Neighborhood Component Feature Selection for High-Dimensional Data

Article *in* Journal of Computers · January 2012

DOI: 10.4304/jcp.7.1.161-168 · Source: DBLP

Wei Yang
8 PUBLICATIONS **85** CITATIONS
SEE PROFILE

Kuanquan Wang
Harbin Institute of Technology
427 PUBLICATIONS **4,602** CITATIONS
SEE PROFILE

Wangmeng Zuo
Harbin Institute of Technology
325 PUBLICATIONS **6,106** CITATIONS
SEE PROFILE

Neighborhood Component Feature Selection for High-Dimensional Data

Wei Yang,Kuanquan Wang and Wangmeng Zuo
Biocomputing Research Centre, School of Computer Science and Technology,
Harbin Institute of Technology, Harbin, 150001
Email: wangkq@hit.edu.cn

***Abstract*— Feature selection is of considerable importance in data mining and machine learning, especially for high dimensional data. In this paper, we propose a novel nearest neighbor-based feature weighting algorithm, which learns a feature weighting vector by maximizing the expected leave-one-out classification accuracy with a regularization term. The algorithm makes no parametric assumptions about the distribution of the data and scales naturally to multiclass problems. Experiments conducted on artificial and real data sets demonstrate that the proposed algorithm is largely insensitive to the increase in the number of irrelevant features and performs better than the state-of-the-art methods in most cases.**

***Index Terms*— feature selection, feature weighting, nearest neighbor**

I. Introduction

With the emergence of a great quantity of high dimensional data in various applications, including information retrieval, automated text categorization, combinatorial chemistry and bioinformatics, feature selection has become more and more important in data mining and machine learning [1]. Feature selection is the technique of selecting a small subset from a given set of features by eliminating irrelevant and redundant features. Proper feature selection not only reduces the dimensions of features and hence amount of data used in learning, but also alleviates the effect of the curse of dimensionality to improve algorithms' generalization performance. Furthermore, it also increases the execution speed and the models' interpretability.

Generally speaking, feature selection algorithms now usually fall into one of the three categories [2]: filter, wrapper and embedded methods. In the filter model, feature selection is done by evaluating feature subset with the criterion functions characterizing the intrinsic properties of the training data, such as interclass distance (e.g., Fisher score), statistical measures (e.g., Chi-squared) and information theoretic measures, not involving the optimization of performance of any specific classifier directly. On the contrary, the last two methods are closely related with specified classification algorithms and perform better than filter methods in most cases. The wrapper model requires one predetermined classifier in feature selection and uses its performance to evaluate the goodness of selected feature subsets. Since the classifier need always be trained for each feature subsets considered, wrapper methods are computationally intensive and thus often intractable for large-scale feature selection problems. In the embedded model, feature selection is built into the classifier construction and gradient descent method is usually used to optimize the feature weights, which indicate the relevance between the corresponding features and the target concept. The advantages of the embedded methods are that they are not only less prone to overfitting but also computationally much more efficient than wrapper methods. In particular, many SVM-based embedded methods have been proposed [3], [4]. More comprehensive reviews on feature selection methodologies can be referred to [2], [5], [6].

Nearest neighbor is a simple and efficient nonlinear decision rule and often yields competitive results compared with the state-of-the-art classification methods, such as support vector machines and neural network. Recently, several nearest neighbor-based feature weighting methods, including RELIEF [7], Simba [8], RGS [9], I-RELIEF [10], LMFW [11], Lmba [12] and FSSun [13], have been successfully developed and shown the better performance on high-dimensional data analysis. Inspired by the previous work [14], we propose a novel nearest neighbor-based feature selection method called neighborhood component feature selection (NCFS) algorithm. The proposed algorithm uses gradient ascent technique to maximize the expected leave-one-out classification accuracy with a regularization term. Experiments conducted on artificial and real data sets show that NCFS is almost insensitive to the increase in the number of irrelevant features and performs better than Simba, LMFW and FSSun in most cases.

The rest of the paper is organized as follows. Section 2 proposes a novel feature selection algorithm based on neighborhood component. Section 3 summarizes some related feature selection approaches. Experiments conducted on toy data and real microarray datasets to evaluate the effectiveness of the proposed algorithm are presented in Section 4. Finally, conclusions are given in Section 5.

II. Neighborhood Component Feature Selection

Let $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_N, y_N)\}$ be a set of training samples, where $\mathbf{x}_i$ is a d-dimensional feature

This work is supported in part by the National Natural Science Foundation of China under No.s 60872099 and 60902099.

