; The decision network can identify the results and then provide the errors for training the whole network; The pairwise connected layer connects two networks.
3. The hard threshold activation function is introduced in the output layer of the fully-connected network. The threshold function has two advantages of sparsity and linearity, that is, it can make the weights of the irrelevant features to be zero, and retain the negative feature scores. The classical ReLU forces the negative feature scores to be zero, which is not reasonable in feature selection applications.

Last but not least, we design two experiments to evaluate the performance of feature selection. For the real experiment, we use 16 benchmark datasets that come from *Feature Selection Repository* [27]. The repository contains around 40 popular feature selection algorithms by Python language and many benchmark datasets. With those comparable results, we demonstrate that our method achieves a better performance of feature identification than multiple competitive feature selection methods.

2. Related work

2.1. Feature selection

Given a dataset $\mathbf{D} \in \mathbb{R}^{n\times m}$ with m dimensionality of features and n size of instances. Assuming that $S \in \mathbb{R}^p$ $(p \ll m)$ is a subset of the features in $\mathbf{D}$, and S only includes all of the features with a strong relation to response such that, based on the features in S, a new dataset $\mathbf{D}^* \in \mathbb{R}^{n\times p}$ can be extracted from$\mathbf{D}$. In particular, $\mathbf{D}^*$ keeps the larger amount of information on $\mathbf{D}$. Therefore, the feature selection aims to find S through different strategies. The sparse linear model chooses S by making the weights of irrelevant features to zero. However, the linear model is hard to deal with non-linear problems. For instance, Fig. 1 shows a typical non-linear binary classification problem. The sub-figure (a) classifies the dataset by using a linear decision plane, which is decided by a simple logistics regression. As shown in Fig. 1(a), we can see that the result is not ideal. Through piling up hidden layers, DNNs can model the non-linearity. As shown in Fig. 1(b), a simple fully-connected neural network with two hidden layers can learn a non-linear decision plane to classify the dataset.

For a supervised machine learning task with n observations $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $\boldsymbol{x}_i \in \mathbb{R}^m$ is an observation with m-dimensional features, $y_i \in \{1, \ldots, c\}$ is the label value. The goal of our method is to learn a deep model by using $(\boldsymbol{x}_i, y_i)$ so that correct sparse feature scores can be obtained. We use $\mathbf{X} = (\boldsymbol{x}_0, \ldots \boldsymbol{x}_n)^{\mathrm{T}} \in \mathbb{R}^{n\times m}$ to represent the input matrix, and $\boldsymbol{y} = (y_0, \ldots, y_n)^{\mathrm{T}} \in \mathbb{R}^n$ denotes the label vector. The feature dimensionality m usually larger than the sample size n. Let $\boldsymbol{s} = (s_0, \ldots, s_m)^{\mathrm{T}} \in \mathbb{R}^m$ be the scores of selected features, where s_i denotes the important score corresponding to $\boldsymbol{x}_i$.

2.2. Deep feature selection methods

Rapid advances in deep learning provide an elegant way to build novel embedded feature selection methods. Through stacking hidden layers, DNN is easy to model the non-linear correlations between features and responses. Thus, selected features with non-linear behaviors by deep models can be done. DNN-based feature selection has gained attention recently and falls into an active research frontier. At many top conferences that focus on machine learning or data mining issues, such as NeurIPS, ICML, and IJCNN, etc., numerous DNN-based feature selection methods with good creativity have been published. For example, Li et al. used a deep model with elastic-net regularization (DeepFS) to identify the important features of regulatory events

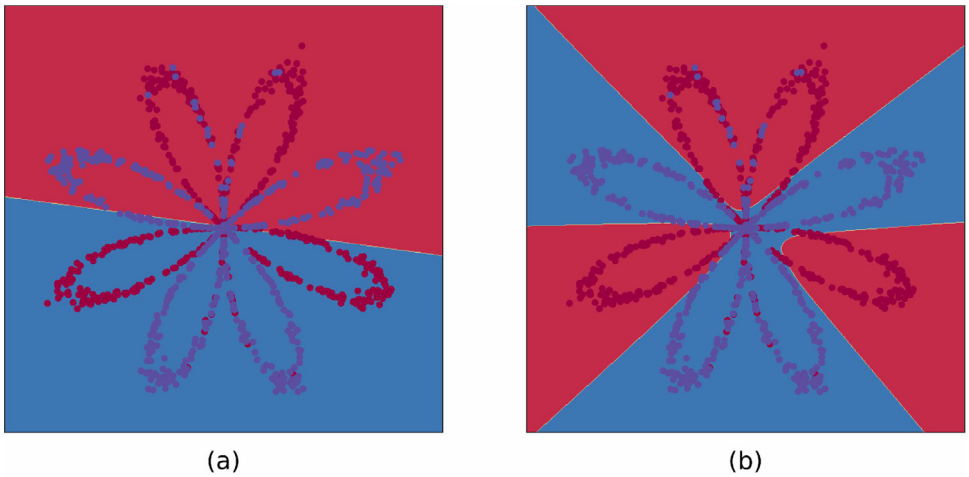


Fig. 1. A classification example for planar data.

in the human genome [23]. The main idea of the method is adding a paired-wise connected layer between the input layer and the first hidden layer, and then use elastic-net regularization to scale the weights of the paired-wise connected layer. Fig. 2 shows an example of such a model. In the input level, the weights being larger in magnitude means that the corresponding features are more important for classification or regression, and the features with zero will be discarded. The feature scores can be learned through minimizing the loss function that formulated below,

$$\begin{aligned} \boldsymbol{s} = \mathbf{w} &= \underset{\boldsymbol{w},\boldsymbol{b},\mathbf{W}}{\operatorname{argmin}} f\left(\mathbf{w}, \boldsymbol{b}, \mathbf{W}\right) \\ &= \underset{\boldsymbol{w},\boldsymbol{b},\mathbf{W}}{\operatorname{argmin}} \left(l\left(\boldsymbol{w}, \boldsymbol{b}, \mathbf{W}\right) + \alpha_1 \left(\frac{1-\alpha_2}{2} \|\boldsymbol{w}\|_2^2 + \alpha_2 \|\boldsymbol{w}\|_1 \right) \right. \\ &\left. + \beta_1 \left(\frac{1-\beta_2}{2} \sum_{k=1}^{K} \|\mathbf{W}_k\|_F^2 + \beta_2 \sum_{k=1}^{K} \|\mathbf{W}_k\|_1 \right) \right). \end{aligned} \tag{2.1}$$

Where $\mathbf{w}$ is the weight vector of the paired-wise connected layer, and $\mathbf{W}_k$ is the weight matrix connecting the $(k-1)$th layer to the kth layer, the hyper-parameter $\alpha_2 \in [0, 1]$ is set to larger greater than the α_1 to obtain sparse weights in practice. Although elastic- net on $\boldsymbol{w}$ forces features to be sparse, elastic-net has two parameters that need to optimize, meaning more opportunity for overfitting compared to the LASSO that just has a single parameter. Simone et al. explored a DNN with group sparse regularization to achieve competitive performances [28]. And also, the research focuses on group lasso regularization for convolutional neural networks and recurrent neural networks. Nevertheless, the sparse group lasso problem is also hard to solve. Yutaro et al. proposed a new procedure of feature selection based on deep model and l_0 regularization term [24]. However, the optimization of a loss with l_0 regularization term will suffer from high variance [29].

A novel method called DeepPINK was presented in NeurIPS [30]. It works well on feature selection by integrating the idea of the knockoff filter and deep learning [31]. DeepPINK adds a pairwise-connected layer with several filters in the input level, where each filter contains the original feature and its knockoff counterpart. Since the constructed counterparts are different from the original features [30], the relevant features should get higher weights than their counterpart. Contrarily, the noises should obtain similarity scores to their counterparts. It is a brilliant way to select features, but the algorithm takes additional time to produce the knockoff counterparts, and the sparsity of feature scores is poor in practical application. Also, learning

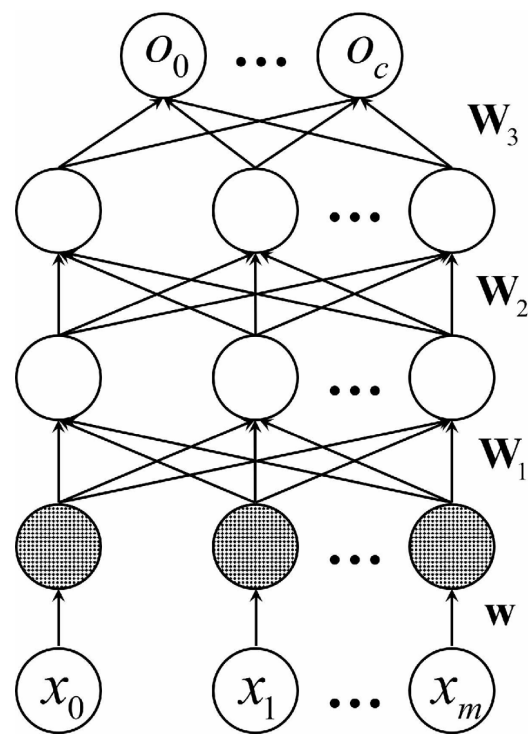


Fig. 2. A graphical illustration of DeepFS.

feature scores by analyzing the activation values of neurons have been gained attention in many scientific areas. For instance, Roy et al. proposed a deep model that computes the activation potential of the input layer to evaluate the discriminability of features, it performs well on action recognition in videos [32]. Shrikumar et al. presented a method that can compute feature scores by propagating activation differences [33]. The methods have nothing to restructure deep models such that, the performance of models is not be affected. Comparing to the analyzing activation of neurons, selecting features based on the deep model with a sparse pairwise connected layer has two advantages: (1) the meanings of the learned feature scores are easier to understand; (2) it avoids to handcraft some explicitly or implicitly feature important measures. There are other ways for feature selection based on the deep model, such as Qin et al. used a deep learning-based feature selection for remote sensing scene

classification [34]. Lei et al. presented a multi-modal deep model with sparse group lasso for feature selection [35]. The method improves the performance of image classification. These methods just regard the deep model as a good feature learning algorithm. Strictly speaking, the methods are not the deep model-based feature selection method. Also, Kabir et al. proposed a method by combining the ideal of the wrapper with a neural network [36]. In the method, the neural network is just regarded as a learning algorithm so the method is not much different from the wrapper methods.

