Knowledge-Based Systems 83 (2015) 81–91

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Accelerating wrapper-based feature selection with *K*-nearest-neighbor

Aiguo Wang [a], Ning An [a,*], Guilin Chen [b], Lian Li [a], Gil Alterovitz [c,d,e]

[a] School of Computer and Information, Hefei University of Technology, Hefei, China
[b] School of Computer and Information Engineering, Chuzhou University, Chuzhou, China
[c] Center for Biomedical Informatics, Harvard Medical School, Boston, USA
[d] Children's Hospital Informatics Program, Boston Children's Hospital, Harvard Medical School, Boston, USA
[e] Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, USA

ARTICLE INFO

Article history:
Received 12 November 2014
Received in revised form 11 March 2015
Accepted 13 March 2015
Available online 21 March 2015

Keywords:
Gene selection
Microarray data
Wrapper
Filter
k-nearest-neighbor

ABSTRACT

Wrapper-based feature subset selection (FSS) methods tend to obtain better classification accuracy than filter methods but are considerably more time-consuming, particularly for applications that have thousands of features, such as microarray data analysis. Accelerating this process without degrading its high accuracy would be of great value for gene expression analysis. In this study, we explored how to reduce the time complexity of wrapper-based FSS with an embedded *K*-Nearest-Neighbor (KNN) classifier. Instead of considering KNN as a black box, we proposed to construct a classifier distance matrix and incrementally update the matrix to accelerate the calculation of the relevance criteria in evaluating the quality of the candidate features. Extensive experiments on eight publicly available microarray datasets were first conducted to demonstrate the effectiveness of the wrapper methods with KNN for selecting informative features. To demonstrate the performance gain in terms of time cost reduction, we then conducted experiments on the eight microarray datasets with the embedded KNN classifiers and analyzed the theoretical time/space complexity. Both the experimental results and theoretical analysis demonstrated that the proposed approach markedly accelerates the wrapper-based feature selection process without degrading the high classification accuracy, and the space complexity analysis indicated that the additional space overhead is affordable in practice.

1. Introduction

The popularization and use of microarray technology in biomedical research and medicine facilitates the high-throughput measurement of many gene expression profiles simultaneously and enables their meaningful application in the diagnosis of cancers and tumor subtypes, the discovery of drug targets and the design of potentially effective drugs at the molecular level [1]. However, the intrinsic nature of microarray data with high dimensionality (as many as thousands of genes) and small sample sizes (as low as tens of samples) limits their powerful potential in practical use. In microarray data classification, the "curse of dimensionality" problem can lead to over-fitting, which can degrade the generalization ability of constructed classifiers in predicting unseen samples [2,3]. In addition, the feature space that is involved can have irrelevant and redundant features and often generates a classifier that has poor performance and weak robustness [4]. The available experimental evidence demonstrates that redundant features deteriorate the performance of the Naïve Bayes classifier and that instance-based learners are sensitive to irrelevant features [5]. One method to alleviate these problems is to remove irrelevant and redundant features from the original feature space using effective feature selection methods [6,7].

Feature subset selection (FSS) for microarray data, which is also known as gene selection, is defined as the process of removing irrelevant and redundant features and the identification of a feature subset that contains the most discriminative information from the original feature space [8]. In addition to reducing the dimensionality of the original feature space, feature selection offers a multitude of advantages that are accompanied by a reduced number of features, such as enhancing the generalization ability of the classifiers, reducing the training time, improving the performance of the classifiers, facilitating data visualization and helping biologists identify the underlying biological mechanisms [9,10].

Based on the framework that has been proposed by Dash and Liu [11], feature selection mainly consists of two components: a

* Corresponding author at: Hefei Tunxi Road 193, Hefei University of Technology, Hefei 230009, China. Tel.: +86 180 1995 6086; fax: +86 551 6290 4642.
E-mail addresses: wangaiguo2546@163.com (A. Wang), ning.g.an@acm.org (N. An), glchen@chzu.edu.cn (G. Chen), llian@hfut.edu.cn (L. Li), gil_alterovitz@hms.harvard.edu (G. Alterovitz).

http://dx.doi.org/10.1016/j.knosys.2015.03.009
0950-7051/

subset generation module and an evaluator module. The feature subset generation module exploits search strategies to generate candidate subsets, whereas the evaluator module measures the goodness of a subset. Depending on whether the evaluator is involved in the classifier, feature selection methods are typically classified into three groups: filter, wrapper and embedded [12]. Filter methods have lower computational complexity and better generalization ability. Because filter methods evaluate the quality of a feature or a subset of features by using only the intrinsic properties of the training samples, they are flexible in combination with a variety of classifiers. In contrast to filter methods, wrapper methods are specific to a given classifier and evaluate the quality of a candidate subset, and these methods tend to obtain better classification performance than the filter methods [13,14]. Embedded methods are special cases of wrapper methods that are characterized by a deeper interaction between the feature selection and the construction of the classifier. Feature subsets are generated when embedded methods are used to construct the classifier. Typical embedded methods include decision tree C4.5 [15] and SVM-RFE algorithms [8].

Although wrapper methods achieve better classification accuracy, a main disadvantage is that they are far more time-consuming in actual use. For an experiment dataset with N features, wrapper methods must evaluate $O(N^2)$ candidate subsets when using the sequential forward selection scheme, and even incremental wrapper methods evaluate a sub-quadratic number of candidate subsets [16,17]. Such high time complexity would require a large amount of CPU time in the case of microarray data, which has thousands of genes [18]. To alleviate this problem and accelerate the process of feature selection, in this study, we investigated the wrapper and incremental wrapper methods with the K-Nearest-Neighbor (KNN) classifier embedded. Rather than considering the KNN classifier as a black box when evaluating the quality of a candidate feature, we constructed and maintained a classifier distance matrix to speed up the feature subset evaluation process and incrementally updated the matrix after adding a candidate feature into the selected feature subset. Because incrementally calculating the distance between any two instances projected over the selected features avoids a large amount of overlapping distance calculations, we expect a large reduction in the time cost. This work is a significant extension of our earlier paper by Wang et al. [19]. In particular, the main contributions of this paper are as follows: (1) we propose to accelerate wrapper-based feature selection by constructing a classifier distance matrix to store the distance between instances projected over the selected features. This helps us incrementally update the matrix when a new feature is selected, avoid constructing a new classifier from scratch, and greatly reduce massively repetitive calculations; (2) the proposed approach can apply to three types of feature selection methods, including wrapper methods with sequential forward/backward selection, incremental wrapper feature selection methods and incremental wrapper feature selection with replacement methods; (3) we test the effectiveness of wrapper-based feature selection method with KNN classifiers on eight benchmark microarray datasets, and compare it with the state-of-the-art feature selectors; (4) we analyze the theoretical time complexity of the proposed approach and experimentally validate its efficiency; (5) we finally analyze the space complexity of the proposed approach.

The remainder of this paper is organized as follows. Section 2 briefly illustrates the classical KNN algorithm. In Section 3, we detail the procedure of wrapper-based sequential forward selection and incremental wrapper-based feature subset selection methods with the KNN classifier and describe an experiment to demonstrate their effectiveness for feature selection in comparison with that of the well-performing feature selector fast correlation-based filter (FCBF). Section 4 describes the improved wrapper method and incremental wrapper method in detail. In Section 5, we first experimentally compare the actual time cost on eight publicly available microarray datasets before and after acceleration, and this experiment is followed by theoretical time and space complexity analysis. The last section concludes the paper with a brief summary and discussion.

2. K-nearest-neighbor classifier

In pattern recognition, K-Nearest-Neighbor (KNN) is a non-parametric learning algorithm that is used for classification and regression [19–21]. Because it is a typical type of instance-based or memory-based learning scheme, all of the computation of KNN is deferred until classification, and no explicit training step is required for constructing a KNN classifier. Therefore, KNN is a very simple but efficient algorithm that exhibits a time complexity of $O(1)$ when training a KNN classifier and of $O(mn + m\log_2 m)$ when classifying a new instance over a training set with m instances and n attributes, where $O(mn)$ is the time complexity for calculating the distances between the new instance and each of the training instances. In addition, $O(m\log_2 m)$ is the time complexity for sorting the distances when finding the k-nearest neighbors of the new instance [22].

