Pattern Recognition 45 (2012) 2672–2689

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Automatic recommendation of classification algorithms based on data set characteristics

Qinbao Song *, Guangtao Wang, Chao Wang

Department of Computer Science & Technology, Xi'an Jiaotong University, China

ARTICLE INFO

Article history:
Received 14 September 2011
Received in revised form
22 November 2011
Accepted 27 December 2011
Available online 11 January 2012

Keywords:
Classification algorithm automatic recommendation
Classification
Data set characteristics extraction
Algorithm performance
k-Nearest Neighbors

ABSTRACT

Choosing appropriate classification algorithms for a given data set is very important and useful in practice but also is full of challenges. In this paper, a method of recommending classification algorithms is proposed. Firstly the feature vectors of data sets are extracted using a novel method and the performance of classification algorithms on the data sets is evaluated. Then the feature vector of a new data set is extracted, and its k nearest data sets are identified. Afterwards, the classification algorithms of the nearest data sets are recommended to the new data set. The proposed data set feature extraction method uses structural and statistical information to characterize data sets, which is quite different from the existing methods. To evaluate the performance of the proposed classification algorithm recommendation method and the data set feature extraction method, extensive experiments with the 17 different types of classification algorithms, the three different types of data set characterization methods and all possible numbers of the nearest data sets are conducted upon the 84 publicly available UCI data sets. The results indicate that the proposed method is effective and can be used in practice.

1. Introduction

As an important machine learning area, classification has been extensively studied, resulting in a large number and different types of algorithms proposed, including tree-based (e.g., ID3 [1], C4.5 [2] and CART [3]), probability-based (e.g., Naive Bayes [4] and AODE [5]), rule-based (e.g., RIPPER [6], CN2 [7] and PART [8]), and association-based (e.g., CBA [9], CMAR [10], MCAR [11] and ACWV [12]). Consequently, many alternative choices exist for a given specific classification problem.

Expectedly or not, the reported performance of these algorithms is quite mixed. For example, algorithm Alg_1 performs better on some data than algorithm Alg_2, but the latter may perform better than the former on other data. In the study by Weiss and Kapouleas [13], in particular, it was found that in terms of accuracy, Back-Propagation neural networks outperformed decision tree method on Iris and Appendicitis data but was defeated by the latter on Breast cancer and Thyroid data. Shavlik et al. [14] observed that the accuracies of Back-Propagation neural networks were greater than those of the decision tree method on Soybean, Audiology, Heart disease and NETtalk 'A' data but less than those of the latter on Chess and NETtalk (808 wds) data. Duin's [15] results show that k-NN is more accurate than ANN on Iris, Imox, 80X, Glass and Sonar data, but not as accurate as the latter on Blood and Dnorm data. Moreover, thorough experimental results by Ali and Smith [16] tell us that: (i) IBK is more accurate than PART, Naive Bayes and OneR on No. 32, 54 and 66 data sets, and less accurate than these algorithms on No. 64, 45 and 33 data sets; (ii) OneR is more accurate than SVM, PART and Naive Bayes on No. 40, 18 and 27 data sets, and less accurate than these algorithms on No. 52, 80 and 73 data sets; (iii) PART is more accurate than SVM, Naive Bayes and C4.5 on No. 42, 74 and 43 data sets, and is less than these algorithms on No. 51, 34 and 48 data sets.

The above is just a brief account on the mixed performance situation of the various classification algorithms. In particular, it reveals that no single algorithm can perform uniformly well over all data sets. This is consistent with Wolpert and Macready's well-known No Free Lunch theorem [17]: "If algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A." This situation makes the selection of a proper algorithm for a specific classification problem very difficult, as the choice of which algorithm(s) to use depends on the data set(s) at hand. As pointed out in [18], a system that can provide such recommendations would be very useful. The cross-validation strategy was used to perform an exhaustive search through the space of candidate classification algorithms. However, as the number of alternatives increases, the time required to search for the best

* Corresponding author. Tel.: +86 29 82668645; fax: +86 29 82668971.
E-mail addresses: qbsong@mail.xjtu.edu.cn (Q. Song), gt.wang@stu.xjtu.edu.cn (G. Wang).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.12.025

algorithm may become impractical due to the computational expense of performing a cross-validation [19]. There have been other two kinds of work conducted subsequently: the development of new algorithms, and the comparison studies of existing different algorithms. Unfortunately, new algorithms still cannot guarantee to perform well on all data sets. On the other hand, while the comparison studies may offer some seemingly useful knowledge about the algorithms, the extendibility of such empirical knowledge is very limited, making it difficult for users to interpret and use.

Actually, what lies behind the mixed performance situation is that, apart from using different experimental methods, different data sets constitute an important factor affecting the performance of a classification algorithm. In other words, there exist some intrinsic relationship between classification algorithm performance and data set characteristics. For example, Michie et al. [20] drew some general conclusions from their extensive experimental results that "decision trees do well on credit datasets, *k*-nearest neighbors excel on problems concerning images, and applications with misclassification cost matrices require algorithms explicitly incorporating them". According to Quinlan [21], "there are certain classification tasks that lend themselves to methods of one or the other type", and "one of the most rewarding enterprises for the next few years will be to extract from this data some understanding of problem types and their relationship to classes of learning algorithms". All these concur that choosing a proper algorithm based on data characteristics can help a lot in practice.

More recently, Ali and Smith's study on the classification algorithm selection problem [16] confirms that "a more useful strategy is to gain an understanding of the dataset characteristics that enable different learning algorithms to perform well, and to use this knowledge to assist learning algorithm selection based on the characteristics of the dataset".

Moving in this direction, many research works have been done, including [16,19,22–28]. All these works, except for [19], firstly use some statistical and information theoretical measures to characterize data sets, and then try to capture the relationship between the measured data set characteristics and the classification algorithm performance by decision tree, instance based rule induction methods or regression models. Finally, these relationships are used to make recommendations as to which classification algorithm could be used for a given data set. Recently, Ho and Basu [29] studied a number of measures that characterize the difficulty of a classification problem based on the geometrical complexity of the class boundary. Afterward, Bernadó-Mansilla and Ho [30] employed a subset of these measures to characterize data sets, and used the relationship between the measures and the performance of classifiers to assist the determination of appropriate algorithms for a new classification problem.

However, different researchers used different measures, and we do not know which of these measures are essential and which are less useful. Moreover, the different choices of these statistical and information theoretical measures may result in different results.

In contrast, our proposed classification algorithm recommendation method[1] deviates from these works in the ways of characterizing a data set and dealing with similar data sets. More specifically, we use structural and statistical information based feature vector (induced from Tatti's work [31]) to characterize each data set, and adopt the *k*-NN method to identify the *k* nearest data sets to the new data set; then the algorithms with better performance on the nearest data sets are recommended for classifying the new data set. The experimental results on the 84 UCI data sets [32] and the 17 different types of classification algorithms show the effectiveness of the proposed method.

The remainder of this paper is organized as follows: Section 2 provides a brief review of the related work. Section 3 presents the proposed method in detail. Section 4 explains the data set characterization method. Section 5 provides the experimental process and the corresponding results. The conclusion of our work is given in Section 6.

2. Related work

The most explored methods of algorithm recommendation involve the use of knowledge on the performance of algorithms and/or the relationship with and the characteristics of data sets. These methods can be either of theoretical or of experimental origin.