3. Methodology

3.1. Mechanism

As we know that the feature scores extracted by filter method at the unsupervised learning scenario, indeed, can be express as $\boldsymbol{s} = \varphi(\mathbf{X})$, where $\varphi(\cdot)$ denotes a handcrafted evaluation function by using similarity or statistical methods. That is to say, there are unknown relations between feature scores and the features, and the relations are usually non-linearity. Deep learning is a good method to model the non-linear behaviors between the inputs and the outputs, and thus, trying to get the feature scores from the output layer of the deep model is a good idea. The model can be denoted as $\boldsymbol{s} = \phi(\mathbf{W}, \mathbf{X})$, where $\mathbf{W}$ is the parameter matrix of the deep model, which can be learned by deep learning technology on $(\mathbf{X}, \boldsymbol{y})$. $\phi(\cdot)$ denotes a nonlinear function that builds the mapping relations between the features and their corresponding scores.

Is the mechanism of a deep model with a sparse penalty term equivalent to that of $\boldsymbol{s} = \phi(\mathbf{W}, \mathbf{X})$. The answer is "no". It is because the feature scores can be extracted from the paired-wise connected layer for the former, but the latter obtains the scores from the output level of the fully-connected network of NeuralFS. Also, the former forces feature scores sparse based on the mechanism of regularization, while the latter uses sparse activation function to obtain sparse feature scores. One of the advantages of $\boldsymbol{s} = \phi(\mathbf{W}, \mathbf{X})$ is no add extra behaviors to make features sparse, instead, it uses the sparse activation function to force the scores of the weak features to become smaller and, in some cases, exactly zero.

3.2. Building a NeuralFS

NeuralFS consists of a fully-connected network, a decision network, and a pairwise connected layer. We use the fully-connected neural network with hidden layers to model the nonlinear function $\phi(\cdot)$. The outputs of $\phi(\cdot)$ are the feature scores corresponding to features, thus, the sizes of the output layer and the input layer of $\phi(\cdot)$ should be fixed to equally, that is, the structure of $\phi(\cdot)$ approximates to the deep Auto Encoder (AE). In classification or regression, we can train the parameters of $\phi(\cdot)$ by adding a decision network behind $\phi(\cdot)$. And then, an elegant design, i.e., a pairwise connected layer is used to connect these two structures. To associate the correspondences between the features and their scores, the outputs of $\phi(\cdot)$ have to multiply by the inputs such that, we fix the weights of the pairwise connected layer equivalent to the inputs. A simple example of NeuralFS is shown in Fig. 3.

For selecting a smaller feature subset, NeuralFS needs to force the scores of the irrelevant features to be zero. To address the issue, the sparse activation function should be used on each neuron of the output layer of $\phi(\cdot)$. Thresholded ReLU (TReLU) [26] is a sparse activation function, and it can force the values that less than a given threshold to be zero. It defined as:

$$\mathrm{TReLU}(x) = \begin{cases} x, & x > theta \\ 0, & x \le theta \end{cases}, \tag{3.1}$$

where x is an unknown value, *theta* denotes a positive value, ReLU is a special instance of TReLU when $theta = 0$. ReLU is the widely used sparse activation function because of its benefits in the gradient descent algorithm. However, in feature selection, forbidding negative feature scores, which is not reasonable. The negative feature scores are as important as the positive scores. Therefore, the linear activation function is likely to be an optimal choice. Nevertheless, the typical linear activation function cannot make feature scores sparse. Fortunately, Thresholded Linear (TL) function [37] is the ideal choice to replace the linear activation function because it can make feature scores sparse and retain the negative scores. The visualization of a TL activation function shown in Fig. 4. In this paper, we use TL function on the output layer of $\phi(\cdot)$ and use ReLU on other hidden layers. The TL activation function can be generally formulated to

$$\mathrm{TL}(x) = \begin{cases} x, & x > theta \text{ or } x < -theta \\ 0, & -theta \le x \le theta \end{cases}. \tag{3.2}$$

Return to the example that is shown in Fig. 3. The model contains a classification layer, a fully-connected network (i.e. $\phi(\cdot)$) with just one hidden layer, and a pairwise connected layer. The pairwise connected layer is added between the fully-connected network and the classification layer. Let we suppose $\boldsymbol{z} = (z_0, \ldots, z_m)^{\mathrm{T}} \in \mathbb{R}^m$ be the output vector of the fully-connected network, and thus, $\boldsymbol{z} = \mathrm{TL}(\mathbf{W}_1 \mathrm{ReLU}(\mathbf{W}_0 \boldsymbol{x}))$, where $\boldsymbol{x} = (x_0, \ldots, x_m)^{\mathrm{T}} \in \mathbb{R}^m$ is an input vector; $\mathbf{W}_i \in \mathbb{R}^{m \times m}$, $i = 0, 1$ is the weight matrix connecting the $(i-1)$th hidden layer to the ith hidden layer. Let $\boldsymbol{o} = (o_0, \ldots, o_c)^{\mathrm{T}} \in \mathbb{R}^c$ be the response vector of the classification layer, therefore $\boldsymbol{o} = g(\mathbf{W}_2(\mathbf{w} \odot \boldsymbol{z}))$, where $\mathbf{W}_2 \in \mathbb{R}^{c \times m}$ means the weight matrix of the decision network. Note that $g(\cdot)$ is an unknown activation function on the classification layer, and commonly use Softmax function, $\mathbf{w} = \boldsymbol{x} \in \mathbb{R}^m$ is the weight vector of the pairwise connected style, and the symbol $\odot$ denotes the entry-wise matrix multiplication. It is worth to notice that $\boldsymbol{w}$ is un-trainable during the process of NeuralFS learning. For simplicity, we use the squared loss for the model such that, the cost function of a NeuralFS can be written as

$$\begin{aligned} loss &= -\frac{1}{2n} \sum_{i=1}^{n} \left\| \boldsymbol{o_i} - \hat{\boldsymbol{y}}_i \right\|_2^2 \\ &= -\frac{1}{2n} \sum_{i=1}^{n} \left\| g(\mathbf{W}_2(\mathbf{w} \odot \boldsymbol{z}_i)) - \hat{\boldsymbol{y}}_i \right\|_2^2 \\ &= -\frac{1}{2n} \sum_{i=1}^{n} \left\| g(\mathbf{W}_2(\boldsymbol{x}_i \odot \mathrm{TL}(\mathbf{W}_1 \mathrm{ReLU}(\mathbf{W}_0 \boldsymbol{x}_i)))) - \hat{\boldsymbol{y}}_i \right\|_2^2, \end{aligned} \tag{3.3}$$

where $\hat{\boldsymbol{y}}_i \in \mathbb{R}^c$ is the one-hot encoding [38] of y_i, $i = 1, \ldots, n$. Assume that $\tilde{\mathbf{W}}_0$ and $\tilde{\mathbf{W}}_1$ are the updated weight matrixes corresponding to $\mathbf{W}_0$ and $\mathbf{W}_1$, respectively. Thus, the feature scores can be obtained by the forward propagation of the fully-connected network. Let $\mathbf{Z} = (\boldsymbol{z}_0, \ldots, \boldsymbol{z}_n) \in \mathbb{R}^{n \times m}$ be the output matrix of the fully-connected network, where $\boldsymbol{z}_j = \left(z_{j,0}, \ldots, z_{j,m}\right)^{\mathrm{T}} \in \mathbb{R}^m$, $j = 1, \ldots, n$, is the activation vector for the jth observation. Therefore, the ith feature score can be computed as

$$s_i = \frac{1}{n} \sum_{k=1}^{n} z_{k,i}. \tag{3.4}$$

The output matrix $\mathbf{Z}$ can be calculated by

$$\begin{aligned} \mathbf{Z} &= \phi\left(\tilde{\mathbf{W}}_0, \tilde{\mathbf{W}}_1, \tilde{\mathbf{X}}\right) \\ &= \left(\mathrm{TL}\left(\mathrm{ReLU}\left(\tilde{\mathbf{X}} \tilde{\mathbf{W}}_0\right) \tilde{\mathbf{W}}_1\right)\right), \end{aligned} \tag{3.5}$$

where $\tilde{\mathbf{X}}$ denotes the normalized input matrix.