In KNN, various distance metrics are used to measure the distance between two instances according to the type of attribute. Given two instances $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{x}' = (x'_1, \ldots, x'_n)$ from experimental samples, the distance $d(x_j; x'_j)$ between two instances projected on an attribute x_j $(1 \leqslant j \leqslant n)$ is calculated as follows. For a categorical variable, $d(x_j; x'_j) = 0$ if $x'_j == x_j$, and $d(x_j; x'_j) = 1$ in other cases; for numerical attributes, the Euclidean and Manhattan distances are among the most commonly used metrics: $d(x_j; x'_j) = \sqrt{(x_j - x'_j)^2}$ for the Euclidean distance and $d(x_j; x'_j) = |x_j - x'_j|$ for the Manhattan distance. In terms of the Euclidean distance metric, the distance $D(\mathbf{x}; \mathbf{x}')$ between $\mathbf{x}$ and $\mathbf{x}'$ can be recursively defined as

$$D(\mathbf{x}; \mathbf{x}')^2 = D(x_1, \ldots, x_{n-1}; x'_1, \ldots, x'_{n-1})^2 + d(x_n; x'_n)^2. \quad (1)$$

In the classification, to predict the class label of a new instance, KNN first finds its k closest neighbors from the training set according to the distance metric and then assigns the dominant label among the k neighbors to the new instance. If $k = 1$, the label of a new instance is determined by its closest neighbor. Due to its implementation simplicity and classification effectiveness, KNN is commonly used as a standard classifier to evaluate and compare the performance of different feature selection algorithms [23–25] and is integrated into the feature selection framework to evaluate the quality of a candidate feature subset [26–29].

3. Wrapper-based feature selection

3.1. Wrapper-based feature selection with sequential forward selection

Because the wrapper method integrates a classifier into the feature selection process to evaluate the quality of a feature or a subset of features, it tends to obtain a classifier with high classification accuracy. Obviously, enumerating all of the possible combinations of feature subsets and evaluating them one by one is the simplest approach and guarantees obtaining the globally optimal feature subset, while the computational complexity grows exponentially at $O(2^N)$ with N number of features. This approach is often unacceptable because it exhibits high time complexity in an actual application, particularly in the case of gene expression profiles, which have thousands of genes. To accelerate this process, researchers have proposed various feature subset search strategies

to generate candidate feature subsets. Commonly used search methods include sequential forward selection (SFS), sequential backward selection (SBS), bidirectional search, sequential floating search, heuristic search, and random search [30]. Among these search strategies, SFS achieves a better tradeoff between the computational complexity and the quality of the obtained feature subset. Starting from an empty set, SFS first selects the feature that is most relevant to the target variable, as evaluated by a classifier and then searches for the next candidate feature that most contributes to the enhancement of the classification accuracy among the remaining features and continues with this process until there is no improvement in accuracy or there is no candidate feature left. Through adopting such a deterministic search strategy, the wrapper method evaluates only $O((S+1)N)$ candidate feature subsets if S features are finally selected and $O(N^2)$ feature subsets in the worst case. Algorithm 1 presents a pseudo-code of the wrapper method with SFS.

3.2. Incremental wrapper-based feature selection

With the aim of further reducing the time cost and obtaining a final feature subset within linear time complexity, a hybrid feature selection method using a combination of filter and wrapper methods, which was denoted incremental wrapper subset selection (IWSS), has been proposed [31]. In contrast to the wrapper method with SFS for selecting the feature that most contributes to the enhancement of the classification accuracy within each run, IWSS first employs a filter method to obtain a sequence of ranked features according to their relevance to the target variable; starting from the first feature, IWSS then incrementally adds features from the sequence of ranked features to the selected subset in a wrapper manner. By integrating the filter and wrapper methods, IWSS not only achieves satisfactory results but also significantly reduces the time complexity to $O(N)$ instead of $O(N^2)$ in SFS [31].

Algorithm 1. Wrapper-based Sequential Forward Selection (SFS)

```
Input:  Data with feature set F and class label C;
Output: S; //selected feature subset
1   acc = 0;
2   S = null;
3   while ~isempty(F) do
4       flag = 0;
5       for i = 1 to length(F) do
6           S_new = add(copy(S); F_i);
7           acc_new = evaluate(classifier, Data^{S_new ∪ {C}});
8           if acc_new > acc then
9               ind = i; acc = acc_new; flag = 1;
10      if flag then
11          S = add(S; F_ind); //add F_ind to S
12          F = del(F; F_ind); //remove F_ind from F
13      else
14          break; //stop feature selection
15  return S;
```

Because the incremental wrapper method adopts the best-first strategy for feature selection, once a feature is selected, it remains in the selected subset until the end of the search process, which limits the search space and could easily lead to a locally optimal solution because a subset that has the best features may not be the best subset. To mitigate this problem, Bermejo et al. proposed the incremental wrapper subset selection with replacement (IWSSr) method that not only considers the addition of a new feature but also allows the interchange between the new feature and one of the previously selected features [32]. Because IWSSr must evaluate the addition and replacement operations, the time complexity of the wrapper evaluation is $O(sN)$ if s features are selected. The worst time complexity is $O(N^2)$ if all of the N features are selected. Algorithm 2 presents the pseudo-code of the IWSSr method, and the IWSS method can be obtained by deleting the replacement section (lines 6 through 11 in Algorithm 2). In Algorithm 2, when evaluating a candidate feature R_i, the IWSSr method addresses R_i by using one of the following three operations: (1) replacing S_j, which is already selected in the feature subset S, with R_i, i.e., the operation swap(S_j, R_i) (lines 7 and 10); (2) adding R_i to the selected subset S, i.e., the operation add(R_i) (line 15); or (3) discarding R_i (line 17). The best operation (*bestOp*) is the operation that contributes the most to the enhancement of the classification accuracy and that satisfies the relevance criteria, as will be discussed in the next subsection. Specifically, in evaluating a candidate feature, if a replacement operation contributes more to the accuracy than an addition operation, the *bestOp* is the replacement operation. In contrast, if the addition operation contributes more to the accuracy than the replacement operation, the *bestOp* is addition, whereas if both the replacement and addition operations fail to enhance the accuracy, the candidate feature being analyzed is discarded. For IWSS, the optional operations are (1) to add the candidate feature to the selected subset or (2) to discard the candidate feature.

3.3. Relevance criteria

In the incremental wrapper subset selection without and with a replacement methods, i.e., IWSS and IWSSr, respectively, the goodness of a candidate feature is assessed by the function evaluate(*classifier*, $Data^{\downarrow S_{new} \cup \{C\}}$), which trains and validates the classifier using a fivefold cross-validation over the dataset *Data* projected over $S_{new} \cup \{C\}$ (C is the target class) [31]. Rather than use the average accuracy of the fivefold cross-validation and conduct a t-test over the fivefold cross-validation results proposed previously [31], we adopted the following criteria [33]: (1) a fivefold cross-validation was employed to decide whether a new feature is added to the selected feature subset S and (2) the new feature f is included only if the average accuracy of the fivefold cross-validation over $Data^{\downarrow S \cup f \cup \{C\}}$ is better than that of the fivefold cross-validation over $Data^{\downarrow S \cup \{C\}}$ and at least MinFoldersBetter (mf) of the five-folds works well. MinFoldersBetter (mf) is actually a counter for recording how many times the five classification accuracies obtained from the fivefold cross-validation over $Data^{\downarrow S \cup f \cup \{C\}}$ is better than the average accuracy of the fivefold cross-validation over $Data^{\downarrow S \cup \{C\}}$. This approach avoids the criticism of using a statistical test with a small sample size. For better control of noise and over-fitting in the feature selection, the recommended empirical values for mf are 2 or 3 [33]. For the wrapper-based SFS method, the criterion is that the new feature f is included only if the average accuracy of fivefold cross-validation over $Data^{\downarrow S \cup f \cup \{C\}}$ is better than that of fivefold cross-validation over $Data^{\downarrow S \cup \{C\}}$. In Algorithm 2, the returned items of the function evaluate(*classifier*, $Data^{\downarrow S_{new} \cup \{C\}}$) include the average classification accuracy acc_{new} of the fivefold cross-validation over $Data^{\downarrow S_{new} \cup \{C\}}$, and the number *num* indicates how many times the five classification accuracies obtained from the fivefold cross-validation are better than the previous average classification accuracy over $Data^{\downarrow S \cup \{C\}}$.