vector, $y_i \in \{1, \ldots, C\}$ is its corresponding class label and N is a number of samples. The goal is to find a weighting vector w that lends itself to select the feature subset optimizing nearest neighbor classification. In terms of the weighting vector $\mathbf{w}$, we denote the weighted distance between two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ by:

$$D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{d} w_l^2 |x_{il} - x_{jl}| \tag{1}$$

where w_l is a weight associated with lth feature.
For nearest neighbor classification to succeed, an intuitive and effective strategy is to maximize its leave-one-out classification accuracy on the training set T. However, due to the true leave-one-out accuracy that selects nearest neighbor as a classification reference point is a non-differentiable function, an effective approximation is that the reference point is determined by a probability distribution. Here, the probability of $\mathbf{x}_i$ selects $\mathbf{x}_j$ as its reference point is defined as:

$$p_{ij} = \begin{cases} \frac{\kappa(D_{\mathbf{w}}(\mathbf{x}_i,\mathbf{x}_j))}{\sum_{k\neq i} \kappa(D_{\mathbf{w}}(\mathbf{x}_i,\mathbf{x}_k))}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{2}$$

where $\kappa(z) = \exp(-z/\sigma)$ is a kernel function and the kernel width σ is an input parameter that influences the probability of each points being selected as the reference point. In particular, if $\sigma \to 0$, only the nearest neighbor of the query sample can be selected as its reference point. On the other hand, if $\sigma \to +\infty$, all the points have the same chance being selected apart from the query point. Based on above definition, the probability of the query point $\mathbf{x}_i$ being correctly classified is given by:

$$p_i = \sum_j y_{ij} p_{ij} \tag{3}$$

where $y_{ij} = 1$ if and only if $y_i = y_j$ and $y_{ij} = 0$ otherwise. Therefore, the approximate leave-one-out classification accuracy can be written as:

$$\xi(\mathbf{w}) = \frac{1}{N}\sum_i p_i = \frac{1}{N}\sum_i \sum_j y_{ij} p_{ij} \tag{4}$$

Note that when $\sigma \to 0$, $\xi(\mathbf{w})$ is the true leave-one-out classification accuracy. Moreover, in order to perform feature selection and alleviate overfitting, we further introduce a regularization term and hence obtain the following object function:

$$\xi(\mathbf{w}) = \sum_i \sum_j y_{ij} p_{ij} - \lambda \sum_{l=1}^{d} w_l^2 \tag{5}$$

where $\lambda > 0$ is a regularization parameter which can be tuned via cross validation. Note that the coefficient $1/N$ in (4) is neglected since it only means that the parameter λ makes a corresponding change and the final solution vector is the same. Since the object function $\xi(\mathbf{w})$ is differentiable, its derivative with respect to w_l can be computed:

$$\begin{aligned} \frac{\partial \xi(\mathbf{w})}{\partial w_l} &= \sum_i \sum_j y_{ij} \left[\frac{2}{\sigma} p_{ij} \left(\sum_{k\neq i} p_{ik}|x_{il} - x_{kl}| \right.\right. \\ &\quad \left.\left. -|x_{il} - x_{jl}|\right) w_l\right] - 2\lambda w_l \\ &= \frac{2}{\sigma}\sum_i \left(p_i \sum_{k\neq i} p_{ik}|x_{il} - x_{kl}| \right. \\ &\quad \left. - \sum_j y_{ij} p_{ij} |x_{il} - x_{jl}|\right) w_l - 2\lambda w_l \\ &= 2\left(\frac{1}{\sigma}\sum_i \left(p_i \sum_{j\neq i} p_{ij}|x_{il} - x_{jl}| \right.\right. \\ &\quad \left.\left. - \sum_j y_{ij} p_{ij}|x_{il} - x_{jl}|\right) - \lambda\right) w_l \end{aligned} \tag{6}$$

Using the above derivative leads to the corresponding gradient ascent update equation. We name the proposed algorithm as neighborhood component feature selection (NCFS) and its pseudocode is given in Algorithm 1. It should be noted that we didn't use the line search method to identify the step length α in the iteration. The main reason is that the evaluation of the objective function requires expensive computation. In Algorithm 1, the update of α is determined by our experience. However, we found that it works well in practice.

Algorithm 1 Neighborhood Component Feature Selection

```
1: procedure NCFS($T, \alpha, \sigma, \lambda, \eta$) ▷ $T$: training set, $\alpha$: initial step length, $\sigma$: kernel width, $\lambda$: regularization parameter, $\eta$: small positive constant;
2:     Initialization: $\mathbf{w}^{(0)} = (1, 1, \ldots, 1), \epsilon^{(0)} = -\infty, t = 0$
3:     repeat
4:         for $i = 1, \cdots, N$ do
5:             Compute $p_{ij}$ and $p_i$ using $\mathbf{w}^{(t)}$ according to (2) and (3)
6:         for $l = 1, \cdots, d$ do
7:             $\Delta_l = 2\left(\frac{1}{\sigma}\sum_i \left(p_i \sum_{j\neq i} p_{ij}|x_{il} - x_{jl}| - \sum_j y_{ij} p_{ij}|x_{il} - x_{jl}|\right) - \lambda\right) w_l^{(t)}$
8:         $t = t + 1$
9:         $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + \alpha\mathbf{\Delta}$
10:        $\epsilon^{(t)} = \xi(\mathbf{w}^{(t-1)})$
11:        if $\epsilon^{(t)} > \epsilon^{(t-1)}$ then
12:            $\alpha = 1.01\alpha$
13:        else
14:            $\alpha = 0.4\alpha$
15:    until $|\epsilon^{(t)} - \epsilon^{(t-1)}| < \eta$
16:    $\mathbf{w} = \mathbf{w}^{(t)}$
17:    return $\mathbf{w}$
```