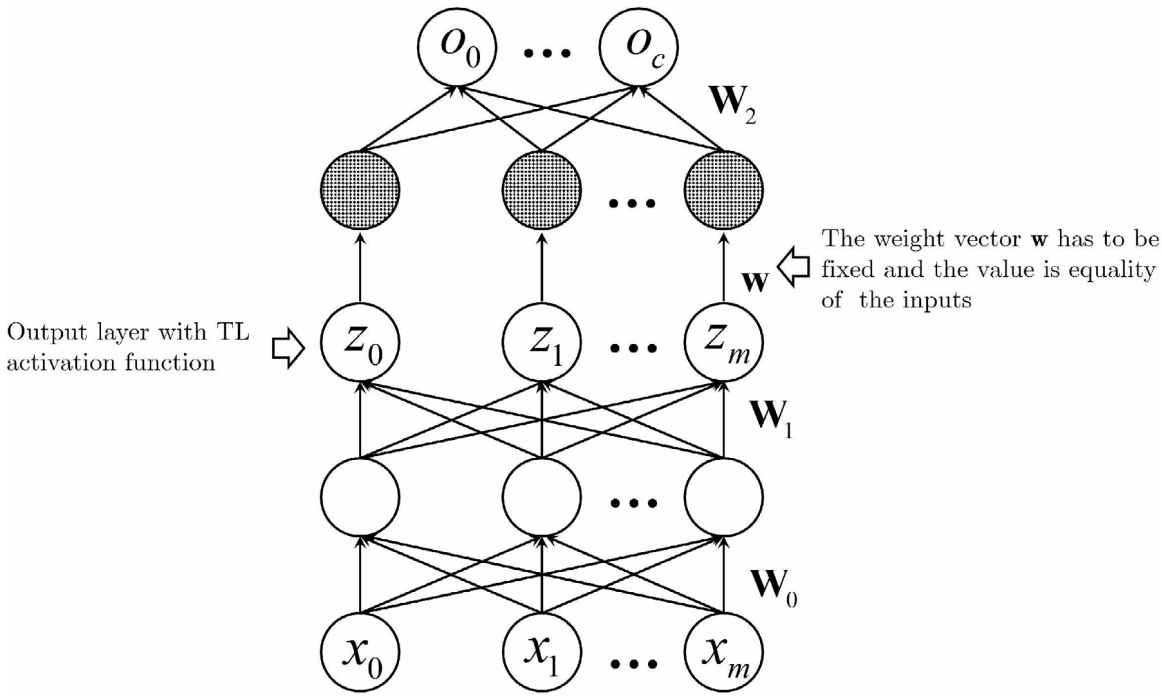


Fig. 3. A simple graphical illustration of NeuralFS.

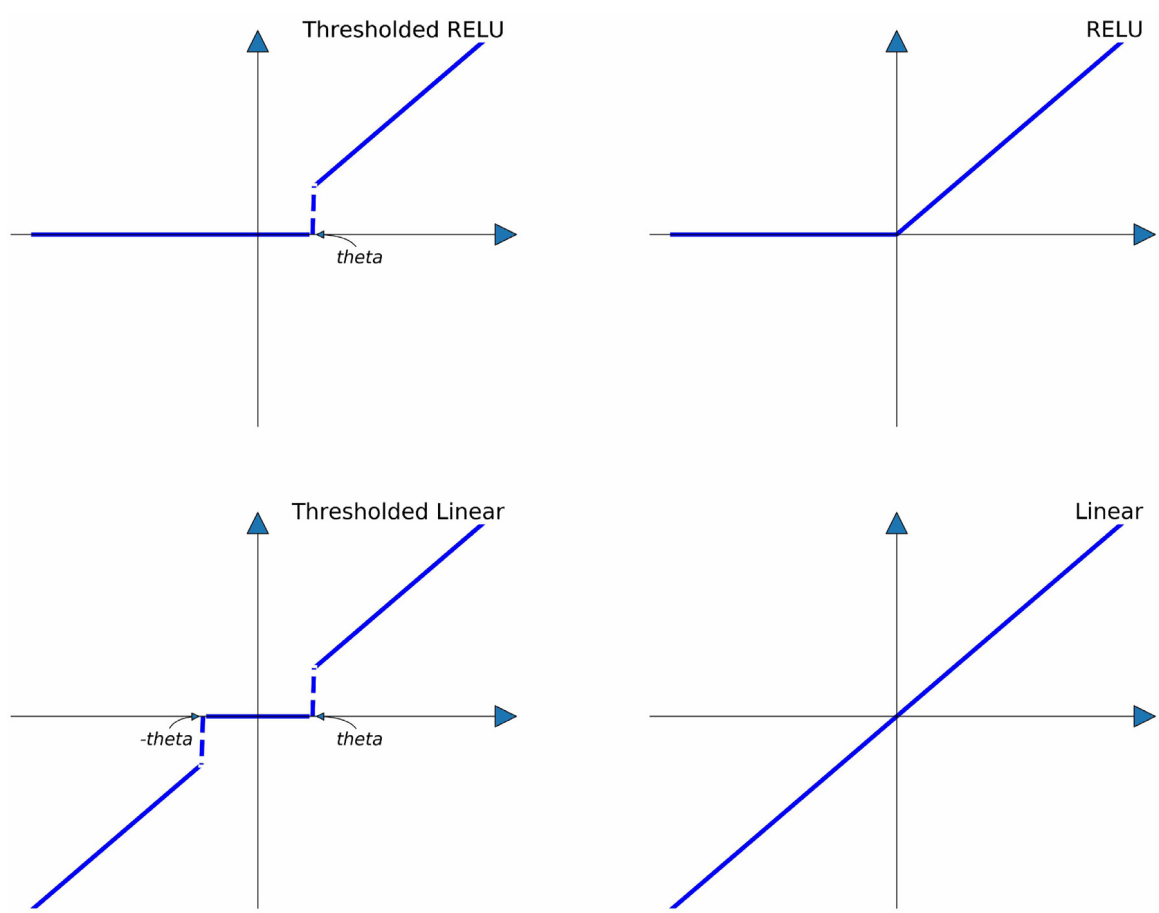


Fig. 4. Activation functions: thresholded ReLU (top-left); ReLU (top-right); thresholded Linear (bottom-left); Linear (bottom-right).

3.3. Reduce overfitting and normalization

For deep learning, an important issue should be noticed: the overfitting appears easily at the deep model. There are many strategies are used to suppress this problem, such as the dropout [39], early stopping [40], and regularization strategies, etc. In this paper, we use the l_2-regularization strategy and the regularization parameter is set flexibly at different machine learning or data mining tasks. Also, the dropout layer is added on NeuralFS with a dropout probability fitted between 0.3 and 0.5. The dropout strategy performs well on preventing overfitting by setting inputs to zero with a given probability. We use Adadelta optimization algorithm [41] to train the entire NeuralFS with the cross-entropy cost function and fix the learning rate as 1.0.

In practice, the input features need to be normalized to avoid the errors of location and scale. Many normalization methods can realize it, such as linear scaling and non-linear scaling methods, etc. The computation of the z-score score is a famous non-linear scaling method. It can be calculated by removing their means and scaling to unit variance. This method is suitable for the distribution of features that look like the standard normal distribution. For the dataset with sparse features, the linear normalization methods are suggested, the methods can scale features between 0 and 1, and retain the sparsity of features.

3.4. Motivation

Why does NeuralFS can learn the feature scores? In fact, the learning process of feature scores can be treated as a regression task (feature scores can be the prediction values), and such, it is natural to create a special regression network (the size of input and output is equally) that can output the prediction values. In the supervised learning scenario, the training dataset with true prediction values is essential for the regression problem. Unfortunately, it is most hardly to provide the prior information. However, in practice, we are not necessary to predict the very precision feature scores, instead, just only the relative value of each feature. For instance, a feature has a strong relation with the response such that, its score will greater than those features that have weak relations. And the noise features would have zero scores. In classification, the label information has enough information to correct the relative feature scores. After optimizing the parameters of NeuralFS, the feature scores can be computed by performing the forward propagation of the fully connected network.

Fig. 5 illustrates our framework. We can find that NeuralFS is not a simple linear model, which has a "shortcut" connection the input level to the output level of the fully-connected network. The "shortcut" provides a direct path for information from the input level. As shown in Fig. 5, after optimization, we can get a group updated parameter of the deep model. Thus, the feature scores $s_1, s_2, \ldots, s_n$ can be calculated by performing the forward propagation of NeuralFS.