Brodley [19] presented a heuristic approach that applies knowledge about the representational biases of a set of learning algorithms to conduct the search from the available algorithm space to find the one that produces the best classifier automatically. This approach tries to capture the knowledge of domain experts concerning the applicability of certain classification algorithms.

Aha [22] described an empirical method for generalizing results from case studies. The method uses these results to focus on a location in some database-characterization space where known performance differences exist, and generates rules characterizing when these differences occur. That is, these rules describe when some algorithms significantly outperform others on some dependent measures. Brazdil et al. [23] characterized various data sets with different statistical and information theoretic measures. After determining which algorithms are applicable to these data sets, both data set characteristics and algorithm applicability information are used by C4.5 to generate a set of rules which, finally, is used to give recommendations as which classification algorithm could be used with a given data set. Although the recommendation is not guaranteed to give the best possible advice, it indeed helps narrow down user choices. Ali and Smith [16] proposed a rule-based classifier selection approach, which is based on the prior knowledge on problem characteristics and the empirical results generated on 100 data sets with eight classification algorithms; the decision tree induction algorithm C5.0 is applied to the empirical results to generate rules, which finally are used to describe which types of algorithms are suited to addressing which types of classification problems. In this method, data set characteristics are measured by following that of Smith et al. [25,26].

Gama and Brazdil [24] proposed using linear regression models, piecewise linear models and instances models to capture the information concerning applicability. These models are generated automatically on the basis of data set characteristics. Brazdil et al. [27] presented a meta-learning method to support selection of candidate classification algorithms. It characterized the data sets with the statistical information and entropy information. The most similar historical data sets are then identified for a new data set with a *k*-NN algorithm. The performance of the candidate algorithms on those historical data sets is graded and ranked to generate a recommendation. Different from the previously introduced methods, this method does not generate rules concerning the applicability of a classification algorithms. Kalousis and Gama [28] aimed to find functions that map data sets to algorithm performance. To this end, meta-learning uses a set of attributes to represent the characteristics of the learning tasks, and searches for the correlations between these attributes and the

[1] The corresponding software can be obtained via qbsong@mail.xjtu.edu.cn.

performance of learning algorithms. Bernadó-Mansilla and Ho [30] utilized a set of measures to characterize the complexity of classification problems, and analyzed the interaction between the problem complexity and the performance of classification algorithms. Then this interaction was employed to select appropriate algorithms.

The above studied methods, except for [19], aimed to capture the relationship between the measured data set characteristics and the performance of the algorithms. Our method is different from these works in the ways of characterizing a data set and dealing with similar data sets, as to be introduced in the next section.

3. Classification algorithm recommendation

3.1. General view of the method

As there is an intrinsic relationship between data set feature[2] and classification algorithm performance on the data sets (see, e.g., [16,20–28]), we assume that if the features of some data sets are similar, then the performance of classification algorithms on these data sets is similar as well. That is, if an algorithm performs well on a data set, it also should perform well on another data set that is similar to the former one. More generally, if an algorithm performs well on a certain group G of data sets, it is reasonable to recommend this algorithm to a new data set that is similar to any of the data sets within G.

Based on the above assumption and analysis, we propose an automatic recommendation method for classification algorithms through mining the relationships between data set features and algorithm performance. In the proposed method, for each existing historical data set, its feature vector is extracted and the applicable classification algorithms are identified firstly. When a new data set is encountered, its k nearest data sets are recognized and their corresponding applicable classifiers are recommended to the new data set. After that, the feature vector and the applicable classification algorithms of the new data set are recorded in a "relationship" database for future recommendations.

As shown in Fig. 1, our proposed method consists of constructing a database of data set-applicable algorithm relationships, recommending classification algorithm(s) for a new data set, and updating the relationship database, as discussed below:

1. *Data set-applicable classifier relationship database construction*: In our proposed method, the relationship between a data set and its applicable classification algorithms is very important. In order to obtain this kind of relationship, for each historical data set, its feature vector is extracted (see Section 4 for details), and the applicable classification algorithms are identified (Section 3.2 provides the details), both of them are recorded as a tuple in the relationship database.
2. *Classifier recommendation for a new data set*: One underlying reason of data sets being similar is that they have something in common. For example, they may come from the same problem domain and/or there may be some classification algorithms that perform well on most of them. Thus, if a new data set is closer to some data sets than to any others, the classification algorithms with higher performance on these data sets should also perform well on the new data set. Furthermore, they can be recommended for classifying the new data set. Section 3.3 provides the details of this task.
3. *Relationship database updating*: The classification algorithm recommendation result for a new data set reveals the relationship between the new data set and the recommended applicable classifiers, the information of which is kept in the data set-applicable classifier relationship database. The larger the database is, the more effective the future recommendation will be. Thus, after each recommendation, the relationship between the new data set and the recommended applicable classifiers is used to update the data set-applicable classifier relationship database as well.

The classification algorithm recommendation process is described by the procedure *ClassifierRecommendation*.

Procedure. ClassifierRecommendation.

inputs: *data* - a new data set
DATA$=\{D_1,D_2,\ldots,D_n\}$
ALG$=\{Alg_1,Alg_2,\ldots,Alg_m\}$
k – the predefined number of the nearest neighbors
RDatabase – the feature vector – applicable classifier relationship database.
output: *RecAlg* – Recommended Classifiers for *data*

```
for i=1 to n do
    VECTOR[i] = FeatureExtraction(D_i); //see Section 4
    AppAlgs[i] = ApplicableAlgIndentification(D_i, ALG);
    //see Section 3.2
    insert < VECTOR[i], AppAlgs[i] > into RDatabase
vector = FeatureExtraction(data)
Neighbors = NeighborRecognition(k, vector, VECTOR)
CandAlgs = AppAlgs of the Neighbors
RecAlg = Recommendation(Neighbors, CandAlgs)
update RDatabase with < vector, RecAlg >
return RecAlg
```

3.2. Applicable classification algorithm identification for historical data sets

The purpose of the applicable classification algorithm identification is to find out the classifiers that perform well on the existing historical data sets and get ready for the construction of the data set-applicable classifier relationship database.

The method of applicable classification algorithm identification is straightforward: for each historical data set, apply each of the available classifiers to it respectively. Then the performance of the classifiers on each data set is evaluated, and the applicable classifiers, which are the classification algorithms that perform better than the rest, for that data set are identified.

When evaluating the performance of a learning algorithm *Alg* on a given data set D, the multicriteria evaluation measure $P_{Alg,D}$ adopted from Brazdil et al. [27], which combines the information about the classification accuracy and total execution time[3] of the learning algorithm, is modified to fit our purpose. Particularly, instead of using the measure to capture the relative performance of any two algorithms for a given compromise between the criteria, $P_{Alg,D}$ is used to reveal the performance of the learning algorithm *Alg* on data set D, that is

$$P_{Alg,D}=\frac{Acc_{Alg,D}}{1+\alpha\times\log(RTime_{Alg,D})}, \quad (1)$$

[2] As the information we extracted from a data set is quite different from that of the existing work, we use "feature" instead of characteristics hereafter.

[3] The execution time includes both the training time and the testing time.