3.4. Experimental evaluation

In this section, we evaluate the quality of the selected feature subset obtained by the above-mentioned methods by comparing the classification accuracy over eight publicly available microarray datasets with high dimensionality and a small sample size, as shown in Table 1; the last column #SFR gives the ratio between the number of samples and the number of features.

Algorithm 2. Incremental Wrapper Subset Selection with Replacement (IWSSr)

```
Input:  Data with feature set F and class label C, MinFoldersBetter mf;
Output: S; //selected feature subset
ranked feature list R using a filter method;
S = {R_1}; //select the first feature of R
acc = evaluate(classifier, Data^{↓S∪{C}});
for i = 2 to n do
    bestOp = null;
    ;// replacement
    for j = 1 to length(S) do
        S_new = update(copy(S), swap(S_j, R_i))
        [acc_new, num] = evaluate(classifier, Data^{↓S_new ∪{C}});
        if (acc_new > acc && num >= mf) then
            bestOp = swap(S_j, R_i);
            acc = acc_new;
    ;// addition
    S_new = update(copy(S), add(R_i));
    [acc_new, num] = evaluate(classifier, Data^{↓S_new ∪{C}});
    if (acc_new > acc && num >= mf) then
        bestOp = add(R_i);
        acc = acc_new;
    ;//replacement or addition
    if bestOp != null then
        update(S, bestOp);
return S;
```

Colon data: *Colon* data consists of 62 samples with 2000 genes in each sample. Of these samples, 40 are diagnosed as tumors, and the remaining 22 are normal samples. The classification task is to distinguish between the tumor and normal samples according to the gene expression profiles [34].

Central Nervous System (*CNS*) data: The task is to predict the patient outcomes for central nervous system embryonal tumors. This dataset contains 60 patient samples with 7129 genes in each sample, and of these samples, 21 are survivors, and 39 are failures [35].

Prostate data: This dataset is composed of 50 non-tumor prostate samples and 52 prostate tumors with 12,600 genes [36]. The task is to identify the expression patterns that correlate with the distinction of prostate tumors from normal samples.

Leukemia1 data: A collection of leukemia patient samples from the bone marrow and peripheral blood is used for distinguishing between acute myeloid leukemia (AML) and acute lymphoma leukemia (ALL) tissues. This dataset contains 72 samples with 7,129 genes: 25 samples of AML and 47 ALL tissues [1]. The classification task is to distinguish these two types of leukemia according to the gene expression profiles.

Leukemia2 data: A collection of leukemia patient samples from bone marrow and peripheral blood for is used for distinguishing between acute myeloid leukemia (AML) and acute lymphoma leukemia (ALL) tissues. The data for the ALL tissues are further divided in terms of B cells and T cells. *Leukemia2* consists of 72 samples with 5327 genes, and of these samples, 38 are of AML, nine are of ALL-B, and 25 are of ALL-T [1]. The task is to build a classification model to distinguish the three subtypes of leukemia.

Table 1
Experimental dataset description.

Dataset	#Features	#Samples	#Classes	#SFR
Colon	2000	62 (40/22)	2	0.031
CNS	7129	60 (39/21)	2	0.008
Prostate	12,600	102 (50/52)	2	0.008
Leukemia1	7129	72 (47/25)	2	0.010
Leukemia2	5327	72 (38/9/25)	3	0.014
DLBCL	7129	77 (58/19)	2	0.011
Ovarian	15,154	253 (91/162)	2	0.017
SRBCT	2308	83 (29/25/11/18)	4	0.036

Diffuse Large-B-Cell Lymphoma (*DLBCL*) data: Diffuse large B-cell lymphomas (DLBCL) and follicular lymphomas (FL) are two B-cell lineage malignancies. There are 7129 genes with 58 DLBCL samples and 19 FL samples in the *DLBCL* data. The goal is to build a classification model to discriminate DLBCL from FL [38].

Ovarian data: The goal of this experiment is to distinguish ovarian cancer from non-cancer using proteomic spectra data. *Ovarian* consists of 253 samples, including 162 ovarian cancers and 91 controls, for all 15,154 identities [39].

Small Round Blue Cell Tumor (*SRBCT*) data: There are four different types of childhood tumors: Ewing's family of tumors (EWS), neuroblastoma (NB), non-Hodgkin lymphoma Burkitt's lymphoma (BL) and rhabdomyosarcoma (RMS). *SRBCT* consists of 83 samples with 2308 genes: 29 samples of EWS, 18 samples of NB, 11 samples of BL and 25 samples of RMS. The classification goal is to distinguish these four subtypes of tumors based on the gene expression profiles [37].

For the purpose of this study, KNN was integrated into the wrapper procedure to evaluate the quality of a candidate feature subset and was also chosen as the classifier to evaluate the final obtained feature subset. In our study, we used ReliefF, which is a distance-based filter measure that has great power in selecting discriminative features and good stability toward the perturbation of the training set [5,40], to generate a ranked feature set from the original feature space. For each method, a 10-fold cross-validation was conducted, and in this process, one fold was used as the test set to evaluate the final selected feature subset while the remaining nine folds were used as the training set [41]. The training set was directed to the IWSS and IWSSr methods for feature selection using the relevance criteria presented in the above section. Specifically, feature selection was conducted on the training set to ensure an unbiased feature selection protocol [42], and the classifier was trained on the training set projected over the selected feature subset and evaluated on the test data projected over the selected features. To demonstrate the effectiveness of the KNN algorithm in wrapper-based feature selection, the commonly used 1-Nearest-Neighbor (1NN) and 3-Nearest-Neighbor (3NN) classifiers were employed to evaluate both the quality of the candidate subsets and the quality of the final selected feature subset. For the 1NN and 3NN classifiers, we used the Euclidean distance metric to calculate the distance between any two instances. In addition, the fast correlation based filter (FCBF) algorithm, a well-performing state-of-the-art feature subset selector [43,44], was used as a comparison with the proposed methods. To measure the quality of the feature subset selected by FCBF, 1NN and 3NN were also used as classifiers.

Tables 2 and 3 present the experimental results for the 1NN and 3NN classifiers, respectively, in terms of the average classification accuracy and the number of selected genes for FCBF, SFS, and IWSS with the IWSSr methods with $mf = \{2, 3\}$ as the superscript. For a comparison, the last two columns present the average accuracy over the original feature space and the number of features of each dataset. The best accuracy achieved by the four methods on each experimental dataset is shown in bold, and the last row "AVE." presents the average accuracy and number of selected genes.

As shown in Table 2, for the IWSS and IWSSr methods, the accuracy with all of the experimental datasets is improved markedly with a large reduction in the feature dimensionality; specifically, the accuracy reaches more than 95% for the *SRBCT*, *Leukemia2* and *DLBCL* datasets with approximately 10.0 features selected, whereas the IWSS method even reaches 100% accuracy with an average of 9.4 features on the *Ovarian* dataset, and its average accuracy increased by 5.1% compared with that obtained with the same approach without feature selection. Although the SFS method does not achieve an accuracy as impressive as that

Table 2
Experimental results of FSS with the 1NN classifier.

Dataset	SFS		$IWSS^2$		$IWSS^3$		$IWSS_r^2$		$IWSS_r^3$		FCBF		Original	
	accu	#gene	accu	#gene	accu	#gene	accu	#gene	accu	#gene	accu	#gene	accu	#gene
Colon	72.4	4.5	76.0	10.8	73.6	11.4	**78.8**	6.8	75.7	5.1	78.0	15.5	77.1	2000
CNS	53.0	4.1	63.1	12.0	64.2	11.0	**66.0**	7.2	65.3	6.1	61.7	38.6	61.5	7129
SRBCT	84.3	5.5	96.7	11.1	**97.8**	12.0	96.1	6.3	88.9	5.2	92.9	42.0	91.1	2308
Leukemia1	87.4	2.0	89.0	8.4	93.0	8.7	93.0	3.5	90.4	3.4	**94.7**	70.6	90.2	7129
Leukemia2	82.0	3.9	95.8	6.1	**98.8**	5.4	94.6	4.4	98.6	3.5	91.3	53.9	88.9	5327
DLBCL	80.5	3.7	90.0	10.5	92.1	10.5	**95.0**	6.2	88.6	5.3	89.1	53.5	87.0	7129
Prostate	79.3	3.9	**92.2**	12.4	88.3	10.8	83.3	6.0	92.1	6.2	90.0	53.4	76.5	12600
Ovarian	98.8	2.9	99.2	8.7	**100.0**	9.4	98.4	4.4	98.8	4.5	**100.0**	31.1	95.2	15154
AVE.	79.7	3.8	87.8	10.0	**88.5**	9.9	88.2	5.6	87.3	4.9	87.2	44.8	83.4	7346.9

Table 3
Experimental results of FSS with the 3NN classifier.