III. Related Work

A. Simba

Simba [8] is a feature weighting algorithm based on the hypothesis-margin of 1-NN. For each sample $\mathbf{x}_i$ in the training set T, its hypothesis-margin is given by:

$$\theta_{\mathbf{w}}(\mathbf{x}_i) = \frac{1}{2}\left(\|\mathbf{x}_i - \mathrm{NM}(\mathbf{x}_i)\|_2 - \|\mathbf{x}_i - \mathrm{NH}(\mathbf{x}_i)\|_2\right) \tag{7}$$

where $\|z\|_2 = \sqrt{\sum_l w_l^2 z_i^2}$, NM($\mathbf{x}_i$) and NH($\mathbf{x}_i$) are the nearest neighbor of $\mathbf{x}_i$ from the different class and the same class under the weighting feature space, respectively. In order to maximize the margins of all the samples, the objective function of Simba is defined as:

$$\varepsilon(\mathbf{w}) = \sum_{i=1}^{N} u(\theta_{\mathbf{w}}(\mathbf{x}_i)) \tag{8}$$

where $u(z)$ is the utility function, which controls the contribution of each margin term to the total sum. In general, the linear utility function $u(z) = z$ and the sigmoid utility $u(z) = 1/(1 + \exp(-\beta z))$ are often used. Note that the regularization is not considered in this model. Therefore,Simba is often prone to overfitting when the training set size is small relative to the dimension of the input.

B. FSSun

FSSun [13] is also a feature weighting algorithm based on the hypothesis-margin.In the algorithm,the Manhattan distance instead of the Euclidean distance is used to define the margin and the nearest neighbors of a given sample are considered as hidden variables. The expectation hypothesis-margin of sample $\mathbf{x}_i$ based on the hidden variables can be written as:

$$\begin{aligned}\rho_{\mathbf{w}}(\mathbf{x}_i) &= \mathrm{E}_{k\sim M_i}\left(\|\mathbf{x}_i - \mathbf{x}_k\|_{\mathbf{w}}\right) - \mathrm{E}_{k\sim H_i}\left(\|\mathbf{x}_i - \mathbf{x}_k\|_{\mathbf{w}}\right)\\ &= \textstyle\sum_{k\in M_i} P\left(\mathbf{x}_k = \mathrm{NM}(\mathbf{x}_i)|\mathbf{w}\right)\mathbf{w}^T|\mathbf{x}_i - \mathbf{x}_k|\\ &\quad - \textstyle\sum_{k\in H_i} P\left(\mathbf{x}_k = \mathrm{NH}(\mathbf{x}_i)|\mathbf{w}\right)\mathbf{w}^T|\mathbf{x}_i - \mathbf{x}_k|\\ &= \mathbf{w}^T\left(\textstyle\sum_{k\in M_i} P\left(\mathbf{x}_k = \mathrm{NM}(\mathbf{x}_i)|\mathbf{w}\right)|\mathbf{x}_i - \mathbf{x}_k|\right.\\ &\quad \left.- \textstyle\sum_{k\in H_i} P\left(\mathbf{x}_k = \mathrm{NH}(\mathbf{x}_i)|\mathbf{w}\right)|\mathbf{x}_i - \mathbf{x}_k|\right)\\ &= \mathbf{w}^T\mathbf{z}_i\end{aligned} \tag{9}$$

where $M_i = \{k : 1 \le k \le N, y_k \ne y_i\}$, $H_i = \{k : 1 \le k \le N, y_k = y_i\}$, $|\cdot|$ is an element wise absolute operator, $P\left(\mathbf{x}_k = \mathrm{NM}(\mathbf{x}_i)|\mathbf{w}\right)$ and $P\left(\mathbf{x}_k = \mathrm{NH}(\mathbf{x}_i)|\mathbf{w}\right)$ are the probabilities of sample $\mathbf{x}_k$ being the nearest miss or hit of $\mathbf{x}_i$, respectively. These probabilities are defined as:

$$P\left(\mathbf{x}_k = \mathrm{NM}(\mathbf{x}_i)|\mathbf{w}\right) = \frac{\kappa(\|\mathbf{x}_i - \mathbf{x}_k\|_{\mathbf{w}})}{\sum_{j\in M_i}\kappa(\|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{w}})}, \forall k \in M_i \tag{10}$$

And

$$P\left(\mathbf{x}_k = \mathrm{NH}(\mathbf{x}_i)|\mathbf{w}\right) = \frac{\kappa(\|\mathbf{x}_i - \mathbf{x}_k\|_{\mathbf{w}})}{\sum_{j\in H_i}\kappa(\|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{w}})}, \forall k \in H_i \tag{11}$$

where $\kappa(\cdot)$ is the same kernel function as before. In particular, FSSun solves the following optimization:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \log\left(1 + \exp\left(-\mathbf{w}^T\mathbf{z}_i\right)\right) + \lambda\|\mathbf{w}\|_1, \text{ s.t. } \mathbf{w} \ge 0 \tag{12}$$

The optimization in (12) can be reformulated to an unconstrained optimization by replacing w_l with w_l^2. Moreover, it should be noted that in FSSun a fixed-point recursion method, in which the derivative of the objective function with respect to $\mathbf{w}$ is obtained by considering $\mathbf{z}_i$ as a constant vector, is used to solve for $\mathbf{w}$ because $\mathbf{z}_i$ implicitly depends on $\mathbf{w}$ through the probabilities.