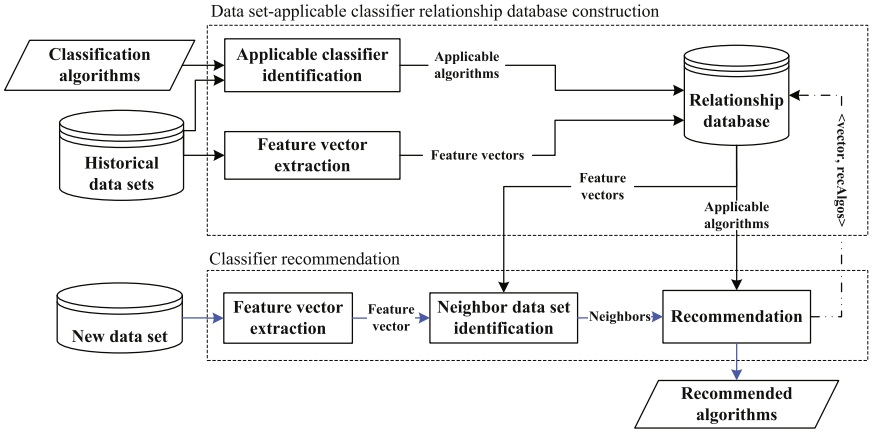

Fig. 1. General view of the proposed classification algorithm recommendation method.

where $Acc_{Alg,D}$ and $RTime_{Alg,D}$ represent the classification accuracy and run time of algorithm *Alg* on data set *D*, respectively, and α represents user-defined relative importance of accuracy and time. In particular, this parameter represents the amount of accuracy the user is willing to trade for a 10 times speedup or slowdown. For example, $\alpha = 10\%$ means that the user is willing to trade 10% of accuracy for 10 times speedup/slowdown.

In order to obtain stable performance and make the best use of the data, a "ten times 10-fold cross-validation" is suggested. That is, when evaluating the performance of a given classification algorithm on a data set, the 10-fold cross-validation is repeated 10 times; each time we randomize the order of the instances of the data set. This is because many of the algorithms exhibit order effects [33], in that certain ordering may dramatically improve of degrade performance. Randomizing the order of the inputs can help diminish the order effects.

The performance of a classification algorithm on each data set is obtained with the procedure *PerformanceEvaluation* presented next page.

Procedure. PerformanceEvaluation.

inputs: *data* - the given data set
classifier - a given classification algorithm;
output: *Performance* - the performance when applying *classifier* on *data*;
M = 10; FOLDS = 10
Performance = 0
for $m = 1$ **to M do**
  randomized order from *data*
  generate FOLDS bins from *data*
  for $i = 1$ **to** FOLDS **do**
    TestData = bin[*i*]
    TrainingData = *data*-TestData
    (*predictor*,$\text{Time}_{training}$) = *classifier*(TrainingData)
    (Accuracy[*i*],Time_{test}) = apply *predictor* to TestData
    $\text{Time}[i] = \text{Time}_{training} + \text{Time}_{test}$
    Performance = *Performance*
    + PerformanceCombination(Accuracy[*i*],Time[*i*])
    //Computing performance with Eq. (1)
Performance = *Performance*/(M × FOLDS)

Once the performance of all available classifiers is obtained, we can determine the applicable classification algorithms for a given data set. The classification algorithm with the highest performance is the first applicable classification algorithm for the data set, and it is commonly considered to be the real best algorithm for that data set. Moreover, any algorithm whose performance falls within the interval $[P_{BestAlg,D} - \beta \times Stdev, P_{BestAlg,D}]$ is considered as applicable as well. Here, $P_{BestAlg,D}$ is the performance of the best algorithm *BestAlg* on data set *D*, *Stdev* viewed as the margin of tolerance is the standard deviation of the performance of all the candidate algorithms on the same data set, and the value of β determines the size of the interval. Clearly, the larger the value of β is, the higher the confidence that the best algorithm will have in the interval.

Next, the procedure *ApplicableAlgIndentification* next page provides the details of the applicable classification algorithm identification for a given data set.

Procedure. ApplicableAlgIdentification.

inputs: *data* = the given data set
ALG = {Alg_1,Alg_2, …,Alg_n};
output: *AppAlgs* - Applicable classifiers for *data*;
for $i = 1$ **to** n **do**
  Performance[*i*] = PerformanceEvaluation(*data*,Alg_i);
P_{Best} = Max(*Performance*);
Stdev = Std(*Performance*);
for $i = 1$ **to** n **do**
  if *Performance*[*i*] ≥ ($P_{Best} - \beta \times Stdev$) **then**
    AppAlgs = *AppAlgs* ∪ Alg_i;
return AppAlgs

3.3. Classification algorithm recommendation for a new data set

With the help of the data set-applicable classifier relationship database, and taking into account our previously stated assumption that classification algorithms with higher performance on some data sets should also perform well on a data set that is similar to these data sets, we can recommend classification algorithms for a new data set with a *k*-NN (Nearest Neighbors) based recommendation method.

The reasons why the *k*-NN method is employed to recommend algorithms are as follows:

- Our proposed data set feature vector is based on Tatti's work [31] that deals with similar data sets. This means it is more suitable to identify similar data sets with this feature vector. Meanwhile, *k*-NN is widely used to search for similar

neighbors. Thus, it is straightforward to use k-NN to choose applicable algorithms for a problem at hand.

- There might be multiple appropriate algorithms for a data set. Thus, the recommendation results could be a set of algorithms rather than a single one. When applying k-NN based recommendation method, once obtaining the most similar data sets for the data set at hand, we can easily get a set of algorithms from the applicable algorithms of the similar data sets. Moreover, we can further rank these algorithms based on the similarities between these neighbors and the data set at hand, and give a more appropriate recommendation results.
  The other learning methods, such as SVM and C4.5, can be used to build recommendation as well. However, the problems addressed by these methods are usually single labeled. Thus, they just recommend a single algorithm once a time. In order to get multiple applicable algorithms, we need to construct multiple recommendation models. What's more, it is difficult to rank these recommended algorithms. Therefore, the k-NN based recommendation method is more efficient and flexible.
- The k-NN based recommendation method is easy to understand and carry out.

The recommendation method consists of the following two steps: (i) identifying the nearest neighbor data sets for a new data set, and (ii) recommending classification algorithms for the new data set with the help of the nearest neighbors.

Procedure. NeighborRecognition.

inputs: k - the number of neighbors ;
vector - the feature vector of the new data set ;
VECTOR - the feature vectors of the historical data sets.
output: *Neighbors* - the neighbor data sets for the new data set;

```
1 i = 1
2 for each v ∈ VECTOR do
3 |  DistanceTable[i] = the distance between vector and v;
4 |  i = i + 1;
5 sort DistanceTable in ascending order;
6 Neighbors = the top k data sets of DistanceTable;
7 return Neighbors
```

3.3.1. The nearest neighbor data set identification

The k-NN method is widely used to identify the k nearest neighbors of an instance among instances; the smaller the distances, the nearer they are. We adopt the method to pick out the k most similar data sets (i.e., the nearest neighbor data sets) for a given data set.

The k-NN method computes the distance between any two instances through the features describing them. Similarly, we work out the distances between two data sets according to the corresponding data set feature vectors (see Section 4 for details of the data set feature extraction method).