Dataset	SFS		$IWSS^2$		$IWSS^3$		$IWSS_r^2$		$IWSS_r^3$		FCBF		Original	
	accu	#gene	accu	#gene	accu	#gene	accu	#gene	accu	#gene	accu	#gene	accu	#gene
Colon	67.4	4.4	**84.0**	11.4	82.4	10.1	72.9	7.1	79.5	6.4	79.0	15.5	81.7	2000
CNS	53.6	2.5	66.7	12.4	70.0	12.2	**76.9**	6.2	56.9	6.9	70.0	38.6	65.9	7129
SRBCT	87.9	7.0	92.3	12.9	95.9	12.1	94.1	7.0	95.1	7.2	95.1	42.0	92.9	2308
Leukemia1	90.7	3.1	97.1	7.6	95.5	7.1	96.3	4.4	**98.3**	3.8	92.0	70.6	88.9	7129
Leukemia2	88.6	3.5	**97.3**	4.2	**97.3**	4.1	97.1	3.5	95.7	3.6	91.6	53.9	90.1	5327
DLBCL	83.3	3.5	**93.6**	10.7	**93.6**	9.9	92.5	5.3	89.5	5.3	92.1	53.5	87.7	7129
Prostate	87.4	4.6	94.3	10.6	90.2	10.0	**96.0**	6.2	89.4	5.9	91.1	53.4	79.3	12600
Ovarian	**99.2**	2.9	99.2	10.5	98.8	10.3	**99.2**	4.8	97.2	4.8	98.8	31.1	94.0	15154
AVE.	82.3	3.9	**90.6**	10.0	90.5	9.5	**90.6**	5.6	87.7	5.5	88.7	44.8	85.1	7346.9

obtained with IWSS and IWSSr, it greatly reduces the feature dimensions, potentially enhances the generalization ability of the KNN classifier, and reduces the time cost associated with constructing the classifier. Compared with FCBF, the IWSS and IWSSr methods achieved better accuracy on the majority of the experimental datasets and always obtained feature subsets with a smaller size. Specifically, $IWSS^2$ obtained an 87.8% average accuracy with 10.0 features selected, and $IWSS^3$ obtained an 88.5% average accuracy with 9.9 features selected compared with the 87.2% average accuracy and 44.8 features of the FCBF. Additionally, $IWSS_r^2$ obtained an 88.2% average accuracy with 5.6 features selected, and $IWSS_r{}^3$ obtained an 87.3% average accuracy with 4.9 features selected. The SFS method obtained feature subsets with a markedly smaller size, but its classification accuracy was not satisfactory and was worse than that of FCBF in our experiments.

Similarly, as shown in Table 3, the IWSS and IWSSr methods improved the classification accuracy with approximately 10 features finally selected, which constituted a large reduction in the feature dimensions on all of the experimental datasets, and the average accuracy increased by 5.5% compared with that without feature selection. Compared with FCBF, the IWSS and IWSSr methods achieved better accuracy and obtained feature subsets with a smaller size on all of the experimental datasets. For example, $IWSS^2$ achieved a 90.6% average accuracy with 10.0 features selected, and $IWSS_r{}^2$ achieved a 90.6% average accuracy with 5.6 features selected, in comparison to the 88.7% average accuracy with the 44.8 features of FCBF.

Tables 2 and 3 show that these methods achieved improved classification accuracy with a significant reduction in the feature dimensionality and that the IWSS and IWSSr methods outperform the well-performing state-of-the-art feature selection algorithm FCBF in terms of classification accuracy and the size of the final selected feature subset, which demonstrates the effectiveness and superiority of the proposed KNN-wrapper-based feature subset selection method. However, their running time is not trivial. For example, in our study, $IWSS^2$ had a cost of 367.9 s (approximately 6 min) on the *Prostate* dataset for the 1NN case and 441.3 s (approximately 7.4 min) for the 3NN case. Thus, in the next section, we explore the implementation details of these methods, particularly the classifier that is used inside, that are required to reduce the time complexity without degrading the quality of the selected feature subset.

4. Improved wrapper feature selection

In the evaluation of the quality of a candidate feature subset, we previously considered the KNN to be a black box and disregarded the inside implementation details. In this approach, we construct a new KNN classifier from scratch each time when evaluating a new candidate feature subset. In the wrapper-based SFS and incremental wrapper feature selection methods, the final feature subsets are obtained incrementally by searching and evaluating the candidate features one by one. In contrast, there are classifiers that can be constructed incrementally along with the sequence of the selected features rather than constructed from the beginning, and the KNN algorithm is a typical case that can be constructed incrementally when a new feature is included in the selected feature subset, as discussed in the second section.

Because there is no explicit training step for constructing a KNN classifier, all of the computation of the KNN is deferred until the classification, and the actual classification is conducted by comparing the distance between the test instance and all of the training instances and then choosing the k nearest neighbors to determine the class label of the test instance. Therefore, we could construct a distance matrix to maintain the distance between any two different instances in the experimental dataset projected over the selected feature subset. When evaluating a candidate feature, we can incrementally construct a new KNN classifier by adding the distance matrix on the candidate feature to the distance matrix over the selected features rather than calculate it over all of the features. To clarify this procedure, we will now introduce some notation and two definitions before illustrating the improved wrapper and incremental wrapper feature selection methods.

We first introduce the following notation, which is used in the subsequent sections: *Data* is the experimental data with m instances, n features and one target variable C; $F = \{F_1, F_2, \ldots, F_n\}$ is the original feature space of *Data*; $R = \{R_1, R_2, \ldots, R_n\}$ is a ranked feature set of F that is obtained using a filter method; $S = \{S_1, S_2, \ldots, S_s\}$ $(1 \leqslant s \leqslant n)$ is the current selected feature subset; and F_i, R_i, and S_i are the ith features in F, R and S, respectively.

Definition 1 (*attribute distance matrix*). Given a predictive attribute F_i, the attribute distance matrix of F_i consists of the distance between any two different instances in the experimental dataset projected over feature F_i. This matrix is noted as $D(F_i)$.

Definition 2 (*classifier distance matrix*). Given the selected feature subset S, the element of a classifier distance matrix is the distance between any two different instances in the experimental dataset projected over the feature subset S. This element is noted as D.

To provide an intuitive impression of the attribute distance matrix and the classifier distance matrix, we present their logical storage structure in Fig. 1. Each cell of the matrix stores the distance between any two instances, and each row or each column is a distance vector between an instance and the other instances. In actuality, because of the symmetry of the distance between two instances, we need to store only the upper or lower semi-triangular matrix to save on the physical storage space cost. For a numerical variable (as is the case of the gene expression variable), KNN with the Euclidean distance metric is often used. In our study, to incrementally update the distance between two instances, the *squared Euclidean distance* is stored in this matrix. Such an approach ensures that we can incrementally construct the KNN classifier along with the selection of features. When using this matrix to find the k closest instances of a test instance, we can use the square root (by taking the square root of each value in the matrix), or we can directly use the value in the matrix because distance is a non-negative and monotonically increasing metric along with the selection of features.

The classifier distance matrix D not only acts as a fast KNN classifier to conduct the cross-validation for the experimental dataset projected over the selected features but also works together with the attribute distance matrix for the incremental construction of a new classifier when evaluating the next candidate feature. Based on the discussion above, we present the improved wrapper-based SFS method and the incremental wrapper method with the KNN classifier embedded.

4.1. Improved wrapper-based SFS method with KNN

Compared with the classical wrapper-based SFS algorithm (refer to Algorithm 1), we evaluated the quality of a feature subset by performing a fivefold cross-validation on the classifier distance matrix rather than first calculating the distance between the test instance and all of the training instances and then conducting a fivefold cross-validation. When considering a candidate feature F_i, we first calculated the attribute distance matrix $D(F_i)$, and we then obtained a candidate classifier distance matrix D_{new} by adding $D(F_i)$ to D and then performed a fivefold cross-validation on D_{new} to evaluate the quality of the feature subset $S \cup F_i$. For each run, we added the feature F_i that achieves the best accuracy to the selected subset S by replacing D with the corresponding D_{new}, and we then continued by selecting the next feature. The stop criterion was that all of the features had been selected into S or that there was no increase in the classification accuracy when evaluating the remaining features. Algorithm 3 presents the details of the improved KNN-embedded wrapper-based SFS method.