Compared to the linear feature selection models, NeuralFS can learn the non-linear relation between features and labels, therefore, NeuralFS can solve the non-linear problems; In contrast to the deep sparse learning-based methods, NeuralFS makes score sparse by using hard threshold function instead of sparse regularization terms. The major advantage of the procedure is to avoid using hybrid sparse regularization terms. The complex regularizations will add additional parameters to the deep model. Therefore, the deep model will be difficult to optimize, and the bad choice of parameters may lead to the degradation of feature identification performance.

4. Simulation studies

4.1. Simulation settings

In this section, we study the feature identification performance of NeuralFS and five existed methods on synthetic datasets, which are generated according to the formula[1]

$$\boldsymbol{y} = 10\sin(\pi \boldsymbol{x}_0 \boldsymbol{x}_1) - 20(\boldsymbol{x}_2 - 0.5)^2 + 10\boldsymbol{x}_3 + 5\boldsymbol{x}_4 + \boldsymbol{\epsilon}, \tag{4.1}$$

where five relevant features $\boldsymbol{x}_0$ to $\boldsymbol{x}_4$ are drawn from a uniform distribution and $\boldsymbol{\epsilon}$ denotes the standard normal deviate $\mathcal{N}(0, 1)$. There are five irrelevant (noise) features $\boldsymbol{x}_5$ to $\boldsymbol{x}_9$ and independent of the response variable, are added into the original dataset. Additionally, we add four redundancy features $\boldsymbol{x}_{10}$ to $\boldsymbol{x}_{13}$ each of which is strongly correlated with $\boldsymbol{x}_0$ to $\boldsymbol{x}_3$, respectively, generated by $f(x) = x + \mathcal{N}(0, 0.01)$. In this simulation study, a binary classification task is considered, that is, if $y_i \geq \bar{\boldsymbol{y}}$ (mean of $\boldsymbol{y}$), the features make up a positive instance, and conversely, $y_i < \bar{\boldsymbol{y}}$, making up a negative instance.

4.2. Simulation results

Five benchmark algorithms include Linear SVM with LASSO penalty term (LSVM), Logistics Regression with LASSO penalty term (LR), DeepFS, DNN, and DeepLIFT. For DNN, the important features are selected by analyzing the activation potential of the input layer of the deep model. For DeepLIFT, the feature scores are computed by using a backpropagation-like algorithm. Among these methods, just DeepLIFT and DNN do not the capability to make features sparse. We simulate 2000 instances to train these models. The visualization of normalized results shown in Fig. 5. It should be pointed out the penalty rates of LR and LSVM are set to 0.1, and the theta of the TL activation function is fixed as 0.005.

As shown in Fig. 6, we can find that DNN-based feature selection methods can identify the non-linear relation between $\boldsymbol{x}_2$ and the response, while the LASSO-based method trends to select only one feature from the highly correlated group. Additionally, the behavior of LASSO-based method will help select the top features for dimension reduction, but not necessarily for data interpretation, i.e., the strong relations of $\boldsymbol{x}_{10}$ to $\boldsymbol{x}_{13}$ do not present. Although DeepFS, DeepLIFT, and DNN can give a score for each of $\boldsymbol{x}_{10}$ to $\boldsymbol{x}_{13}$, there have redundancy features in the top 5 performing features, such as in the top 5 performing features for DNN, $\boldsymbol{x}_1$ and $\boldsymbol{x}_{11}$ are selected. Our algorithm can make a useful compromise between data interpretation and top feature selection. And the scores of irrelevant features $\boldsymbol{x}_6$ to $\boldsymbol{x}_9$ are zero.

To evaluate further the power of feature selection on the irrelevant features increasing scenario, we add 10 and 50 irrelevant features into the original dataset that is generated from formula (4.1). The results of our algorithm, LSVM, and LR on the synthetic dataset are illustrated in Figs. 7 and 8. It is clear that our algorithm picks out all features that have a strong relation with the response, and the top 5 performing features have not noise and redundancy ones, except $\boldsymbol{x}_1$ and $\boldsymbol{x}_{11}$ are selected in the top 5 on the synthetic dataset with 10 irrelevant features. The LSVM and LR have close power for irrelevant features removal on two synthetic datasets, and the top 5 features selected include irrelevant or redundancy ones. It should be pointed out the features that have non-linear behaviors can be identified by our algorithm, but LSVM and LR cannot. This is consistent with the linear model for feature selection presented in Section 2.

[1] http://blog.datadive.net/selecting-good-features-part-iv-stability-selection-rfe-and-everything-side-by-side/

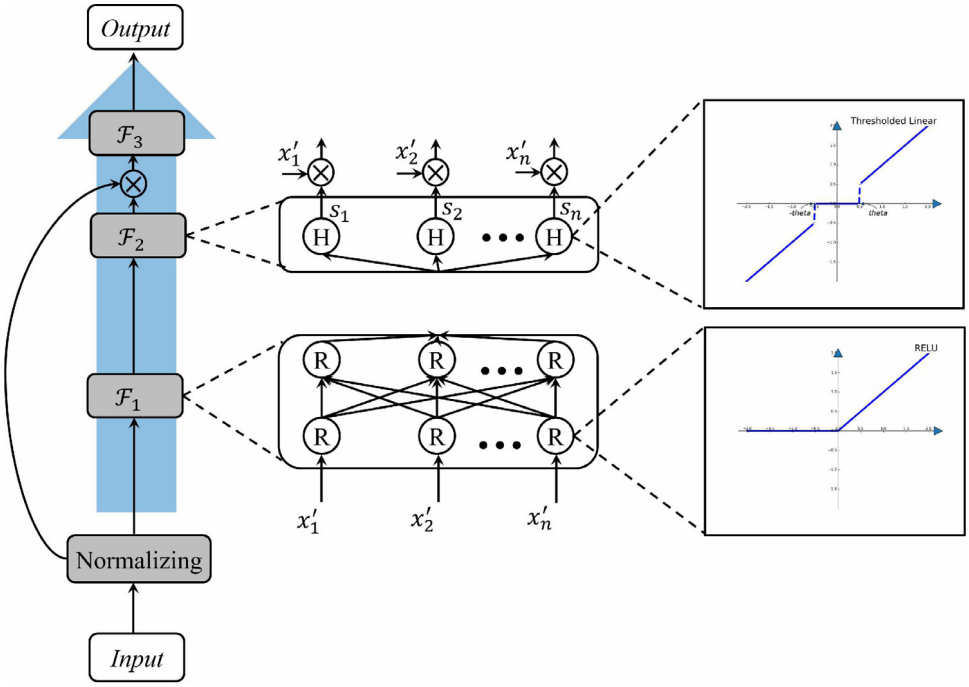


Fig. 5. The network architecture of proposed method. $\mathcal{F}_1$ represents a non-linear function, $\mathcal{F}_2$ is a linear function, $\mathcal{F}_3$ is a decision network, x_i is the ith feature, x'_i the ith normalized value, and s_i means the score of the ith feature.

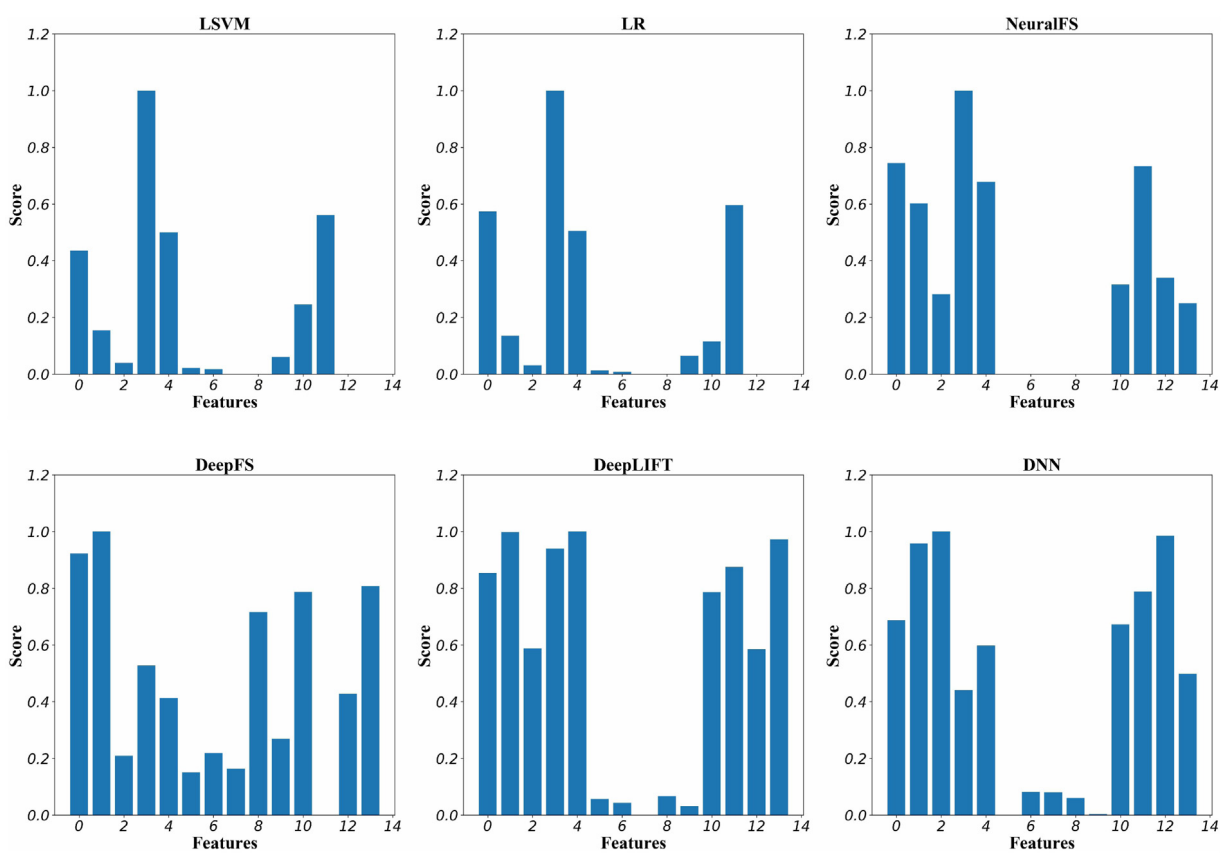


Fig. 6. The feature scores of six method on synthetic dataset with 5 irrelevant features.