The nearest neighbor data set identification method involves (i) extracting the feature vector for the new data set, and (ii) identifying its k nearest neighbor data sets from the historical data sets according to their feature vectors. The procedure *NeighborRecognition* in the preceding page provides the details.

3.3.2. Classification algorithm recommendation with the nearest neighbors

After obtaining the k nearest neighbor data sets of a new data set, we determine which classification algorithms can be used to classify the new data set according to the data set-applicable classifier relationship database and the applicable classifiers for the k nearest neighbor data sets.

However, the applicable classifiers of the nearest neighbor data sets can be different. Thus, before recommending any classification algorithms to classify the new data set, we have to thoroughly evaluate the applicable classifiers of the nearest neighbor data sets first.

When evaluating the applicable classifiers, all of them are viewed as the candidate algorithms. Then the performance of these candidate algorithms is pair-wise compared and the winners are recommended to classify the new data set. For comparison, both the significant Win/Draw/Loss [34] record and the mean performance of a classification algorithm are used as the measures.

The significant Win/Draw/Loss record presents three values on a given measure, i.e. the numbers of data sets for which algorithm Alg_1 obtains better, equal, and worse performance than algorithm Alg_2, respectively. The measure can be classification accuracy, running time, or performance. In our context, performance is used. A win or loss is only counted if the difference in the given measure is determined to be significant at the 0.05 level by a paired sign test. If the sign test result is significantly low, then it is reasonable to conclude unlikely the outcome was obtained by chance, hence the record (of win or loss) represents a systematic underlying advantage or disadvantage to one of the algorithms with respect to the type of domains on which they have been tested.

After obtaining the significant Win/Draw/Loss records and the mean performance across all the neighbor data sets for all candidate algorithms, we are able to identify the r ($r \in N$) applicable classification algorithms from the candidate list $CL = \{Alg_1, Alg_2, \ldots, Alg_n\}$ as follows:

1. For each $i \neq j$ ($i,j = 1,2,3,\ldots,n$), compare the significant Win/Draw/Loss counts between algorithms Alg_i and Alg_j. If Count(Win, $Alg_i : Alg_j$) > Count(Loss, $Alg_i : Alg_j$) holds[4] for each pair of $\{Alg_i, Alg_j\}$, we say that Alg_i defeats all the other candidates, and then algorithm Alg_i is selected.
2. Otherwise, the results are conflicting, so candidates Alg_i, Alg_j and Alg_k are not defeated by each other although they all defeat the others. That is, there exist cases of Alg_i defeats Alg_j, Alg_j defeats Alg_k and Alg_k defeats Alg_i. In this situation, the corresponding average performance of those algorithms over all data sets is calculated, and the algorithm with the highest average performance is selected.
3. Move the selected algorithm from the candidate list CL into the ranking list RL. Repeat steps 1 and 2 r times.

The ranking list $RL = \{Alg'_1, Alg'_2, \ldots, Alg'_r\}$ ($r \in N$) can be used to recommend algorithms to a new classification problem. The recommendation process is detailed by procedure *Recommendation* next page.

Procedure. Recommendation.

inputs *Neighbors* - the neighbor data sets of the new data set;
ALG = $\{Alg_1, Alg_2, \ldots, Alg_n\}$;// ALG *is the candidate classifier list*;
output: *RecAlg* - the recommended classifiers;

```
1 Rounds = 3;
2 generate WDL for ALG on Neighbors; //
  WDL-Win/Draw/Loss record;
```

[4] Count(Win/Loss, Alg_i:Alg_j) denotes the count of algorithm Alg_i won/lost against Alg_j.

```
for i = 1 to Rounds do
    AL = algorithm(s) ∈ ALG with the least losses according to WDL;
    if |AL| = 1 then
        AlgApp = AL[1]
    else
        AL = algorithm(s) ∈ AL with the most wins according to WDL;
        if |AL| = 1 then
            AlgApp = AL[1];
        else
            AlgApp = algorithm(s) ∈ AL with the greatest mean performance;
    ALG = ALG−AlgApp;
    RecAlg[i] = AlgApp;
    update WDL by removing the row and column corresponding to AlgApp;
return RecAlg;
```

4. Data set feature extraction

Data set features are a noteworthy factor that affects the performance of a classification algorithm. If different data sets can be described by their own features, the relationship between data set features and classification algorithms' performance can thus be obtained.

In this section, we first introduce some basic concepts related to the data set feature, and then, elaborate the proposed feature extraction method. Finally, we illustrate the method with an example.

4.1. Basic concepts

The following basic concepts from Tatti's [31] work are adopted and refined to fit our purpose.

Binary data set: For a given data set, the types of attribute values could be varied. They can be nominal or numerical. A data set is a *binary data set* if and only if all its attribute values are just 0 or 1. A *binary data set* is denoted by D_B, and its attribute set is denoted by $A=\{a_1,a_2,\ldots,a_k\}$, where $k=|A|$ is the number of attributes.

An ordinary data set can be transformed into a *binary data set*. In the binary format, we only focus on whether an instance has some properties or not. For example, if an attribute value is 1, we say the related instances have the corresponding property, and vice versa. It does not concern what the property is. In this way, the statistics and the structural information of a data set could be evaluated.

Item set: Given a binary data set D_B, each of its attribute, a_i $(i=1,2,\ldots,k)$, is also called an item. A non-empty subset of the attribute set A is referred to as an *item set*. The value pattern of item sets can be viewed as features of a data set.

In order to more clearly describe a binary data set, we extend the concept of *item set* to *one-item set*, *two-item set* and *n-item set*. The *one-item set* I of a data set consists of single attribute items, i.e., $I=\{\{a_i\}|i=1,2,\ldots,k\}$. Correspondingly, the *two-item set* of a data set contains all the attribute pairs of the data set, and is denoted by $II=\{\{a_i,a_j\}|i,j=1,2,\ldots,k;i<j\}$. The *n-item set* could be easily derived from extending the concept of *two-item set*. The pattern information of *one-item set* explicitly tells the information of each individual attributes. In contrast, the *two-item set* tells the correlation information of the attribute pairs. They describe the different but complementary aspects of the data set. Because of the expensive computation cost of *n-item set*, our method only considered the *one-item set* set and *two-item set*.

Feature function: A *feature function* is defined as $S:\Omega\rightarrow R^N$ (N indicates the dimensionality of the range space of S), which maps a point in the sample space Ω to a real vector.

A feature function can be a parity function which maps a binary vector to a binary value [31]. Given an item set $B=\{a_{i_1},a_{i_2},\ldots,a_{i_L}\}$ and the sample space $\Omega=\{\omega|\omega=(\omega_1,\omega_2,\ldots,\omega_k),\omega_j=0,1;1\le j\le k\}$, the parity function $T_B(\omega)$ is defined as

$$T_B(\omega)=\omega_{i_1}\oplus\omega_{i_2}\oplus\cdots\oplus\omega_{i_L}, \tag{2}$$

where $\oplus$ is the *XOR* operator. $T_B(\omega)$ is 1 if and only if the number of active variables included in B is odd.

The frequency θ of T_B taken with respect to a data set D_B is the average of T_B values taken over the data set, that is, $\theta=(1/|D_B|)\sum_{\omega\in D_B}T_B(\omega)$, where ω is an instance in D_B. It is denoted by $T_B(D_B)$.

Thus, the feature vector of a data set could be defined according to the feature function and its frequency.