Algorithm 3. KNN-embedded wrapper-based SFS

```
Input:  Data with feature set F and class label C;
Output: S; //selected feature subset
D = null;
acc = 0;
S = null;
while ~isempty(F) do
    flag = 0;
    for i = 1 to length(F) do
        compute attribute distance matrix D(F_i);
        D_new = D + D(F_i);
        acc_new = (5-fold cross validation on D_new);
        if acc_new > acc then
            ind = i;  flag = 1;
            acc = acc_new;  D_best = D_new;
    if flag then
        S = add(S; F_ind); //select the best feature
        D = D_best;
        F = del(F; F_ind); //del F_ind from F
    else
        break;  //stop feature selection
return S;
```

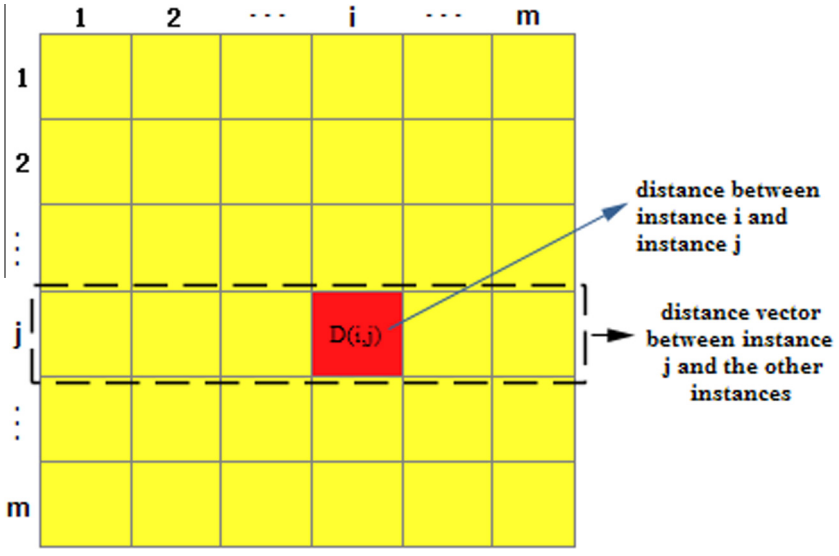

Fig. 1. Illustration of the classifier distance matrix.

4.2. Improved incremental wrapper method with KNN

4.2.1. KNN-embedded IWSS

For the incremental wrapper methods, the candidate features are evaluated by a classifier that runs over the ranked feature set R. At the start of the algorithm, the first feature R_1 is included into S, and the classifier distance matrix D is calculated on S. When evaluating a candidate feature R_i, KNN-embedded IWSS first calculates the attribute distance matrix $D(R_i)$ and generates a candidate classifier distance matrix D_{new} by adding D and $D(R_i)$. If the result of a fivefold cross-validation on D_{new} satisfies the relevance criteria illustrated in the *Relevance Criteria* section, the candidate feature R_i is included in S, and D is replaced by D_{new}; otherwise, R_i is disregarded, and D is maintained unchanged. The above procedure is repeated until no candidate feature is left in R.

4.2.2. KNN-embedded IWSSr

With the exception of the replacement operation in IWSSr, the procedure of IWSSr is not different from that of IWSS. In terms of the replacement operation in IWSSr, for a candidate feature R_i, we must evaluate the quality of the candidate feature subset $\{S_1, S_2, \ldots, S_{j-1}, S_{j+1}, \ldots, S_s, R_i\}$, which is obtained by swapping S_j and R_i in the candidate classifier distance matrix D_{new}. The result can be calculated with the following formula:

$$D_{new} = D - D(S_j) + D(R_i). \tag{2}$$

For the replacement operation, IWSSr evaluates the candidate subset that is obtained by swapping S_j and R_i and records the swapping operation that achieves the best improved performance. In each run, IWSSr updates the selected feature subset with the addition or replacement operation or keeps it unchanged. Because the attribute distance matrix $D(S_j)$ $(S_j \in S)$ has been computed and stored in memory earlier when S_j was evaluated, the proposed approach is expected to greatly accelerate the process. Algorithm 4 presents the pseudo-code of the KNN-embedded IWSSr method.

Algorithm 4. KNN-embedded IWSS$_r$

```
Input:  Data with feature set F and class label C, MinFoldersBetter mf;
Output: S; //selected feature subset
ranked features R using a filter method;
S = {R_1}; //selected feature subset
compute the classifier distance matrix D over S;
acc = (5-fold cross-validation from D);
for i = 2 to n do
    compute attribute distance matrix D(R_i);
    bestOp = null;
    D^best = null;
    ;// replacement
    for j = 1 to length(S) do
        D_new = D − D(S_j) + D(R_i);
        [acc_new, num] = (5-fold cross validation on D_new);
        if (acc_new > acc && num >= mf) then
            bestOp = swap(S_j, R_i);
            acc = acc_new;
            D^best = D_new;
    ;// addition
    D_new = D + D(R_i);
    [acc_new, num] = (5-fold cross validation on D_new)
    if (acc_new > acc && num >= mf) then
        bestOp = add(R_i);
        acc = acc_new;
        D^best = D_new;
    ;// replacement or addition
    if bestOp != null then
        update(S, bestOp);
        D = D^best;
return S;
```

5. Experiments and theoretical analysis

5.1. Experimental results

To demonstrate the performance gain in terms of a reduction in the time cost, experiments were conducted over the same eight microarray datasets used in the *Experimental Evaluation* subsection of Section 3. We implemented these algorithms in the Matlab programming language and ran the experiments on a Quad-core Intel Core i5 CPU (with a 3.2-GHz processor and 4G RAM). For the KNN classifier, 1NN and 3NN were used as the evaluation function in the feature selection. The experimental results are presented in Tables 4 and 5.

Table 4 shows the actual time cost of the SFS, IWSS and IWSSr methods using 1NN as the evaluation function, and Table 5 presents the experimental results obtained for 3NN. Each cell in the table contains the time cost on the corresponding data: the former is the time(s) spent for the case that considers KNN as a black box, and the latter is the time cost obtained when KNN is explicitly embedded. The last row "AVE." represents the average time cost over the eight microarray datasets. We also present the mean time cost comparison on the eight microarray datasets for the two cases, as shown in Figs. 2 and 3.

As shown in Table 4 and Fig. 2, the time cost is significantly reduced in the case in which 1NN is explicitly embedded into the wrapper procedure compared with the case in which 1NN is considered a black box inside. Specifically, in terms of running time, it presents approximately 4.8-, 2.3- and 2.9-fold improvements for the SFS, IWSS and IWSSr methods, respectively. As illustrated in Table 5 and Fig. 3, a high time cost reduction was also achieved when 3NN is explicitly embedded in the feature selection. This case results in 3.4-, 1.7- and 3.9-fold improvements in the time cost for the SFS, IWSS and IWSSr methods, respectively. The experimental results for both 1NN and 3NN demonstrate the efficiency of our proposed approach. Furthermore, because the procedure used for feature selection, i.e., selecting features in a wrapper manner with a sequential forward selection or incremental scheme, and the criteria to include a candidate feature, i.e., using the *relevance criteria* discussed in Section 3, in the KNN-embedded case are not different from those of the black box case, the proposed approach guarantees obtaining the same feature subset and achieving the same high accuracy as in the black box case.

5.2. Time complexity analysis

As discussed in this section of the manuscript, we analyzed the theoretical time complexity of the two types of feature selection methods: the black box case in which the KNN-inside implementation details are disregarded and the case in which KNN is explicitly embedded. For an experimental dataset with m instances, n features and one target variable ($m \ll n$), we used a fivefold cross-validation-based *Relevance Criteria* ($\frac{1}{5}m$ test instances, and $\frac{4}{5}m$ training instances) to determine whether a candidate feature is included in the selected subset, similarly to the experiments. Therefore, the total computational complexity consists mainly of the training time complexity and the test time complexity. Both the average time complexity and the worst time complexity were analyzed.

5.2.1. Black box case

In this case, there is no explicit training phase for the KNN classifier; thus, the training time complexity for SFS, IWSS and IWSSr is a constant, which is noted as O(1). Therefore, we needed to analyze only the test time complexity assuming that $S = \{S_1, S_2, \ldots, S_s\}$ is the selected feature subset, where s is the number of selected features.