C. LMFW

LMFW [11] is a feature weighting algorithm based on KNN and largely inspired by the work of Weinberger et al. [15]. For easy of comparison with FSSun and NCFS, we describe LMFW under the Manhattan distance. The main idea of LMFW is that the k-nearest neighbors of each sample $\mathbf{x}_i$ should share the same label y_i while samples with different labels should be separated by a large margin. Before constructing the model of LMFW, the target neighbors of each sample $\mathbf{x}_i$, which are the samples with the same label y_i that we wish to be closest to $\mathbf{x}_i$, should be first determined. In the algorithm, the target neighbors are identified as the k nearest neighbors in the original feature space and do not change during the learning process. The notation $j \to i$ is used to indicate that sample $\mathbf{x}_j$ is a target neighbor of sample $\mathbf{x}_i$. In particular, the objective function of LMFW is given by:

$$\begin{aligned}(1-\mu)\sum_{i=1}^{N}\sum_{j\to i} D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j) + \mu\sum_{i=1}^{N}\sum_{j\to i}\sum_{k=1}^{N}(1 - y_{ik})\,[1\\ + D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j) - D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_k)]_+ + \lambda\sum_{l=1}^{d} w_l^2\end{aligned} \tag{13}$$

where the hinge loss function $[z]_+ = \max(z, 0)$, the indicator variable $y_{ik} = 1$ if and only if $y_i = y_k$, and $y_{ik} = 0$ otherwise, $\mu \in [0, 1]$ is a balance parameter and $\lambda > 0$ is a regularization parameter. It should be noted that the objective function has three competing terms, the first term penalizes large distances between each samples and its target neighbors, and the second term penalizes small distance between each sample and other samples with different labels, while the final term is expect to increase the sparseness of selected features. LMFW is implemented with Linear Programming in [11]. In fact, there exists a fast gradient descent method to minimize the objective function (13). For the detailed description, please refer to the literature [16].

IV. Experimental Results

Two different sets of experiments were carried out. In the first set, the NCFS algorithm was applied to well controlled, toy data in order to assess the ability of eliminating irrelevant features and to show the impact of different parameter settings. In the second, eight widely used microarray datasets were considered. The aim of these experiments was to illustrate the effectiveness of our algorithm in real application. Meanwhile, the comparison of NCFS with the other three algorithms FSSun, LMFW and Simba was made.

A. Toy data

In this section, our objective is to illustrate the behavior of the proposed method in the presence of the irrelevant features and to show the effects of different parameter

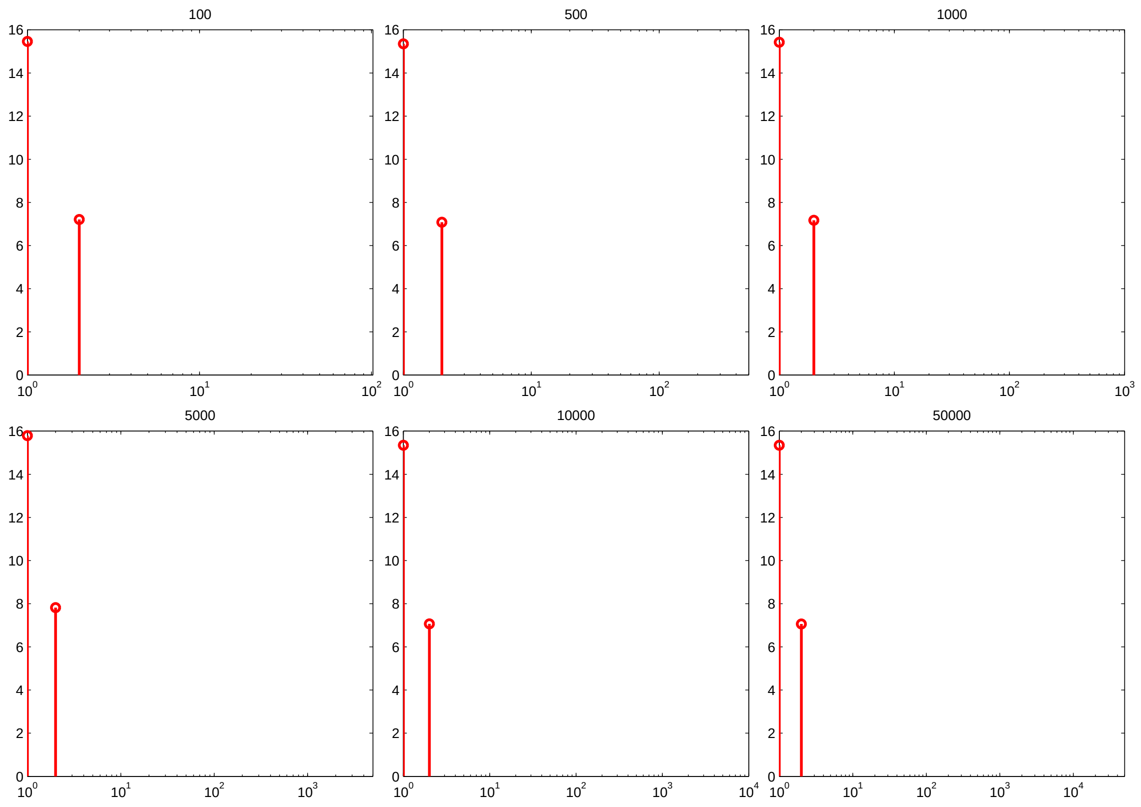

Figure 1. Feature weighting factors learned on the toy dataset with different numbers of irrelevant features, ranging from 100 to 50000. The horizontal coordinate represents the index of each feature, and the vertical coordinate denotes the corresponding weighting factors learned by the proposed algorithm. The two original features are always fixed in the first two indexes.