4.2. Data set feature extraction method

Inspired by Tatti's [31] work, we present in this section a feature extraction method for non-binary data sets.

Given an ordinary data set D and its attribute set $A=\{A_1,A_2,\ldots,A_k\}$, firstly, the data set D is transformed into the corresponding binary data set; then, the feature vector of the binary data set is extracted according to the item sets and the feature function; after that, all the feature vectors of different data sets are post-processed so as to obtain unified feature vectors.

4.2.1. Data set transformation

The data set transformation aims to transform a non-binary data set into a corresponding binary data set without losing semantic information. The nominal to binary transformation of attribute is widely used in the case of learning algorithms that cannot deal with multi-valued nominal attributes directly. So in our case, we take the data set as a whole, and transform all the attributes into binary format. The transformation process involves two steps as follows.

Step 1: *Distinct attribute value identification*.

For an ordinary data set D, each attribute has to be transformed into a group of new binary attributes. In order to facilitate the transformation, firstly we have to figure out how many distinct values each attribute has. This actually is the number of new attributes which are going to replace that attribute. Then the numbers of distinct values for all the attributes are accumulated and denoted by m, revealing how many different attribute values are there in the data set D and how many attributes will be in the corresponding binary data set.

The distinct values of an attribute A_i are encapsulated into a set C_{A_i}, which is referred to as the attribute value set of A_i.

For example, Fig. 2(a) shows an ordinary data set. From it we observe that the attribute A_2 has three distinct values: a_5, a_6, and a_7. These three values constitute the attribute set $C_{A_2}=\{a_5,a_6,a_7\}$ as shown in Fig. 2(b). Meanwhile, the number of all distinct values, m, is 9. This means that the dimensionality of the corresponding binary data set is 9.

Step 2: *Attribute and its value replacement*.

In this step, we replace each attribute $A_i\in A$ by its attribute value set C_{A_i}. All the C_{A_i} (for $i=1,2,\ldots,k$) constitute the attributes of the corresponding binary data set D_B.

For each instance $d\in D$, we then replace the value of A_i $(i=1,2,\ldots,k)$, which is denoted by $Value_{A_i}$, by the values of the

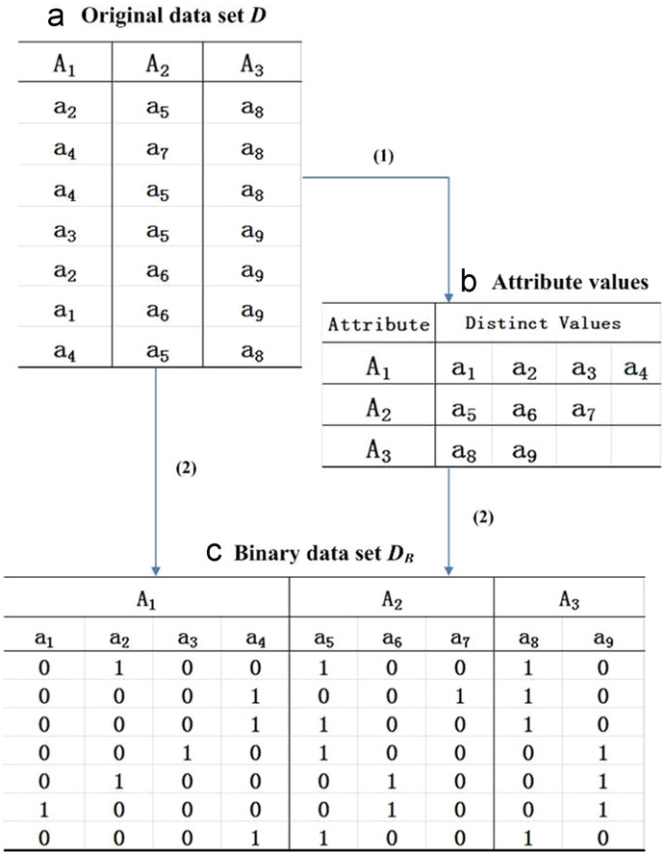

Fig. 2. Data set transformation.

attribute set C_{A_i}, denoted as $Value_{C_{A_i}}[j]$ $(j=1,2,\ldots,|C_{A_i}|)$, with

$$Value_{C_{A_i}}[j]=\begin{cases}1 & \text{if } Value_{A_i}=C_{A_i}[j]\\ 0 & \text{otherwise.}\end{cases}$$

For example, in Fig. 2(a), attribute A_2 is replaced by its attribute set $C_{A_2}=\{a_5,a_6,a_7\}$, which constitutes the three attributes of the corresponding binary data set as shown in Fig. 2(c). For the second instance in Fig. 2(a), $Value_{A_2}=\{a_7\}$ is replaced by the $Value_{C_{A_2}}=\{0,0,1\}$, which constitutes a part of the second instance as shown in Fig. 2(c).

4.2.2. Data set feature extraction

Data set feature extraction is based on the frequency of the feature function T_B with respect to the combination of the *one-item set I* and *two-item set II* for a given data set.

As we mentioned before in the common definition of the *two-item set*, $II=\{\{a_i,a_j\}|i,j=1,2,\ldots,k;i<j\}$, the attribute pair (a_i,a_j) comes from two different attributes. However, in the transformed data set, attributes a_i and a_j may belong to the same original attribute $A_p(p<k)$. Such type of combination makes no sense. As there would be no value pairs like (1, 1), this sort of item sets contain the same information as that of the *one-item sets*. Therefore, for the transformed data set, the *two-item set* is modified as $II=\{\{a_i,a_j\}|i,j=1,2,\ldots,k$ and $i<j;a_i\in C_{A_p},a_j\in C_{A_q},p,q=1,2,\ldots,k$ and $p\neq q\}$.

The features derived from V_I essentially capture the distribution of the values of a given attribute in a data set, while V_{II} gives the correlation between two features as the parity function is used. Therefore, for a given data set D_B, we define its feature vector as $V_{I\&II}$, where $V_{I\&II}$ is the combination of a *one-item set* V_I and a *two-item set* V_{II}. That is $V_{I\&II}=\{V_I,V_{II}\}$.

The member of V_I is the frequency of the feature function T with respect to a *one-item set* B_I over the data set D_B which is denoted as $T_{B_I}(D_B)=(1/|D_B|)\sum_{\omega\in D_B}T_{B_I}(\omega)$. The V_I is comprised of the frequencies of all one-item set elements $\{a_i\}(i=1,2,\ldots,k)$. Therefore

$$V_I(D_B)=\left(\frac{1}{|D_B|}\sum_{\omega\in D_B}T_{a_1}(\omega),\frac{1}{|D_B|}\sum_{\omega\in D_B}T_{a_2}(\omega),\ldots,\frac{1}{|D_B|}\sum_{\omega\in D_B}T_{a_k}(\omega)\right) \tag{3}$$

Similarly, a member of V_{II} is the frequency of the feature function T with respect to a two-item set B_{II} over the data set D_B, denoted as $T_{B_{II}}(D_B)=(1/|D_B|)\sum_{\omega\in D_B}T_{B_{II}}(\omega)$. The V_{II} is comprised of the frequencies of all two-items $\{a_i,a_j\}(i,j=1,2,\ldots,k;i<j)$. Therefore

$$V_{II}(D_B)=\left(\frac{1}{|D_B|}\sum_{\omega\in D_B}T_{a_{i_1},a_{j_1}}(\omega),\frac{1}{|D_B|}\sum_{\omega\in D_B}T_{a_{i_2},a_{j_2}}(\omega),\cdots,\frac{1}{|D_B|}\sum_{\omega\in D_B}T_{a_{i_L},a_{j_L}}(\omega)\right) \tag{4}$$

where $\{a_{i_n},a_{j_n}\}\in II$ and $T_{a_{i_n},a_{j_n}}(\omega)=\omega_{i_n}\oplus\omega_{j_n}$.