(a) SFS:

- Testing: When including a new feature from the remaining features, SFS must evaluate $(n - s)$ candidate features. For each evaluation, the time complexity for classifying an instance is $O(\frac{4}{5}ms + \frac{4}{5}m\log_2\frac{4m}{5})$; there are $\frac{1}{5}m$ test instances

Table 4
Time(s) cost comparison for the black box and 1NN-embedded methods.

Dataset	SFS	$IWSS^2$	$IWSS^3$	$IWSS_r^2$	$IWSS_r^3$
Colon	213.6/34.8	40.6/18.3	40.6/18.3	284.2/80.3	228.9/65.9
CNS	691.0/120.6	140.2/63.3	140.4/63.3	1079.4/259.5	908.3/272.7
SRBCT	326.7/43.2	53.8/23.2	54.0/23.2	369.2/99.0	310.5/97.1
Leukemia1	433.5/129.6	154.0/68.7	154.3/68.9	664.4/300.0	642.7/258.7
Leukemia2	575.8/108.3	121.7/55.3	121.7/54.7	642.2/253.6	532.7/239.7
DLBCL	711.0/132.4	162.5/70.8	162.2/70.7	1127.4/280.8	977.5/312.0
Prostate	1633.4/281.3	367.9/153.9	364.6/154.4	2385.5/745.4	2463.8/529.4
Ovarian	3016.4/678.5	963.0/426.4	977.5/426.6	4567.6/1790.6	4610.7/1751.8
AVE.	950.2/191.1	250.5/110.0	251.9/110.0	1390.0/476.1	1334.4/440.9

Table 5
Time(s) cost comparison for the black box and 3NN-embedded methods.

Dataset	SFS	$IWSS^2$	$IWSS^3$	$IWSS_r^2$	$IWSS_r^3$
Colon	260.7/64.5	47.8/31.7	47.9/31.6	336.6/82.6	321.0/82.5
CNS	550.2/220.9	166.8/110.9	166.4/100.0	1134.2/241.6	1181.9/252.2
SRBCT	494.8/76.9	62.6/38.4	62.4/38.5	476.7/108.1	497.3/100.4
Leukemia1	725.8/244.0	185.6/121.2	183.3/121.2	975.0/339.5	867.6/279.0
Leukemia2	635.1/186.5	140.0/91.8	139.4/92.3	629.7/364.6	638.2/211.1
DLBCL	833.9/252.9	196.7/126.4	195.6/126.7	1187.5/353.5	1199.3/303.4
Prostate	2288.5/535.9	441.3/267.6	438.6/267.4	2978.6/537.0	2868.7/536.8
Ovarian	4653.4/1505.9	1432.8/754.6	1418.7/760.2	7276.9/1782.3	7441.9/1566.8
AVE.	1305.3/385.9	334.2/192.8	331.5/192.2	1874.4/476.1	1877.0/416.5

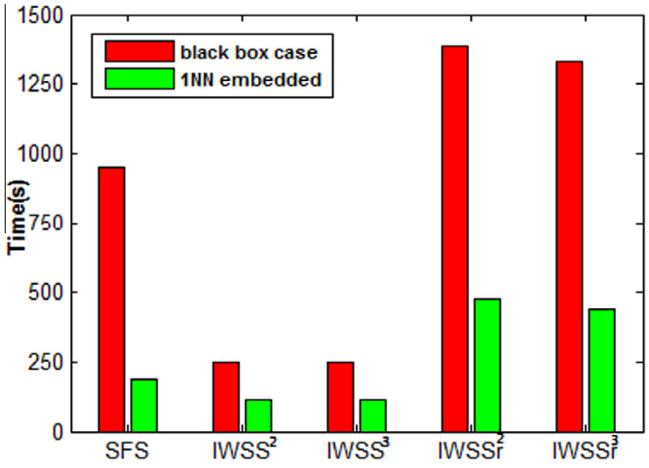

Fig. 2. Mean time cost comparison on the eight microarray data with the 1NN classifier.

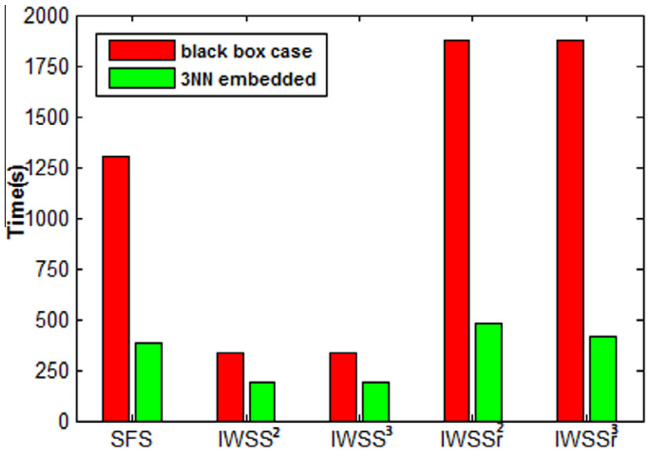

Fig. 3. Mean time cost comparison on the eight microarray datasets with the 3NN classifier.

to be evaluated, and the above process is repeated five times because of the fivefold cross-validation. Thus, the time complexity is $O(\sum_{k=1}^{s}((\frac{4}{5}mk+\frac{4}{5}m\log_2\frac{4m}{5})*\frac{1}{5}m*5*(n-k)))=O(m^2(3n-2s)(s^2+s)+m^2ns\log_2 m)$, with s features finally selected, and the worst case is $O(m^2n^3+m^2n^2+m^2n^2\log_2 m)=O(m^2n^3)$ if all of the features are selected ($n=s$).

(b) IWSS:

- Testing: The time complexity for classifying an instance is $O(\frac{4}{5}ms+\frac{4}{5}m\log_2\frac{4}{5}m)$; the number of iterations is n because n features need to be evaluated; and the process is repeated five times because of the fivefold cross-validation for all of the $\frac{1}{5}m$ test instances. Thus, the time complexity is $O((\frac{4}{5}ms+\frac{4}{5}m\log_2\frac{4m}{5})*n*5*\frac{1}{5}m)=O(m^2ns+m^2n\log_2 m)$ for the average case and $O(m^2n^2)$ for the worst case when all of the features are included ($n=s$).

(c) IWSSr:

- Testing: Slightly different from IWSS, IWSSr must evaluate $(s+1)$ candidate subsets with s replacement and one addition operation within each iteration. Thus, the test time complexity of IWSSr is $O((\frac{4}{5}ms+\frac{4}{5}m\log_2\frac{4m}{5})*n*(s+1)*5*\frac{1}{5}m)=O(m^2ns^2+m^2ns+m^2n\log_2 m)$, and the worst case is $O(m^2n^3)$ when all of the features are included ($n=s$).

5.2.2. KNN-embedded case

For the KNN-embedded case, because the fivefold cross-validation-based relevance criteria calculation is conducted on the classifier distance matrix D, we therefore must maintain this matrix when evaluating the candidate features, and we defined the calculation of D as the training of the KNN classifier. In this case, we must consider the training time complexity.

(a) SFS:

- Training: For the attribute distance matrix of each feature, we must perform a calculation with time complexity $O(nm^2)$ when selecting the first feature. During each iteration, we must calculate $(n-s)$ classifier distance matrices with a time complexity of $O(m^2(n-s))$; the above process is repeated s times because s features are selected. Thus, the time complexity is $O(m^2(ns+n-s^2))$.

Table 6
Summary of the time complexity for the black box and KNN-embedded methods.

Type	SFS		IWSS		IWSSr		Item
	Average	Worst	Average	Worst	Average	Worst	
Black box	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	Training
	$O(m^2(3n-2s)(s^2+s)+m^2ns\log_2 m)$	$O(m^2n^3)$	$O(m^2ns+m^2n\log_2 m)$	$O(m^2n^2)$	$O(m^2ns^2+m^2ns+m^2n\log_2 m)$	$O(m^2n^3)$	Test
	$O(m^2(3n-2s)(s^2+s)+m^2ns\log_2 m)$	$O(m^2n^3)$	$O(m^2ns+m^2n\log_2 m)$	$O(m^2n^2)$	$O(m^2ns^2+m^2ns+m^2n\log_2 m)$	$O(m^2n^3)$	Total
Embedded	$O(m^2(ns+n-s^2))$	$O(m^2n^2)$	$O(m^2n)$	$O(m^2n)$	$O(m^2ns+m^2n)$	$O(m^2n^2)$	Training
	$O(m^2(ns+s-s^2)\log_2 m)$	$O(m^2n)$	$O(m^2n\log_2 m)$	$O(m^2n\log_2 m)$	$O(m^2n(s+1)\log_2 m)$	$O(m^2n^2\log_2 m)$	Test
	$O(m^2(ns+s-s^2)\log_2 m)$	$O(m^2n^2)$	$O(m^2n\log_2 m)$	$O(m^2n\log_2 m)$	$O(m^2ns\log_2 m)$	$O(m^2n^2\log_2 m)$	Total

Table 7
Summary of the space complexity for the black box and KNN-embedded methods.