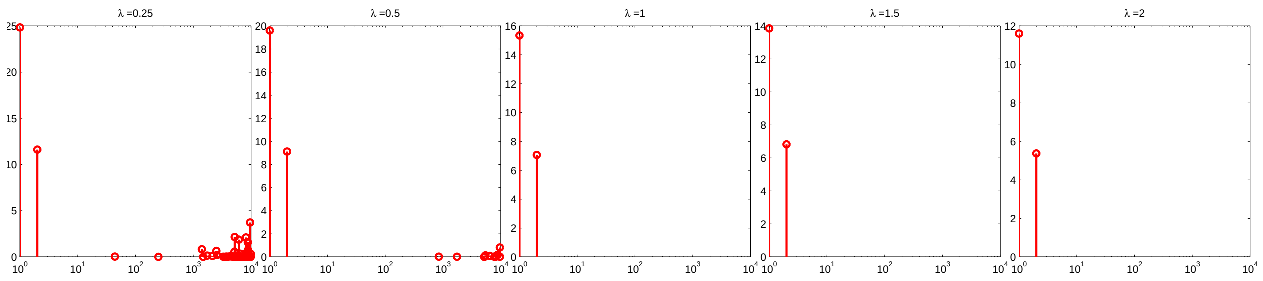

Figure 2. Feature weighting factors learned on the toy dataset with 10000 irrelevant features. The value of kernel width σ is set to 1, and regularization parameter λ is chosen from $\{0.25, 0.5, 1, 1.5, 2\}$.

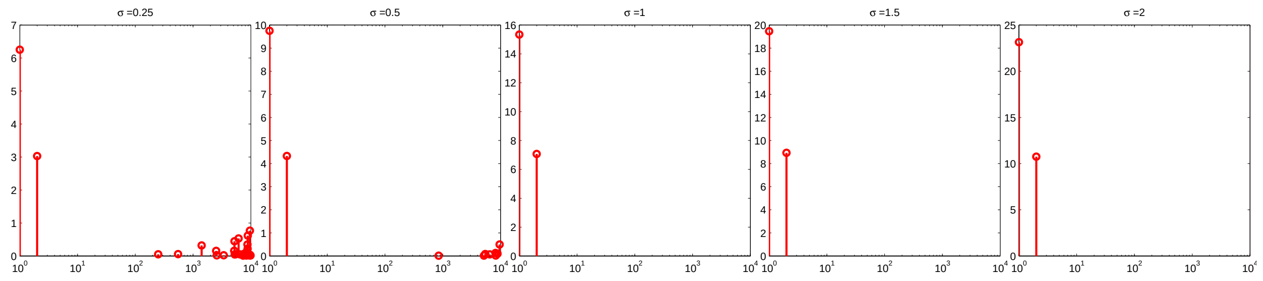

Figure 3. Feature weighting factors learned on the toy dataset with 10000 irrelevant features. The value of regularization parameter λ is set to 1, and kernel width σ is chosen from $\{0.25, 0.5, 1, 1.5, 2\}$.

settings. An artificial non-linear problem, previously considered in [11], [17], [18], was used. It involves two classes in two dimensions. The data for the first class are drawn from $N(\mu_1, \Sigma)$ or $N(\mu_2, \Sigma)$ with equal probability, where mean vectors $\mu_1 = \{-0.75, -3\}$, $\mu_2 = \{0.75, 3\}$ and covariance matrix $\Sigma = \mathbf{I}$. The data for the second class are drawn again from two normal distributions with equal probability, having mean vectors $\mu_1 = \{3, -3\}$,

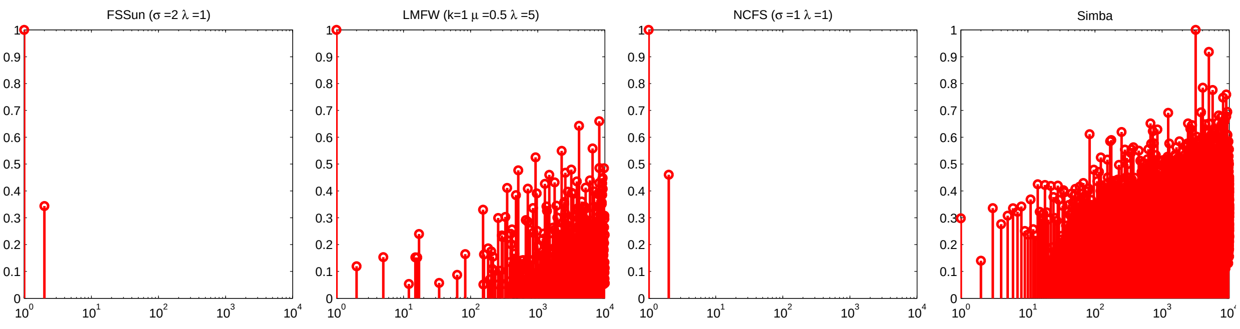

Figure 4. Normalized feature weighting factors of four algorithms learned on the toy dataset with 10000 irrelevant features.