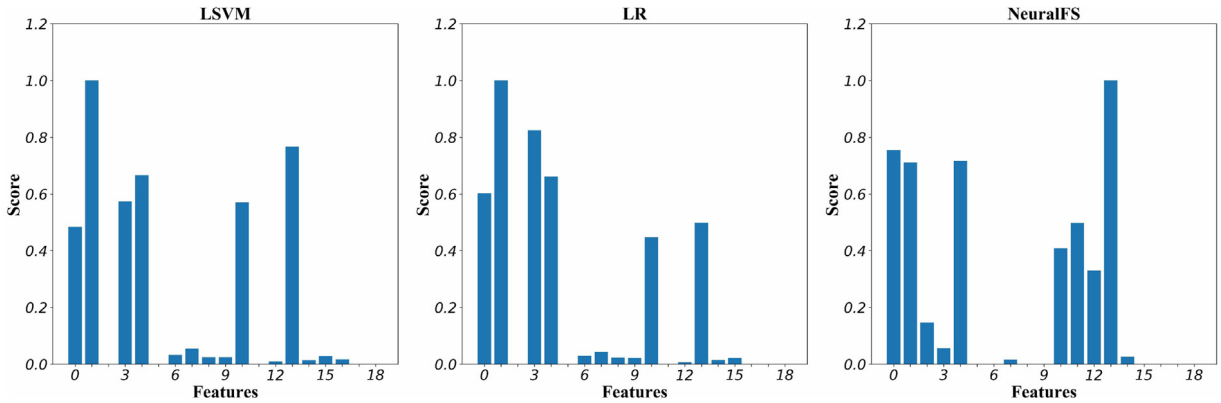


Fig. 7. The feature scores of LSVM, LR, and NeuralFS on synthetic dataset with 10 irrelevant features.

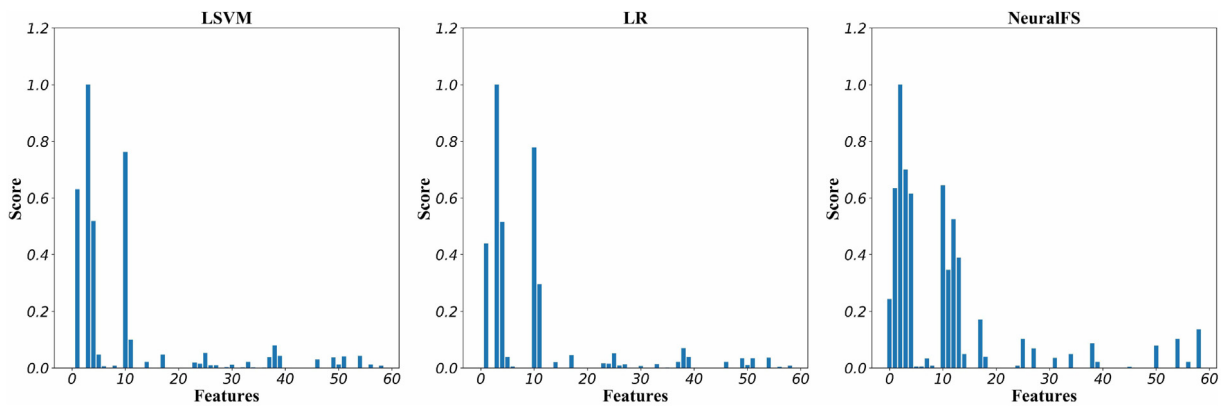


Fig. 8. The feature scores of LSVM, LR, and NeuralFS on synthetic dataset with 50 irrelevant features.

Table 1
Average BERs on 1-NN classifier. (the lower of the BER, the better).

DATASET	NeuralFS	DNN	DeepFS	DeepLIFT	LSVM	LR
ALLAML	***0.122***	0.269	0.206	0.251	0.156	0.198
BASEHOCK	***0.108***	0.196	0.126	0.166	0.125	0.122
GLI_85	***0.186***	0.330	0.223	0.224	0.293	0.371
GLIOMA	0.372	0.282	0.294	***0.270***	0.325	0.314
ISOLET	0.211	0.344	***0.195***	0.409	0.217	0.203
LEUKEMIA	***0.045***	0.092	0.224	0.152	0.093	0.167
LUNG_DISCRETE	***0.188***	0.266	0.302	0.278	0.239	0.357
LUNG	0.190	0.187	0.199	0.216	***0.117***	***0.116***
ORL	***0.165***	0.215	***0.169***	0.269	0.174	0.221
RELATHE	***0.227***	0.390	0.235	0.280	0.296	0.249
USPS	***0.063***	0.103	***0.062***	0.122	0.091	0.076
WARPAR10P	***0.259***	0.497	0.466	0.480	0.347	0.371
WARPPIE10P	***0.023***	0.040	***0.028***	0.061	0.033	0.041
GISETTE	***0.063***	0.217	***0.065***	0.075	0.233	0.227
PROSTATE_GE	***0.171***	0.217	***0.168***	0.281	0.184	0.179
PCMAC	0.204	0.273	***0.188***	0.284	0.230	***0.182***
WIN	***12***	0	7	1	1	2

Table 2
Average BERs on linear SVM classifier. (the lower of the BER, the better).

DATASET	NeuralFS	DNN	DeepFS	DeepLIFT	LSVM	LR
ALLAML	***0.052***	0.198	0.134	0.150	0.070	0.129
BASEHOCK	***0.069***	0.141	0.084	0.099	0.096	0.087
GLI_85	***0.185***	0.328	***0.185***	0.204	0.265	0.317
GLIOMA	***0.349***	0.385	0.374	0.355	0.420	0.366
ISOLET	***0.166***	0.298	***0.167***	0.367	0.180	0.176
LEUKEMIA	***0.046***	0.088	0.137	0.123	0.069	0.128
LUNG_DISCRETE	***0.248***	0.312	0.340	0.340	***0.249***	0.377
LUNG	0.171	0.195	0.228	0.243	***0.104***	***0.100***
ORL	0.193	0.239	***0.179***	0.301	0.185	0.240
RELATHE	***0.190***	0.357	***0.193***	0.242	0.256	0.230
USPS	***0.096***	0.137	***0.096***	0.158	0.117	0.106
WARPAR10P	0.179	0.301	0.176	0.222	***0.170***	0.361
WARPPIE10P	***0.016***	0.023	0.020	0.036	0.030	0.035
GISETTE	***0.050***	0.157	***0.051***	0.075	0.197	0.191
PROSTATE_GE	0.162	0.180	0.121	0.219	***0.111***	***0.116***
PCMAC	***0.130***	0.190	***0.126***	0.166	0.184	0.178
WIN	***12***	0	7	0	4	2

5. Benchmark dataset analysis

5.1. Datasets descriptions

In this section, we evaluate the classification performance of six different algorithms on extensive benchmark datasets, which can be described below.

1. BASEHOCK, PCMAC, and RELATHE are text recognition datasets with binary class. In BASEHOCK and PCMAC, the number of positive classes and the negative classes are mostly equal, while in RELATHE dataset, the negative instances have more 20 percent than the positive instances.
2. USPS is an optical character recognition dataset. It contains 9298 instances, and each instance includes 256 features. These instances come from 10 different classes.
3. ISOLET is a simple dataset for predicting which letter of the alphabet is spoken. This dataset is comprised of 1560 instances, and each of that contains 617 continuous features, and each class has 60 instances in it.
4. LUNG, LUNG_DISCRETE, ALLAML, GLI_85, GLIOMA, and LEUKEMIA are the biology classification datasets. They contain 5, 7, 2, 2, and 4 different classes, respectively. In

Table 3
Average BERs (N) on linear SVM classifier for the selected subset.