Type	SFS		IWSS		IWSSr	
	Average	Worst	Average	Worst	Average	Worst
Black box	$O(mn+m)$	$O(mn+m)$	$O(mn+m)$	$O(mn+m)$	$O(mn+m)$	$O(mn+m)$
Embedded	$O(m^2n+m^2)$	$O(m^2n+m^2)$	$O(mn+m^2)$	$O(mn+m^2)$	$O(m^2s+mn-ms+m^2)$	$O(m^2n+m^2)$

- Testing: The time complexity for classifying an instance is $O(\frac{4}{5}m\log_2\frac{4m}{5})$ because the classifier distance matrix D facilitates us in finding its k closest neighbors by sorting the distance vector of D; in each iteration, $(n-s)$ classifier distance matrices are evaluated for each of $\frac{1}{5}m$ test instances, and the above process is repeated five times because of the fivefold cross-validation. Thus, the time complexity is $O(m^2(ns+s-s^2)\log_2 m)$, and the worst case is $O(m^2n\log_2 m)$.

(b) IWSS:

- Training: For each of the n iterations, we must calculate an attribute distance matrix and a classifier distance matrix with a time complexity of $O(m^2)$; thus, the training time complexity is $O(m^2n)$.
- Testing: There are $\frac{1}{5}m$ test instances; the time complexity for classifying an instance is $O(\frac{4}{5}m\log_2\frac{4m}{5})$, and the time of the iterations is n; the above process is repeated five times because of the fivefold cross-validation. Thus, the time complexity is $O(\frac{4}{5}m\log_2\frac{4m}{5}*n*\frac{m}{5}*5)=O(m^2n\log_2 m)$.

(c) IWSSr:

- Training: Slightly different from IWSS, IWSSr must calculate $(s+1)$ classifier distance matrices rather than one. Thus, the time complexity is $O(m^2n(s+1))$.
- Testing: For each iteration, we must evaluate s additional candidates; thus, the test time complexity is $O(\frac{4}{5}m\log_2\frac{4m}{5}*n*(s+1)*5*\frac{m}{5})=O(m^2n(s+1)\log_2 m)$. The worst is $O(m^2n^2\log_2 m)$.

A summary of the training and test time complexities as well as the total time complexity for the three methods is presented in Table 6. Although the training time complexity for the KNN-embedded case increases, the test complexity is greatly reduced, which leads to a reduction in the total time complexity. We observed that KNN-embedded methods are approximately $s/\log_2 m$ times faster than the black box cases for an average case and at least $n/\log_2 m$ times faster in the worst case, which demonstrates that a larger number of features selected is associated with a greater reduction in the time cost.

5.3. Space complexity analysis

5.3.1. Black box case

Because the experimental data with m instances, n attributes and one target variable needs to be loaded into memory and there is no KNN classifier to be trained, the space complexity is $O(m*(n+1))=O(mn+m)$ for SFS, IWSS and IWSSr.

5.3.2. KNN-embedded case

We are required to not only load the experimental data into memory but also maintain a classifier distance matrix and the corresponding attribute distance matrices. Therefore, the total space complexity arises mainly from these three parts.

For SFS, the space complexities are $O(mn+m)$ for loading experimental data into memory, $O(m(m-1))$ for storing the classifier distance matrix and $O(nm(m-1))$ for storing the attribute distance matrices. Therefore, the total space complexity is $O(m^2n+m^2)$, as determined by adding the three complexities together.

For IWSS, IWSS is not required to maintain the attribute distance matrix and instead maintains the classifier distance matrix; thus, the space complexity is $O((m+mn)+m(m-1))=O(m^2+mn)$.

For IWSSr, an additional s attribute distance matrices are required to be stored compared with IWSS because of the replacement operation; thus, the space complexity is $O(m^2+mn+m^2s-ms)$, and the worst case is $O(m^2n+m^2)$ when all of the features are evaluated $(n=s)$.

Table 7 summarizes the space complexity for both the average case and the worst case. The results show that, even for the worst case in SFS, the difference in the space complexity between the black box case and the KNN-embedded case is very small (less than a factor of m), as is the case for the IWSSr method. For IWSS, the space complexity for the embedded case is $O(mn+m^2)$ compared with $O(mn+m)$ for the black box case, and the additional space cost is negligible because $m \ll n$. For example, for the case of the *Ovarian* dataset, which has the highest dimensions (15,154 genes) and the largest number of samples (253 samples) in our experiment, the space costs obtained assuming that 8 bytes are required to code a double are 29.7 MB for SFS, 29.7 MB for IWSS and 34.6 MB for IWSSr in the KNN-embedded case compared to 14.9 MB for the black box case. The space complexity analysis demonstrates that the extra space cost in the RAM memory is affordable in current practices.

6. Conclusions

In this study, we proposed an approach for accelerating wrapper-based feature subset selection methods with an embedded

KNN classifier. The time cost in evaluating the quality of a candidate feature arises primarily from the inner fivefold cross-validation when using the KNN classifier as a black box. Considering this, we proposed the construction and dynamic maintenance of a classifier distance matrix (which consists of the distance between instances projected over the selected feature subset) rather than recalculation of the distance starting from scratch each time when a new feature is considered. This approach thus can greatly speed up the evaluation process and reduce the actual running time cost by avoiding massively repetitive calculations. Also, the proposed approach can apply to accelerating three types of feature selection methods, including wrapper methods with SFS, IWSS and IWSSr. Since the feature selection procedure and the criteria to include a candidate feature of the proposed approach are not different from the original approach, it is guaranteed that the proposed methods achieve the same feature subset as the original ones. To show the effectiveness of wrapper-based SFS, IWSS and IWSSr methods in selecting informative features, experiments were first conducted on eight publicly available microarray datasets. In comparison with the well-performing state-of-the-art feature selection method FCBF, the wrapper method with KNN outperforms FCBF in terms of classification accuracy and the size of the finally selected features. To demonstrate the performance gain in terms of time cost reduction, we then analyzed the theoretical time complexity and conducted an experimental study on the eight publicly available microarray datasets to show the actual time cost for both the black box case and the KNN-embedded case. The theoretical analysis and experimental results demonstrated the efficiency of the proposed approach in terms of running time without degrading the accuracy. In addition, a space complexity analysis showed that the additional space overhead is clearly affordable in practice when handling gene expression profiles.

Notably, in our study, the squared Euclidean distance rather than the Euclidean distance is stored in the classifier distance matrix to save on the computational cost. Because distance is a non-negative metric, the squared Euclidean distance and Euclidean distance are equal for measuring the relative distance between the test instance and the training instances, which guarantees obtaining the same feature subset. If the Euclidean distance is stored in the classifier distance matrix, we would need to first square the distance and then add it to the attribute distance matrix to obtain a candidate classifier distance matrix for a distance comparison. Each time after selecting a new feature, we first calculate the square root of the distance and then store it in the classifier distance matrix. Obviously, the latter performs additional calculations and is more time-consuming compared with the former.

Furthermore, compared with the case of considering KNN as a black box, the space complexity of our proposed method is $O(m^2n + m^2)$ for SFS, $O(mn + m^2)$ for IWSS and $O(m^2n + m^2)$ for IWSSr. Typically, if we maintain only a temporary attribute distance matrix rather than keeping the attribute distance matrices for all of the features, the space complexities of SFS and IWSSr are equal to that of IWSS, $O(mn + m^2)$. In handling gene expression profiles with high dimensionality (thousands of genes) and small sample sizes (as low as tens of samples), the space complexity of the proposed method is approximately equal to that of the black box case, i.e., $O(mn + m)$, which indicates that the additional space cost of the proposed method is quite small and can be easily met by today's computers for gene expression profile analysis. In handling data with ultra-large dimensionality and samples, the classifier distance matrix may not fit into the memory. Then, we can turn to the distributed computing paradigm, such as the MapReduce Framework, to divide the distance matrix into several small parts by row or column and store them on distributed hosts [45]. We would then use Map operations to calculate the attribute distance matrix and the candidate classifier distance matrix and use Reduce operations to decide whether to select a candidate feature and update these matrices in parallel [46]. In our future research, we plan to study other search strategies, such as sequential backward selection and sequential floating selection, as well as to explore other learning algorithms that have similar properties.