$\mu_2 = \{-3, 3\}$ and the same Σ as before. The dataset with 200 samples is obtained by drawing 100 samples from each class independently. To assess the ability of the algorithm to filter irrelevant features, we added a varying number of irrelevant features to each sample. In this work, the added irrelevant features are drawn from a normal distribution with zero mean and variance 20 and the number of which is set to {100, 500, 1000, 5000, 10000, 50000}. All features are then normalized to the range between 0 and 1. Plots of the learned feature weighting factors of our algorithm on the toy dataset with different numbers of irrelevant features are presented in Fig.1. For the results reported here, the kernel width and regularization parameter of our algorithm are set to 1 and 1, respectively. From the figure, we can see that the proposed algorithm is insensitive to the increase of irrelevant features. For the significantly varying number of irrelevant features, our algorithm gives the almost identical feature weight factors for the two original features. The weights associated with the irrelevant feature are all nearly close to zero, which means that our method can filter the noise features effectively.

Like FSSun, there are a kernel width parameter σ and a regularization parameter λ needing to be adjusted in our algorithm. To study the influences of different parameter settings on the learned feature weighting factors, we performed a test on the toy dataset with 10000 irrelevant features by fixing one parameter while altering the other. The experimental results under different parameter settings are shown in Figs. 2 and 3. It can be observed that the proposed method always give the reasonable weighting factors although the resulting weights vary as the parameters change. Therefore, the ability of our algorithm performing feature selection is insensitive to a specific choice of the two parameters σ and λ, which is the same as FSSun.

In addition, we compare our algorithm with three other algorithms, including FSSun, LMFW and Simba, on the toy dataset with 10000 irrelevant features. The learned feature weights of four algorithms are plotted in Fig.4. For ease of comparison, the maximum value of the learned feature weights of each algorithm was normalized to 1. From the figure, we can see that only FSSun and NCFS could obtain sparse weight vector and successfully identify the two original features by giving them the largest weights. LMFW identifies one and Simba none. The parameter settings of three algorithms FSSun, LMFW and NCFS are also given in Fig.4. In fact, we test a wide range of parameter values of LMFW and found that the resulting weights have no significant difference. The poor performance of LMFW may be because its target neighbors are determined by nearest neighbors in original feature space at the outset of learning and do not change during the learning process. However, the nearest neighbors in original feature space may not be true in the weighted feature space, especially when the feature dimension is very high. Furthermore, the maintenance of unit margin damages the sparseness of its weight vector to some extent. For Simba algorithm, its matlab source code was downloaded from [8]. The sigmoid utility function with default parameter is used since it is less sensitive to outliers than the linear utility. The number of passes of training data and the number of starting points are set to 10 and 5, respectively. Due to the fact that the implementation of Simba is based on stochastic gradient ascent algorithm, we found that its solution vector is different in two runs with the same settings.

Moreover, it should be noted that if we perform feature selection by threshold, i.e., the features whose normalized weighting factor is bigger than a certain threshold are considered useful, the algorithms without sparse solution vector, such as LMFW and Simba, will choose more irrelevant features. Since these algorithms also have possibility to give the larger weight factors for important features, therefore, it is unfair to compare FSSun and NCFS with them based on the threshold. In fact, it is also difficult to determine a suitable threshold for FSSun and NCFS in practical application. Therefore, in the next section, we will perform comparative experiments by weight ranking for fair consideration.

B. *Microarray datasets*

To investigate the ability of the proposed method in practical applications, we further performed the experiments to compare it with three different algorithms LMFW, Simba and FSSun on the eight widely used microarray datasets [19], [20], [21], including Brain_Tumor2, Colon_Cancer, DLBCL, Leukemia1, Lung_Cancer, Lymphoma, Prostate_Tumor and SRBCT. For these datasets, which include 4 two class problems