4.2.3. Data set feature unification

For different data sets, the number of attributes and the number of the distinct values for a specific attribute can be different. Therefore the lengths of the *one-item* feature vector V_I and the *two-item* feature vector V_{II} are different. In this way the lengths of the data set vector $V_{I\&II}$ can be different for different data sets. This means we are unable to compare the data set vectors of different data sets directly.

In order to address this problem, a common statistical method, namely, the five-number summary [35] of a data set, is extended and used to approximate the original vectors.

Specifically, both of the V_I and V_{II} are sorted in ascending order. Then *the minimum*, 1/8 *quantile*, 2/8 *quantile*, 3/8 *quantile*, 4/8 *quantile*, 5/8 *quantile*, 6/8 *quantile*, 7/8 *quantile*, and *the maximum* are computed for V_I and V_{II}, respectively.

By now, we can obtain the unified feature vector $V_{I\&II}$ for a data set by merging the *one-item* feature vector V_I and the *two-item* feature vector V_{II}. As the corresponding length is fixed to 18, different data sets can now be represented and compared.

4.3. An example

For the sake of a more clear understanding, we use an example to illustrate the proposed data set feature extraction method.

We take a part of the data set *Car* as example data, which consists of 10 instances and 3 nominal attributes. Table 1 shows its details.

Firstly, the distinct values of each attribute are identified. As attribute *Buying* has four different nominal values, the attribute value set $C_{Buying}=\{vhigh,high,med,low\}$ is obtained. By the same

Table 1
Description of the data set Car.

Instance ID	Attribute		
	Buying	*Doors*	*Class*
1	*vhigh*	'2'	*vgood*
2	*med*	'5+'	*unacc*
3	*high*	'4'	*acc*
4	*vhigh*	'2'	*good*
5	*low*	'3'	*vgood*
6	*low*	'2'	*vgood*
7	*vhign*	'2'	*unacc*
8	*high*	'3'	*acc*
9	*med*	'4'	*vgood*
10	*high*	'5+'	*good*

Table 2
The transformed data set.

Instance ID	Attribute											
	vhigh	high	med	low	'2'	'3'	'4'	'5+'	unacc	acc	good	vgood
1	1	0	0	0	1	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	1	1	0	0	0
3	0	1	0	0	0	0	1	0	0	1	0	0
4	1	0	0	0	1	0	0	0	0	0	1	0
5	0	0	0	1	0	1	0	0	0	0	0	1
6	0	0	0	1	1	0	0	0	0	0	0	1
7	1	0	0	0	1	0	0	0	1	0	0	0
8	0	1	0	0	0	1	0	0	0	1	0	0
9	0	0	1	0	0	0	1	0	0	0	0	1
10	0	1	0	0	0	0	0	1	0	0	1	0

token we can get the attribute value sets $C_{Doors}=\{'2','3','4','5+'\}$ for attribute *Doors*, and $C_{Class}=\{unacc,acc,good,vgood\}$ for attribute *Class*.

Secondly, each attribute is replaced by its attribute value set. That is, *Buying* is replaced by C_{Buying}, *Doors* replaced by C_{Doors} and *Class* replaced by C_{Class}. Through this we get the attributes of the transformed binary data set.

For each instance of the data set *Car*, each value $Value_A$ of attribute A is replaced by the values $Value_{C_A}$ of the corresponding attribute value set C_A. For example, in the first instance, (i) the value $Value_{Buying}$ of the first attribute *Buying* is *vhigh*, and $C_{Buying}[1]=\{vhigh\}$, so $Value_{C_{Buying}}=\{1,0,0,0\}$ will replace $Value_{Buying}=\{vhigh\}$; (ii) the value $Value_{Doors}$ of the second attribute *Doors* is '2', and $C_{Doors}[1]='2'$, so $Value_{C_{Doors}}=\{1,0,0,0\}$ will replace $Value_{Doors}=\{'2'\}$; and (iii) the value $Value_{Class}$ of the third attribute *Class* is *vgood*, and $C_{Class}[4]=vgood$, so $Value_{C_{Class}}=\{0,0,0,1\}$ will replace $Value_{Class}=\{vgood\}$.

Repeat the process for the remainder instances, and the data set can be transformed into binary format. Table 2 contains the results.

Thirdly, given the transformed binary data set, the computation of the *one-item set* vector V_I is straightforward. That is, for each single attribute, we just need to count the number of '1's, and then divide it by the number of instances; the result is a member of V_I corresponding to a single attribute item set.

For example, in the transformed data set, the count of the first element of *one-item set* $\{vhigh\}$ is 3. The number of instances is 10, so $3/10=0.3$ constitutes the first member of V_I. In this way, we can obtain the *one-item set* vector {0.3, 0.3, 0.2, 0.2, 0.4, 0.2, 0.2, 0.2, 0.2, 0.2, 0.4}.

The computation of the *two-item set* vector V_{II} is not as simple as that of V_I. For each element $\{a_i,a_j\}\in II$, we first compute $T_{a_i,a_j}(\omega)=\omega_i\oplus\omega_j$. Then we count the number of '1's for the *two-item set* over all the instances and divide it by the number of instances. In this way we get a member of the *two-item set* vector V_{II} corresponding to the item set $\{a_i,a_j\}$.

For example, in the transformed binary data set, the first element of the *two-item set* is $\{vhigh, 2\}$. For the first instance $\omega=\{1,1\}$, $T_{vhigh,2}(\omega)=1\oplus 1=0$; for the second instance $\omega=\{0,0\}$; $T_{vhigh,2}(\omega)=0\oplus 0=0$; and $T_{vhigh,2}(\omega)=0\oplus 0=0$ for the third instance, and so on. Count the number of '1's we get 1, so the first member of V_{II} is $1/10=0.1$. In this way we can get the *two-item set* vector $V_{II}=\{0.1,0.5,0.5,\ldots,0.2,0.6\}$, which is a 48-dimension[5] vector.

Finally, sort both of the feature vector V_I and V_{II} in ascending order. Then compute *the minimum*, $1/8$ *quantile*, $2/8$ *quantile*, ..., $7/8$ *quantile*, and *the maximum* for each of them. After that we obtain our data set feature vector as desired: $V_{I\&II}=\{0.2,0.2, 0.2,0.2,0.2,0.2, 0.3,0.4,0.4,0.1,0.2,0.3,0.4,0.4,0.4,0.4,0.5,0.7\}$.

[5] Each of the three attributes has four different values, so a total of $48=3\times(4\times 4)$ elements are obtained.

5. Empirical study

5.1. Experimental setup

For the purposes of evaluating the performance and effectiveness of our proposed classification algorithm recommendation method, verifying whether or not the method is potentially useful in practice, and allowing other researchers to confirm our results, we set up our experimental study as follows.