Acknowledgments

This work was supported partially by the "111 Project" of the Ministry of Education and State Administration of Foreign Experts Affairs (No. B14025), the International S&T Cooperation Program of China (No. 2014DFA11310), the Major Project of the Natural Science Foundation for Anhui Province Higher Education (No. KJ2011ZD06), the Natural Science Foundation of China (Nos. 61472057, 61305064, 51274078), and the "University Featured Project" of the Ministry of Education (No. TS2013HFGY031). Aiguo Wang was a visiting Ph.D. student at the Center for Biomedical Informatics at Harvard Medical School who was sponsored by the China Scholarship Council. The authors are very grateful to the anonymous reviewers for their constructive comments and suggestions for the improvement of this research.

References

[1] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, et al., Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (5439) (1999) 531–537.
[2] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.
[3] G. Piatetsky-Shapiro, P. Tamayo, Microarray data mining: facing the challenges, ACM SIGKDD Explor. Newslett. 5 (2) (2003) 1–5.
[4] J. Hua, W.D. Tembe, E.R. Dougherty, Performance of feature-selection methods in the classification of high-dimension data, Pattern Recogn. 42 (3) (2009) 409–424.
[5] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, An ensemble of filters and classifiers for microarray data classification, Pattern Recogn. 45 (1) (2012) 531–539.
[6] A. Jain, R. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (1) (2000) 4–37.
[7] I.A. Gheyas, L.S. Smith, Feature subset selection in large dimensionality domains, Pattern Recogn. 43 (1) (2010) 5–13.
[8] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (1–3) (2002) 389–422.
[9] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. (2003) 1157–1182.
[10] D. Koller, M. Sahami, Toward optimal feature selection, 1996.
[11] M. Dash, H. Liu, Consistency-based search in feature selection, Artif. Intell. 151 (1) (2003) 155–176.
[12] W.J. You, Z.J. Yang, G.L. Ji, PLS-based recursive feature elimination for high-dimensional small sample, Knowl.-Based Syst. 55 (2014) 15–28.
[13] R. Kohavi, H. George, Wrappers for feature subset selection, Artif. Intell. 97 (1) (1997) 273–324.
[14] I. Inza, P. Larrañaga, R. Blanco, A.J. Cerrolaza, Filter versus wrapper gene selection approaches in DNA microarray domains, Artif. Intell. Med. 31 (2) (2004) 91–103.
[15] J. Gama, R. Rocha, P. Medas, Accurate decision trees for mining high-speed data streams, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003.
[16] P. Bermejo, J. Gámez, J. Puerta, Speeding up incremental wrapper feature subset selection with Naive Bayes classifier, Knowl.-Based Syst. 55 (2014) 140–147.
[17] P. Bermejo, L. Ossa, J. Gámez, J. Puerta, Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking, Knowl.-Based Syst. 25 (1) (2012) 35–44.
[18] M. Gutlein, E. Frank, M. Hall, A. Karwath, Large-scale attribute selection using wrappers, in: IEEE Symposium on Computational Intelligence and Data Mining, CIDM'09, IEEE, 2009.
[19] A. Wang, N. An, G. Chen, L. Li, G. Alterovitz, Accelerating incremental wrapper based gene selection with *K*-Nearest-Neighbor, in: IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE, 2014, pp. 21–23.
[20] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66.
[21] P. Langley, W. Iba, Average-case analysis of a nearest neighbor algorithm, in: IJCAI, 1993.
[22] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inform. Theor. 13 (1) (1967) 21–27.
[23] E. Xing, M. Jordan, R. Karp, Feature selection for high-dimensional genomic microarray data, in: ICML, vol. 1, 2001, pp. 601–608.

[24] X. Sun, Y. Liu, M. Xu, H. Chen, J. Han Sun, Feature selection using dynamic weights for classification, Knowl.-Based Syst. 37 (2013) 541–549.
[25] K. Moorthy, M. Mohamad, Random forest for gene selection and microarray data classification, Bioinformation 7 (3) (2011) 142.
[26] H. Liu, L. Liu, H. Zhang, Ensemble gene selection for cancer classification, Pattern Recogn. 43 (8) (2010) 2763–2772.
[27] S. Li, E. Harner, D. Adjeroh, Random KNN feature selection – a fast and stable alternative to Random Forests, BMC Bioinformatics 12 (1) (2011) 450.
[28] G. Guo, D. Neagu, M. Cronin, Using kNN model for automatic feature selection, in: Pattern Recognition and Data Mining, Springer, Berlin, Heidelberg, 2005, pp. 410–419.
[29] H.L. Chen, B. Yang, G. Wang, J. Liu, X. Xu, S.J. Wang, D.Y. Liu, A novel bankruptcy prediction model based on an adaptive fuzzy *k*-nearest neighbor method, Knowl.-Based Syst. 24 (8) (2011) 1348–1359.
[30] M. Dash, H. Liu, Feature selection for classification, Intell. Data Anal. 1 (3) (1997) 131–156.
[31] R. Ruiz, J. Riquelme, J. Aguilar-Ruiz, Incremental wrapper-based gene selection from microarray data for cancer classification, Pattern Recogn. 39 (12) (2006) 2383–2392.
[32] P. Bermejo, J. Gámez, J. Puerta, Incremental wrapper-based subset selection with replacement: an advantageous alternative to sequential forward selection, in: IEEE Symposium on Computational Intelligence and Data Mining, CIDM'09, IEEE, 2009.
[33] P. Bermejo, J. Gámez, J. Puerta, On incremental wrapper-based attribute selection: experimental analysis of the relevance criteria, in: Proceedings of IPMU'08, 2008, pp. 638–645.
[34] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, A.J. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, Proc. Natl. Acad. Sci. USA 96 (12) (1999) 6745–6750.
[35] S.L. Pomeroy, P. Tamayo, M. Gaasenbeek, L.M. Sturla, M. Angelo, M.E. McLaughlin, T.R. Golub, Prediction of central nervous system embryonal tumour outcome based on gene expression, Nature 415 (6870) (2002) 436–442.
[36] D. Singh, P.G. Febbo, K. Ross, D.G. Jackson, J. Manola, C. Ladd, P. Tamayo, et al., Gene expression correlates of clinical prostate cancer behavior, Cancer Cell 1 (2) (2002) 203–209.
[37] J. Khan, J.S. Wei, M. Ringner, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, et al., Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks, Nat. Med. 7 (6) (2001) 673–679.
[38] M.A. Shipp, K.N. Ross, P. Tamayo, A.P. Weng, J.L. Kutok, R. Aguiar, M. Gaasenbeek, et al., Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning, Nat. Med. 8 (1) (2002) 68–74.
[39] E.F. Petricoin III, A.M. Ardekani, B.A. Hitt, P.J. Levine, V.A. Fusaro, S.M. Steinberg, L.A. Liotta, Use of proteomic patterns in serum to identify ovarian cancer, Lancet 359 (9306) (2002) 572–577.
[40] M. Robnik-Šikonja, I. Kononenko, Theoretical and empirical analysis of ReliefF and RReliefF, Mach. Learn. 53 (1–2) (2003) 23–69.
[41] U.M. Braga-Neto, E.R. Dougherty, Is cross-validation valid for small-sample microarray classification?, Bioinformatics 20 (3) (2004) 374–380
[42] S.K. Singhi, H. Liu, Feature subset selection bias for classification learning, in: V. Anbu (Ed.), Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006.
[43] L. Yu, H. Liu, Feature selection for high-dimensional data: a fast correlation-based filter solution, in: ICML, vol. 3, 2003.
[44] L. Yu, H. Liu, Efficient feature selection via analysis of relevance and redundancy, J. Mach. Learn. Res. 5 (2004) 1205–1224.
[45] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[46] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.H. Bae, J. Qiu, G. Fox, Twister: a runtime for iterative mapreduce, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ACM, 2010.