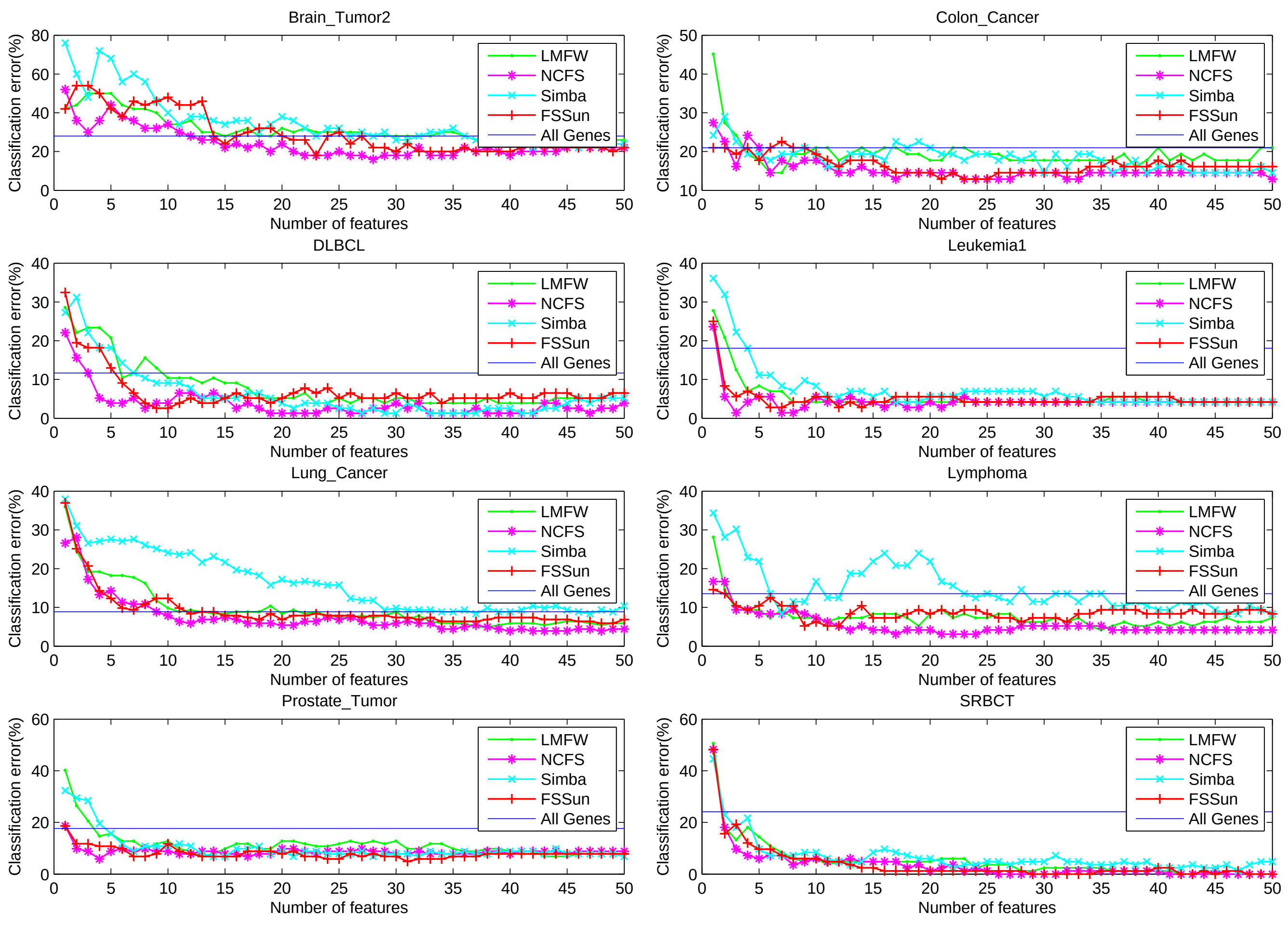

Figure 5. Comparison of four algorithms via the classification error on eight microarray datasets.

TABLE I.
MICROARRAY DATASETS USED IN THE EXPERIMENTS.

Datasets	Samples	Classes	Features
Brain_Tumor2	50	4	10367
Colon_Cancer	62	2	2000
DLBCL	77	2	5469
Leukemia1	72	3	5327
Lung_Cancer	203	5	12600
Lymphoma	96	2	4026
Prostate_Tumor	102	2	10509
SRBCT	83	4	2308

and 4 multi-class problems, one significant characteristic is that the number of samples is remarkably less than the number of features. The detailed data information is given in Table I. Note that the intention of microarray-based analysis is to mine a set of genes that show a strong relationship to phenotypes of interest. Since redundant genes will increase medical examination costs unnecessarily in clinical experiments, the number of resulting genes should be as small as possible. This means that a good feature selection method for microarray analysis should achieve a better prediction performance with the smaller number of features.

As in [10], [13], each feature variable in the raw data is normalized to $[0, 1]$. Since the sample number is very small, we used the leave-one-out cross validation method to carry out a test. Note that the four feature selection methods LMFW, Simba, FSSun and NCFS are all based on nearest neighbor model. Therefore, the KNN with Manhattan distance is used to evaluate the quality of selected features of each algorithm. The only parameter K for KNN is set to 3 for all the tests as in [13]. Before performing the leave-one-out tests, the parameters of four algorithms LMFW, Simba, FSSun and NCFS must be determined. For the first algorithm LMFW, the number of target neighbors is set to 3 and the parameters μ and λ were chosen from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $\{0.1, 0.5, 1, 5, 10\}$ by cross validation, respectively. The settings of Simba are the same as before. For FSSun and NCFS, their kernel widths and regularization parameters are set to 5, 1 and 1, 1, respectively. It should be noted that the parameter settings of FSSun in this study are the same as in [13].

The curves of classification errors of KNN based on the 50 top ranked features derived from four feature selection algorithms on eight microarray datasets are plotted in Fig.5. In order to make a comparison, the classification errors of KNN using all genes are also reported. It can be seen that KNN obtain the significant performance

TABLE II.
CLASSIFICATION ERRORS OF FOUR ALGORITHMS ON EIGHT MICROARRAY DATASETS.