DATASET	NeuralFS	ACO	LSVM	LR
ALLAML	***0.039*** (0.461)	0.226 (0.936)	0.060 (***0.002***)	0.111 (***0.001***)
BASEHOCK	0.073 (0.130)	***0.066*** (0.971)	***0.063*** (0.104)	***0.067*** (***0.058***)
GLI_85	***0.145*** (0.704)	/	0.154 (***0.001***)	0.166 (***0.001***)
GLIOMA	***0.260*** (0.966)	0.312 (0.763)	0.484 (***0.011***)	0.398 (***0.001***)
ISOLET	0.077 (***0.441***)	0.076 (0.840)	0.073 (0.830)	***0.067*** (0.565)
LEUKEMIA	***0.047*** (0.437)	0.244 (0.943)	0.067 (0.003)	0.067 (***0.001***)
LUNG_DISCRETE	***0.174*** (0.885)	0.201 (0.455)	0.230 (0.283)	0.412 (***0.029***)
LUNG	***0.062*** (0.779)	0.193 (0.931)	0.077 (0.044)	0.112 (***0.012***)
ORL	0.057 (0.658)	***0.047*** (0.823)	0.062 (0.612)	0.092 (***0.097***)
RELATHE	0.172 (0.266)	***0.137*** (0.976)	0.146 (0.111)	0.145 (***0.067***)
USPS	***0.072*** (0.740)	***0.069*** (0.867)	***0.072*** (0.941)	***0.071*** (***0.832***)
WARPAR10P	0.042 (0.623)	***0.010*** (0.780)	0.076 (0.072)	0.225 (***0.007***)
WARPPIE10P	***0.000*** (0.850)	0.003 (0.801)	***0.000*** (0.033)	0.006 (***0.041***)
GISETTE	0.035 (0.081)	0.032 (0.975)	0.116 (0.136)	***0.018*** (***0.066***)
PROSTATE_GE	***0.096*** (0.699)	0.113 (0.613)	0.184 (0.081)	0.179 (***0.027***)
PCMAC	0.218 (0.159)	0.173 (0.971)	0.145 (0.176)	***0.135*** (***0.110***)
WIN	***9*** (1)	5 (0)	3 (3)	5 (***15***)

addition to LEUKEMIA and LUNG_DISCRETE, each instance of other datasets comprised of continuous features.

5. ORL, WARPAR10P, and WARPPIE10P are the face image datasets, and each instance of them comprised of continuous features. They come form 40, 10, and 10 classes, respectively. And each class has a similar size of instances.
6. GISETTE is a handwritten digit recognition dataset. It contains 10 different classes, and each instance has 5000 continuous features.

Note that these above datasets can be found at the *feature selection repository*. And also, the repository contains around 40 popular feature selection algorithms by python language.

5.2. Assessment metric and model setting

Considering that some datasets (USPS, RELATHE, and LUNG, etc.) are imbalanced in class cardinality, the simple error rate is failed to be a good metric of classification performance. In this section, Balanced Error Rate (BER) [42] was used to evaluating the power of classification. This metric is the average error rate of all classes and will work well on the imbalanced datasets.

In what follows the experiment, the parameters of models will be set as follow, except where noted. For NeuralFS, the *theta* of TL is set as 0.009, and the fully-connected network includes two hidden layers with 100 neurons each of layer. For the LASSO-based methods, the penalty rates are fixed as 0.1, because the same configuration schemes of them work well in simulation studies. Also, the source codes of DeepFS and DeepLIFT can be found in GitHub [43,44], and we use the default values on parameters in source code as the models with the configuration can perform well on the synthetic dataset. The LSVM and LR can be constructed by using Scikit-Learn [45], which is the famous machine learning tool. The DNN can be easily built based on Keras [46]. Note that, the architecture of DeepLIFT and DNN is the same as NeuralFS. For each dataset, it is split into 60 percent for training and 40 percent for testing. We set the number of repetitions for each experiment to 5 such that, the mean of 5 trials will be deemed to the final result. Note that all experiments are repeated independently with different random seeds.

5.3. Classification performance of feature ranking

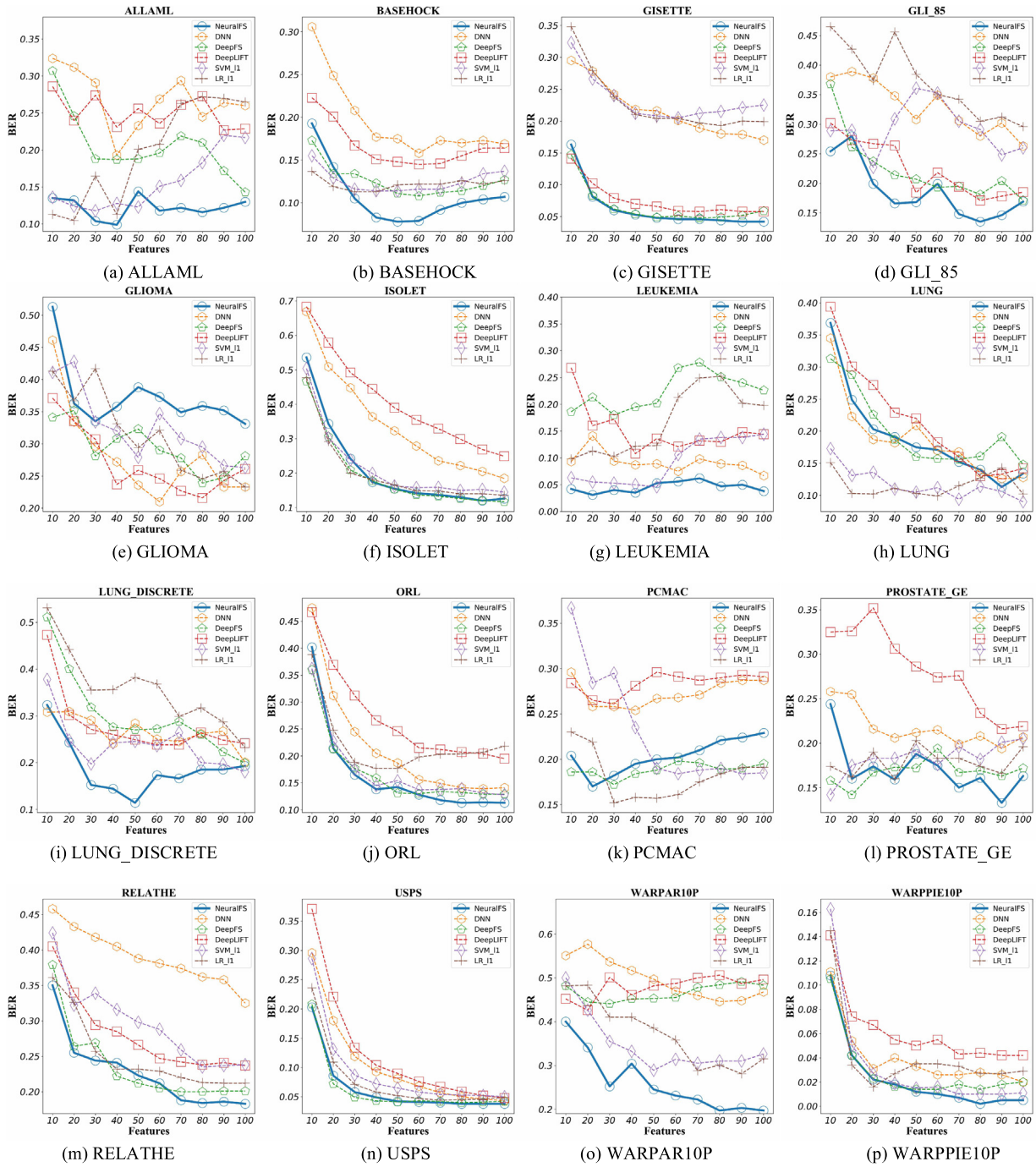
To evaluate the classification performance of selected features, two classifiers are chosen, including the linear Support Vector Machine (SVM) classifier and k-Nearest Neighbor (k-NN) classifier ($k = 1$). Averaged BER of six algorithms on benchmark datasets with top n selected features are recorded in Tables 1 and 2. We choice n, i.e., $n = \{10, 20, 30, \ldots, 100\}$. In the two tables, the lowest redundancy rates and the ones without a significant difference to the lowest are highlighted in boldface and italic. It is clear that NeuralFS performs best on both SVM and 1-NN classifier, that is, it wins on most of the datasets. DeepFS achieves similar results on the SVM classifier and 1-NN classifier, and better than LSVM, LR, DNN, and DeepLIFT. Additionally, we can seem that LSVM performs well on the 1-NN classifier, but the performance is unsatisfactory on the SVM classifier. Also, DeepLIFT and DNN have the worst classification performance.

The variations of BER on the SVM classifier and 1-NN classifier are shown in Figs. 9 and 10. Generally speaking, in terms of classification performance, it tends to increase when all relevant features are selected and the irrelevant features are removed. It is clear that all of the algorithms are meet the trend on all benchmark, except NeuralFS performs classification on LEUKEMIA by the 1-NN classifier. In sum, the results underline that NeuralFS can pick out the top-performing features better than the remaining ones.