1. 84 data sets from the UCI repository [32] are used in the experiments.
   Table 3 presents the number of instances, the number of attributes by which each instance is described (not including the class label), and the number of classes for each data set.
   In order to facilitate the calculation of the feature vectors, for data sets containing continuous values, the MDL method [36] is employed to divide the continuous values into intervals. This method has been widely used by the researchers of data mining and machine learning communities in many different situations, such as feature selection (e.g., CFS [37], FCBF and INTERACT [38]), association rule mining (e.g., Apriori, FP-growth [39]), and classification (e.g., CBA [9], CMAR [10], MCAR [11]), etc. Thus, it is reasonable to use this method to assist handling continuous values.
2. 17 different types of classification algorithms are selected to classify data sets.
   They are probability-based Averaged One-Dependence Estimators (AODE), Naive Bayes (NB) and Bayes Net (BN); tree-based C4.5 and CART, rule-based RIPPER, PART and OneR [40]; neural networks-based Multilayer Perceptron (MLP); lazy learning algorithm IB1 [41]; and the support vector algorithm Sequential Minimal Optimization (SMO) [42].
   Besides these single learning algorithms, we also employ the ensemble classifier algorithms. Bagging [43] is applied with the three simple classifiers C4.5, PART and Naive Bayes as the base classifier, respectively. At the same time, Boosting [44] is also employed with the same three simple classifiers as the base classifier, respectively.
3. When simulating the classification algorithm recommendation process, the *jackknife* strategy is employed. That is, each data set has the chance of being recommended classification algorithms and all others are viewed as historical data sets. This allows us to compute the corresponding statistics and

Table 3
Description of the 84 data sets.

ID	Name	Attributes	Instances	Classes	ID	Name	Attributes	Instances	Classes
1	anneal.ORIG	38	898	5	2	anneal	38	898	5
3	arrhythmia	279	452	13	4	audiology	69	226	24
5	australian	14	690	2	6	autos	25	205	6
7	balance-scale	4	625	3	8	breast-cancer	9	286	2
9	breast-w	9	699	2	10	car	6	1728	4
11	cleve	11	303	2	12	cmc	9	1473	3
13	colic	22	368	2	14	connect-4	42	13 512	3
15	Credit-a	15	690	2	16	credit-g	20	1000	2
17	crx	15	690	2	18	cylinder-bands	39	540	2
19	dermatology	34	366	6	20	diabetes	8	768	2
21	ecoli	7	336	8	22	flags	29	194	8
23	german	15	1000	2	24	glass	9	214	6
25	haberman	3	306	2	26	hayes-roth	4	132	3
27	heart-c	13	303	2	28	heart-h	13	294	2
29	heart-statlog	13	270	2	30	hepatitis	19	155	2
31	horse-colic.ORIG	21	368	2	32	hypo	23	3163	2
33	hypothyroid	29	3772	4	34	ionosphere	34	351	2
35	iris	4	150	3	36	kdd_JapaneseVowels_1	14	5687	9
37	kdd_JapaneseVowels_2	13	4274	9	38	kdd_synthetic_control	61	600	6
39	kr-vs-kp	36	3196	2	40	Labor	16	57	2
41	led7	7	3200	10	42	Letter	16	20 000	26
43	liver-disorders	6	345	2	44	lung-cancer	56	32	3
45	lymph	18	148	4	46	Mfeat-fourier	76	2000	10
47	mfeat-karhunen	64	2000	10	48	Mfeat-morphological	6	2000	10
49	mfeat-zernike	47	2000	10	50	molecular-biology_promoters	58	106	2
51	monks-problems-1	6	556	2	52	monks-problems-2	6	601	2
53	monks-problems-3	6	554	2	54	mushroom	22	8124	2
55	nursery	8	12 960	5	56	optdigits	64	5620	10
57	page-blocks	10	5473	5	58	pendigits	16	10 992	10
59	pima	6	768	2	60	postoperative-patient-data	8	90	3
61	primary-tumor	17	339	21	62	segment	19	2310	7
63	shuttle-landing-control	6	15	2	64	sick	29	3772	2
65	solar-flare_1	12	323	2	66	solar-flare_2	12	1066	3
67	sonar	60	208	2	68	soybean	35	683	19
69	spambase	57	4601	2	70	spect	22	267	2
71	spectrometer	102	531	48	72	splice	61	3190	3
73	sponge	45	76	3	74	tae	5	151	3
75	tic-tac-toe	9	958	2	76	trains	32	10	2
77	transfusion	3	748	2	78	vehicle	18	846	4
79	vote	16	435	2	80	vowel	13	990	11
81	waveform-5000	40	5000	3	82	wine	13	178	3
83	yeast	7	1484	10	84	zoo	17	101	7

further to provide a overview of the performance of our classification algorithm recommendation method.

4. When conducting the recommendation for each data set, the number of nearest neighbors is set to be 10% of the data sets available as used by Soares and Brazdil [45] and Brazdil et al. [27].
5. The three different values 0%, 0.05% and 0.1% are set to the parameter α of Eq. (1), respectively. This permits us to test the usability of our proposed method in different situations.
6. The traditional data set characteristics, which were used by many researchers (such as Michie et al. [20], King et al. [46], and Smith [47]), is chosen to compare with our proposed new data set feature $V_{I\&II}$. Table 4 shows the traditional data set characteristics, the corresponding values are obtained from the original data sets. In this paper, we refer the traditional data set characteristics as $V_{S\&I}$.
   In addition, the recently developed data set characteristics [30] is selected as a benchmark as well. Table 5 briefly summarizes these characteristics, and the details can be found in [30,29]. In this paper, we refer these data set characteristics as V_C.

5.2. Evaluation criteria

To evaluate the performance of our proposed classification algorithm recommendation method, three metrics, which are *Classification Accuracy* of the recommended classification algorithms, *Recommendation Accuracy* and *Hit Rate*, are employed.

Classification accuracy: For most classification problems, classification accuracy is one of the most important concerns. Thus, it is expected that the recommended algorithms are as accurate as the most applicable ones for the problem at hand. This means the smaller the classification accuracy differences between the recommended algorithms and the best ones are, the better the recommendation method is.

Recommendation accuracy (*RA*): A higher classification accuracy does not always imply a better recommendation, since it is possible that all the candidate classification algorithms perform well on a given data set and the differences among them are small. On the other hand, it is better to know that compared with other algorithms, how well the recommended algorithms are. So the new metric *RA* is introduced.

For a given data set D, Let *AlgR* be the recommended classification algorithm, *AlgA* the most applicable classification algorithm (i.e., the best algorithm for data set D) and *AlgW* the worst candidate classification algorithm. *RA* is defined as

$$RA = \frac{P_{AlgR,D} - P_{AlgW,D}}{P_{AlgA,D} - P_{AlgW,D}} \times 100\%, \tag{5}$$

where $P_{X,Y}$ denotes the performance of algorithm X on data set Y.

Table 4
Statistic and information theory data set characteristics.