Datasets	LMFW	NCFS	Simba	FSSun
Brain_Tumor2	24.0(41)	**16.0**(28)	22.0(42)	18.0(23)
Colon_Cancer	14.52(6)	**12.9**(17)	14.52(30)	12.9(21)
DLBCL	3.9(23)	**1.3**(19)	**1.3**(27)	2.6(9)
Leukemia1	4.17(8)	**1.39**(3)	4.17(17)	2.78(6)
Lung_Cancer	4.93(37)	**3.94**(40)	8.37(37)	5.91(48)
Lymphoma	4.17(35)	**3.12**(17)	8.33(7)	5.21(9)
Prostate_Tumor	6.86(13)	5.88(4)	6.86(14)	**4.9**(31)
SRBCT	**0**(42)	**0**(26)	1.2(47)	**0**(29)
win/tie/loss	0/1/7	4/3/1	0/1/7	1/2/5

The number in the brackets represents the number of optimal features at which the minimum classification error is obtained. Bold number indicates the minimum classification error of each row. The last row records the win/tie/loss of each algorithm according to the classification error.

improvement by performing feature selection. Moreover, in Table II, we record the minimum classification error of four algorithms on the 50 top ranked features. The numbers of optimal features at which the minimum classification error is obtained are also presented in Table II. From the table, it can be observed that except for Prostate_Tumor, in which FSSun obtains the best classification result, for the remaining seven datasets, NCFS is the clear winner compared to LMFW, Simba and FSSun according to the classification error and the number of optimal features. One possible explanation is that the model of NCFS is more closely related to the leave-one-out classification error than that of the other three methods. We can also find that among the used feature selection methods, FSSun is most comparable to NCFS. Moreover, it is interesting to note that, on DLBCL dataset, Simba also achieves the minimum classification error although with more features compared to NCFS, whereas FSSun and LMFW don't. The win/tie/loss of four algorithms based on the classification error is also given in Table II.

V. CONCLUSION

In this paper, we present a novel feature weighting method in the context of NN. The proposed method, which is called NCFS, uses the gradient ascent technique to maximize the expected leave-one-out classification accuracy with a regularization term. The effectiveness of this algorithm has been evaluated through a number of experiments involving a toy data and eight microarray datasets. Meanwhile, the impact of two parameters, the kernel width σ and the regularization parameter λ, has been studied empirically. Overall, the proposed method is insensitive to a specific choice of the two parameters.

REFERENCES

[1] H. Liu, E. Dougherty, J. Dy, K. Torkkola, E. Tuv, H. Peng, C. Ding, F. Long, M. Berens, L. Parsons *et al.*, "Evolving feature selection," *IEEE Intelligent Systems*, vol. 20, no. 6, pp. 46–76, 2005.

[2] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, *Feature Extraction: Foundations and Applications*. Springer-Verlag, 2006.

[3] S. Maldonado, R. Weber, and J. Basak, "Simultaneous feature selection and classification using kernel-penalized support vector machines," *Information Sciences*, vol. 18, pp. 115–128, 2011.

[4] J. Miranda, R. Montoya, and R. Weber, "Linear penalization support vector machines for feature selection," *Pattern Recognition and Machine Intelligence*, pp. 188–192, 2005.

[5] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[6] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[7] K. Kira and L. Rendell, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning*, 1992, pp. 249–256.

[8] R. Gilad-Bachrach, A. Navot, and N. Tishby, "Margin based feature selection-theory and algorithms," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, pp. 43–50.

[9] A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia, "Nearest neighbor based feature selection for regression and its application to neural activity," in *Advances in Neural Information Processing Systems 18*. MIT Press, 2006, pp. 995–1002.

[10] Y. Sun, "Iterative relief for feature weighting: Algorithms, theories, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1035–1051, 2007.

[11] B. Chen, H. Liu, J. Chai, and Z. Bao, "Large margin feature weighting method via linear programming," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1475–1488, 2009.

[12] Y. Li and B. Lu, "Feature selection based on loss-margin of nearest neighbor classification," *Pattern Recognition*, vol. 42, no. 9, pp. 1914–1921, 2009.

[13] Y. Sun, S. Todorovic, and S. Goodison, "Local learning based feature selection for high dimensional data analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 99, 2009.

[14] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems 17*. MIT Press, 2005, pp. 513–520.

[15] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 1473–1480.

[16] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[17] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for svms," in *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 668–674.

[18] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 131–159, 2002.

[19] A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis," *Bioinformatics*, vol. 21, no. 5, pp. 631–643, 2005.

[20] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and

normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, vol. 96, no. 12, pp. 6745–6750, 1999.

[21] A. Alizadeh, M. Eisen, R. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, X. Yu *et al.*, "Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, no. 6769, pp. 503–511, 2000.

Wei Yang is currently a PhD candidate in the Biocomputing Research Centre at the Harbin Institute of Technology. His interests include bioinformatics and pattern recognition.

Kuanquan Wang is a full professor and PhD supervisor with School of Computer Science and Technology at Harbin Institute of Technology. He is a senior member of IEEE, a senior member of China Computer Federation and a senior member of Chinese Society of Biomedical Engineering. His main research areas include image processing and pattern recognition, biometrics, biocomputing, virtual reality and visualization. So far, he has published over 200 papers and 6 books, got 10 patents, and won 1 second prize of National Teaching Achievement.

Wangmeng Zuo is an associate professor in School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. He received the PhD degree in computer application technology from Harbin Institute of Technology in 2007. From July to December 2004, from November 2005 to August 2006, and from July 2007 to February 2008, he was a research assistant in the Department of Computing, Hong Kong Polytechnic University. From August 2009 to February 2010, he was a visiting professor in Microsoft Research Asia. His research interests include sparse representation, biometrics, pattern recognition, and computer vision.