5.4. Classification performance of the selected subset

Here, we evaluate the classification performance of the selected subset. The sparse learning-based methods can make feature scores sparse to get a feature subset. Also, the wrapper is a method that can select an optimal subset by score the subsets of feature space. Thus, we add an improved Ant Colony Optimization-based method (ACO) [19] to compare the classification performance. The ACO method can find the (near) optimal subsets in the feature space, however, it takes a very high space and time complexity because to compute correlation matrix and relevance vector. As DNN and DeepLIFT can‘t get a subset, and DeepFS is usually cannot make the feature scores sparse in practical application. Thus, we chose LSVM, LR, ACO, and NeuralFS to test in this trial. The Average BERs of subset on linear SVM classifier are shown in Table 3. Note that, we do not use the 1-NN classifier to test because its results are easy to be affected by the size of subset space. The source code by Python language of ACO for feature selection can be found in GitHub [47]. And we use the defaults configured in the code.

Also, we assess the size of the selected subset. The ideal feature selection method is to get better performance with a small subset of features. We use N to denotes the ratio of features selected. N equals the size of the subset divided by the size of the feature space. The lower of N, the better. Table 3 shows the average BERs and N on linear SVM classifier.



(a) ALLAML (b) BASEHOCK (c) GISETTE (d) GLI_85

(e) GLIOMA (f) ISOLET (g) LEUKEMIA (h) LUNG

(i) LUNG_DISCRETE (j) ORL (k) PCMAC (l) PROSTATE_GE

(m) RELATHE (n) USPS (o) WARPAR10P (p) WARPPIE10P

Fig. 9. Average BERs of the selected feature subsets on the 1-NN classifier.

In classification performance, NeuralFS outperforms significantly better than LSVM, LR, and ACO, where the ACO-based method and LR are the second, LSVM is worst. The main reason for the superiority of NeuralFS is the strong non-linear fitting capability of the deep model. For the number of selected features, LR performs very well and is far better than others. Note that, ACO takes very high execution time on GLI_85 with 22283 features. Thus, we ignore the dataset because the final result is not changed.

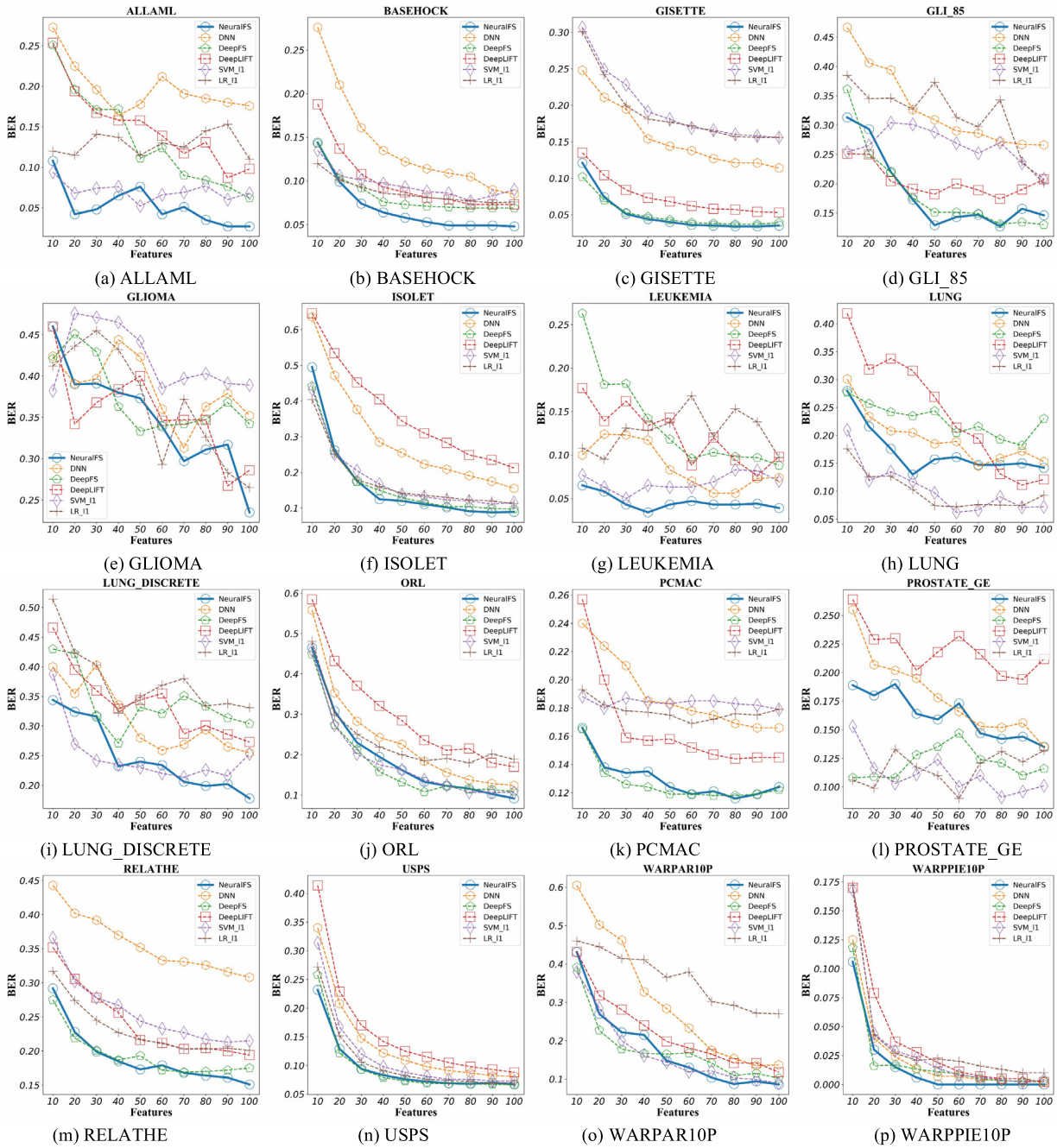
5.5. Redundancy rate analysis

For the selected features, people common hope the correlations among selected features are minimum (low redundancy rate), that is, just select only one feature when several features have similar relations with the response. Therefore, we evaluate further the redundancy rates for those selected features. Redundancy rate can be measured by

$$\mathrm{RED}\,(\boldsymbol{S}) = \frac{1}{m\,(m-1)} \sum_{f_i, f_j \in \boldsymbol{S}, i \neq j} \left|\rho_{i,j}\right|^2, \tag{5.1}$$

where $\boldsymbol{S}$ is the set of selected features; m denotes the size of $\boldsymbol{S}$; $\rho_{i,j}$ is the similarity correlations between two features f_i and f_j. Here, we use the cosine similarity to assess the similarity correlation, i.e.,

$$\rho_{i,j} = \frac{f_i^{\mathrm{T}} f_j}{\|f_i\| \, \|f_j\|}. \tag{5.2}$$



(a) ALLAML (b) BASEHOCK (c) GISETTE (d) GLI_85
(e) GLIOMA (f) ISOLET (g) LEUKEMIA (h) LUNG
(i) LUNG_DISCRETE (j) ORL (k) PCMAC (l) PROSTATE_GE
(m) RELATHE (n) USPS (o) WARPAR10P (p) WARPPIE10P

Fig. 10. Average BERs of the selected feature subsets on the linear SVM classifier.

The averaged redundancy rates of the top 20 features selected by different algorithms are present in Table 4. Similarly, the lowest redundancy rates and the ones without a significant difference to the lowest are highlighted in boldface and italic. We choose the top 20 features because if the number of selected features is larger than the size of the dataset, any feature can be expressed linearly by the remaining ones, lead to unnecessary redundancy is introduced in the evaluation stage [5]. And we can see: LSVM, LR, and NeuralFS get a much lower average redundancy rate than other methods on most of the dataset. The result suggests that NeuralFS has the effectiveness of redundancy removal.

6. Conclusion and future directions

We have introduced NeuralFS, a novel feature selection method that can help to improve the classification performance. NeuralFS consists of a fully-connected neural network to output the feature scores, a decision layer, and a pairwise connected layer to connect the two structures. NeuralFS is different from the existing sparse learning-based methods with sparse regularization terms, NeuralFS projects the features into their corresponding scores. Also, NeuralFS makes feature scores sparse by using the sparse activation function. The effectiveness of NeuralFS is demonstrated empirically on synthetic datasets and benchmark datasets. The results demonstrate NeuralFS can effectively identify relevant features to get a good classification performance, and also, the selected subset has a low redundancy rate.

Due to the fully-connection network is no friendly to image data, and also, the threshold parameter is difficult to choose, and the effect of the parameter change on the performance of NeuralFS is unknown. Thus, there are two directions of this research that deserve further study. (1) An adaptive threshold parameter

Table 4
Average redundancy rate on top 20 selected features. (the lower of the redundancy rate, the better).

DATASET	NeuralFS	DNN	DeepFS	DeepLIFT	LSVM	LR
ALLAML	0.152	0.080	***0.061***	0.069	0.078	***0.065***
BASEHOCK	***0.017***	***0.013***	***0.013***	***0.015***	***0.014***	***0.018***
GLI_85	0.097	***0.045***	0.051	0.050	0.075	0.097
GLIOMA	***0.119***	0.169	0.166	0.170	0.139	0.200
ISOLET	***0.053***	0.072	0.070	0.164	0.077	0.072
LEUKEMIA	0.160	0.080	***0.057***	***0.060***	0.082	***0.060***
LUNG_DISCRETE	0.118	0.092	0.086	0.128	***0.078***	0.093
LUNG	0.089	0.119	0.114	0.088	***0.074***	0.108
ORL	***0.068***	0.078	0.103	0.121	0.083	0.103
RELATHE	0.017	0.030	***0.004***	0.040	***0.003***	***0.005***
USPS	0.071	0.114	***0.065***	0.129	0.073	***0.068***
WARPAR10P	***0.096***	0.205	0.128	0.171	***0.091***	0.198
WARPPIE10P	***0.085***	0.208	0.161	0.199	***0.086***	0.112
GISETTE	0.034	0.036	0.026	0.116	***0.018***	***0.021***
PROSTATE_GE	***0.126***	0.355	0.149	0.202	0.171	0.215
PCMAC	0.019	0.018	***0.015***	***0.014***	***0.013***	***0.012***
WIN	7	2	6	3	***8***	7

learning may be an ideal scheme. The continuous shrinkage functions, like soft threshold function and garrote function, maybe a good choice to replace the hard threshold. The continuous shrinkage functions have the sub-gradient in the whole real number space. Thus, the value of the threshold can be learned by the famous Back Propagation (BP) algorithm. (2) In the proposed method, the input of NeuralFS is a vector such that, it not friendly to the image data. If the image data is transformed into a vector, the local spatial information of the image will be destroyed. Thus, how to use the convolution layer and pooling layer instead of the full connection layer, is another future direction of our method.

CRediT authorship contribution statement

Yingkun Huang: Conceptualization, Methodology, Software, Investigation, Writing - original draft, Writing - review & editing. **Weidong Jin:** Validation, Formal analysis, Visualization, Writing - review & editing. **Zhibin Yu:** Resources, Writing - review & editing, Supervision, Data curation. **Bing Li:** Resources, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was partly supported by the Fundamental Research Funds for the Central Universities (No. 2682017CX046); the Equipment Development Department Funds grant 61403120304, and the Science and Technology Major Project of the Science and Technology Department of Sichuan Province (2018GZDZX0043).

References

[1] J.D. Li, K. Cheng, S. Wang, H. Wang, F. Morstatter, P.T. Robert, J. L.Tang, H. Liu, Feature selection: A data perspective, ACM Comput. Surv. 50 (6) (2018) 94–168.
[2] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (6) (2003) 1157–1182.
[3] F.P. Nie, S.M. Xiang, Y.G. Jia, C.S. Zhang, S.C. Yan, Trace ratio criterion for feature selection, in: Association for the Advance of Artificial Intelligence, AAAI, MIT, 2008, pp. 671–676.
[4] M.R. Sikonja, I. Kononenko, Theoretical and empirical analysis of reliefF and rreliefF, Mach. Learn. 53 (1–2) (2003) 23–69.
[5] Z. Zhao, L. Wang, H. Liu, J. Ye, On similarity preserving feature selection, IEEE Trans. Knowl. Data Eng. 25 (3) (2013) 619–632.
[6] J. Wang, J.M. Wei, Z. Yang, S.Q. Wang, Feature selection by maximizing independent classification information, IEEE Trans. Knowl. Data Eng. 29 (4) (2017) 828–841.
[7] D. Wang, F. Nie, H. Huang, Feature selection via global redundancy minimization, IEEE Trans. Knowl. Data Eng. 27 (10) (2015) 2743–2755.
[8] N.X. Vinh, J. Chan, S. Romano, J. Bailey, Effective global approaches for mutual information-based feature selection, in: ACM Knowledge Discovery and Data Mining, SIGKDD, ACM, 2014, pp. 512–521.
[9] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Neural Information Processing Systems, NIPS, MIT, 2006, pp. 507–514.
[10] T.T. Le, R.J. Urbanowicz, J.H. Moore, B.A. Mckinney, Statistical inference relief (STIR) feature selection, Bioinformatics 35 (8) (2019) 1358–1365.
[11] Q. Song, H. Jiang, J. Liu, Feature selection based on FDA and f-score for multi-class classification, Expert Syst. Appl. 81 (2017) 22–27.
[12] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1–2) (1997) 273–324.
[13] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1998) 95–99.
[14] P.M. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection, IEEE Trans. Comput. 1977 (9) (1997) 917–922.
[15] L.M. Abualigah, A.T. Khader, Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering, J. Supercomput. 73 (11) (2017) 4773–4795.
[16] L. Abualigah, Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications, Neural Comput. Appl. (2020) 1–21.
[17] H. Chen, Q. Zhang, J. Luo, Y. Xu, X. Zhang, An enhanced bacterial foraging optimization and its application for training kernel extreme learning machine, Appl. Soft Comput. 86 (2019) 105884.
[18] X. Zhao, D. Li, B. Yang, C. Ma, Y. Zhu, H. Chen, Feature selection based on improved ant colony optimization for online detection of foreign fiber in cotton, Appl. Soft Comput. 24 (2014) 585–596.
[19] S. Kashef, H. Nezamabadi-Pour, An advanced ACO algorithm for feature subset selection, Neurocomputing 147 (2015) 271–279.
[20] R. Tibshirani, Regression shrinkage and selection via the LASSO, J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1) (1996) 267–288.
[21] H.A. Thi, T.P. Dinh, M. Thiao, Efficient approaches for $l_2 - l_0$ regularization and applications to feature selection in SVM, Appl. Intell. 45 (2) (2016) 549–565.
[22] J. Han, Z. Sun, H. Hao, l_0-norm-based structural sparse least square regression for feature selection, Pattern Recognit. 48 (12) (2015) 3927–3940.
[23] Y. Li, C.Y. Chen, W.W. Wasserman, Deep feature selection: theory and application to identify enhancers and promoters, J. Comput. Biol. 23 (5) (2016) 322–336.
[24] Y. Yamada, O. Lindenbaum, S. Negahban, Y. Kluger, Feature selection using stochastic gates, 2019, arXiv preprint arXiv:1810.04247v6.
[25] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. Ser. B Stat. Methodol. 67 (2) (2005) 301–320.
[26] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, J. R. Stat. Soc. Ser. B Stat. Methodol. 68 (1) (2006) 49–67.
[27] Feature Selection Repository, http://featureselection.asu.edu/index.php.
[28] S. Scardapane, D. Comminiello, A. Hussain, A. Uncini, Group sparse regularization for deep neural networks, Neurocomputing 241 (2017) 81–89.
[29] A. Mnih, D.J. Rezende, Variational inference for Monte Carlo objectives, 2016, arXiv preprint arXiv:1602.06725v2.
[30] Y. Lu, Y. Fan, J. Lv, W.S. Noble, DeepPINK: reproducible feature selection in deep neural networks, in: Neural Information Processing Systems, NIPS, MIT, 2018, pp. 8676–8686.
[31] R.F. Barber, E.J. Candes, Controlling the false discovery rate via knockoffs, Ann. Statist. 43 (5) (2015) 2055–2085.
[32] D. Roy, K.S.R. Murty, C.K. Mohan, Feature selection using deep neural networks, in: International Joint Conference on Neural Networks, IJCNN, IEEE, 2015.
[33] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: International Conference on Machine Learning, ICML, DBLP, 2017, pp. 3145–3153.
[34] Q. Zou, L. Ni, T. Zhang, Q. Wang, Deep learning-based feature selection for remote sensing scene classification, IEEE Geosci. Remote Sens. Lett. 12 (11) (2015) 2321–2325.
[35] L. Hao, Q. Hu, W. Wang, Heterogeneous feature selection with multi-modal deep neural networks and sparse group LASSO, IEEE Trans. Multimed. 17 (11) (2015) 1936–1948.
[36] M.M. Kabir, M.M. Islam, K. Murase, A new wrapper feature selection approach using neural network, Neurocomputing 73 (16) (2010) 3273–3283.
[37] K. Konda, R. Memisevic, D. Krueger, Zero-bias autoencoders and the benefits of co-adapting features, 2014, arXiv preprint arXiv:1402.3337.

[38] J. Buckman, A. Roy, C. Raffel, I. Goodfellow, Thermometer encoding: one hot way to resist adversarial examples, in: International Conference on Learning Representations, ICLR, 2018.
[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[40] Y. Yao, L. Rosasco, A. Caponnetto, On early stopping in gradient descent learning, Constr. Approx. 26 (2) (2007) 289–315.
[41] M.D. Zeiler, Adadelta: an adaptive learning rate method, 2012, arXiv preprint arXiv:1212.5701.
[42] I. Guyon, J.W. Li, T. Mader, P.A. Pletscher, G. Schneider, M. Uhr, Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark, Pattern Recognit. Lett. 28 (12) (2007) 1438–1444.
[43] DeepFS source code by Keras, https://github.com/tonyabracadabra/DeepFeatureSelection.
[44] DeepLIFT source code by Keras, https://github.com/kundajelab/deeplift.
[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blon-del, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (10) (2012) 2825–2830.
[46] Keras-Python deep learning API, https://github.com/fchollet/keras.
[47] ACO for feature selection, https://github.com/L-ashwin/Advanced-ACO.