Measures	Definitions
Inst.Num	Number of instances
Attr.Num	Number of attributes
Class.Num	Number of classes
BinAttr.Prop	Proportion of binary attributes
NomAttr.Prop	Proportion of nominal attributes
NumAttr.Prop	Proportion of numerical attributes
AttrWithOutlier.Prop	Proportion of numerical attributes with outliers over all numerical attributes
Skewness	Skewness of data based on numerical attributes
Kurtosis	Kurtosis of data based on numerical attributes
MeanAbsCoef	Mean absolute coefficient of attribute pairs
$H(C)$	Entropy of classes
$\overline{H}(X)$	Mean entropy of nominal attributes
$\overline{M}(C,X)$	Mean mutual information of classes and attributes based on nominal attributes
En.attr	Equivalent number of attributes $H(C)/\overline{M}(C,X)$
Ns.ratio	Noise-signal ratio $\overline{H}(X)/\overline{M}(C,X)-1$

Table 5
Problem complexity based data set characteristics.

Measures	Definitions
Bound.Len	Length of class boundary
Adherence.Prop	Proportion of retained adherence subsets
Intra/Inter.Ratio	Ratio of average intra/interclass nearest neighbors
NN.Nonlinerity	Nonlinearity of Nearest Neighbors classifier
Linear.Nonlinerity	Nonlinearity of linear classifier
Fisher.Ratio	Maximum Fisher's discriminant ratio
Ins/Attr	Training set size relative to feature space dimensionality

RA compensates classification accuracy in two ways. (i) It takes into account classification performance that includes both classification accuracy and running time, since running time also is one of our concerns. (ii) Instead of using absolute performance, relative performance is employed. It allows us to know that compared with the worst candidate classification algorithm, how well the recommended algorithms are.

From the definition of the algorithm performance (see Eq. (1)) we know that, when $\alpha=0$, it turns into classification accuracy. So *RA* can be viewed as a general form of classification accuracy.

Note that *RA* only describes the performance of each individual recommendation. In order to evaluate our proposed method over all data sets and provide a big picture of the recommendations, the average recommendation accuracy (*ARA*) is defined as

$$ARA=\frac{\sum_{i=1}^{N} RA_i}{N}\times 100\%, \tag{6}$$

where RA_i is the *RA* on data set *i* and *N* denotes the number of the test data sets.

Hit Rate (*HR*): *HR* is the proportion of the data sets being correctly recommended classification algorithms among all data sets. It is a complementary measure to *RA*.

Suppose $S_{Alg,a}$ is the set of real applicable algorithms, which is identified by the procedure *ApplicableAlgIndentification*, for a data set and $S_{Alg,r}$ is the set of recommended algorithms by the recommendation method for the same data set, if the intersection of $S_{Alg,a}$ and $S_{Alg,r}$ is not empty, that is $S_{Alg,a}\cap S_{Alg,r}\neq\phi$, then let the hit count *HCount* be 1, indicating that the recommendation hits the target successfully. So the *Hit Rate* (*HR*) presents the proportion of hitting times on the total recommendation times, defined as

$$HR=\frac{\sum_{i=1}^{N} HCount_i}{N}\times 100\%, \tag{7}$$

where $HCount_i$ is the *HCount* on data set *i* and *N* denotes the number of data sets.

5.3. Experimental results and analysis

In this section, we present the recommendation results of our proposed method for the 84 data sets in terms of classification accuracy of the recommended algorithms, recommendation accuracy (*RA*) and hit rate (*HR*), respectively.

We also provide the comparison results of our proposed classification algorithm with data set feature and the traditional data set characteristics.

Moreover, the results are presented at two levels: detailed information for each individual data set and the summarized information over all data sets.

5.3.1. Classification accuracy of the recommendations

In order to provide an intuitive image on the performance of our proposed algorithm recommendation method, a scatter plot is employed to show the accuracies of the recommended algorithms vs. those of the real best algorithm over the 84 data sets. For each plot, a line that is parallel to the diagonal is drawn, so 90% of recommendations fall into the area between the diagonal and the parallel line. Figs. 3–5 show the classification accuracies with $V_{I\&II}$, $V_{S\&I}$ and V_C, respectively.

From Figs. 6–8, it can be observed that:

- Most points are on or very close to the diagonal. This indicates that most of the recommended algorithms are as good as or nearly as good as the real best algorithm.
- For our proposed $V_{I\&II}$ and the problem complexity based V_C, the margins between the diagonal and the parallel line are less than those of $V_{S\&I}$, and their maximum margins are both 5.81% that are even smaller than the smallest margin of $V_{S\&I}$, 6.46%. This indicates that the recommendations with $V_{I\&II}$ and V_C are better that those of with the traditional $V_{S\&I}$.
  The margins between the diagonal and the parallel line of our proposed $V_{I\&II}$ are smaller than or equal to those of V_C, and the distribution of the classification accuracies with $V_{I\&II}$ is more concentrative. This means the recommendations with $V_{I\&II}$ are superior to those with V_C.
- For each of the three different types of data set characteristics, the margins between the diagonal and the parallel line increase with the ascending of the α values. This is because that α values represent the amount of the compromise

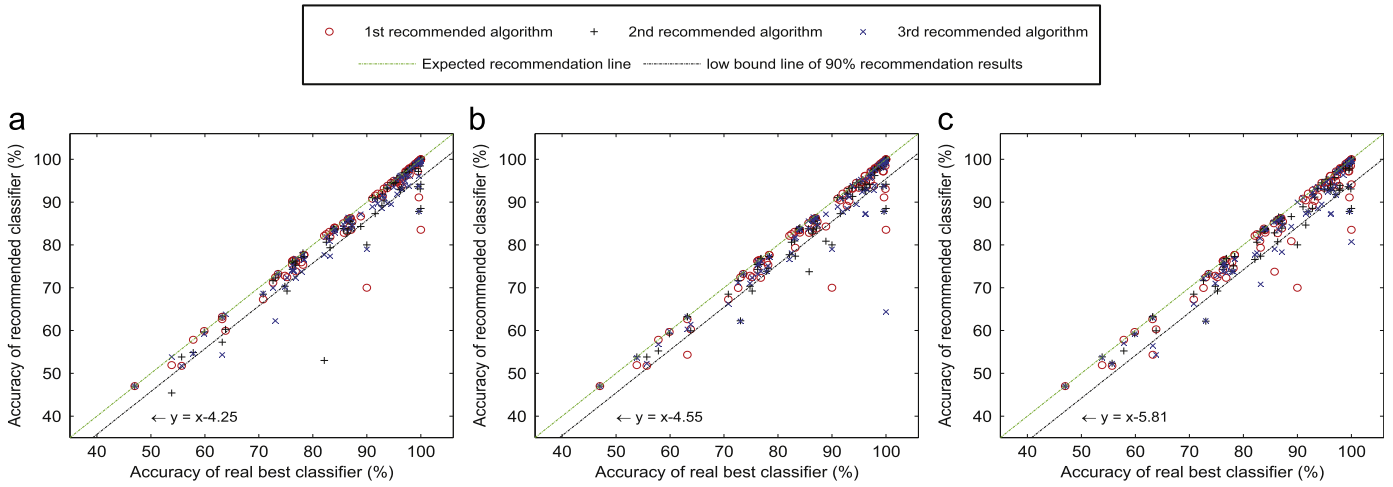

Fig. 3. Classification accuracy of the real best algorithm vs. classification accuracy of the recommended algorithms under our proposed $V_{I\&II}$. (a) α=0%, (b) α=0.05% and (c) α=0.1%.

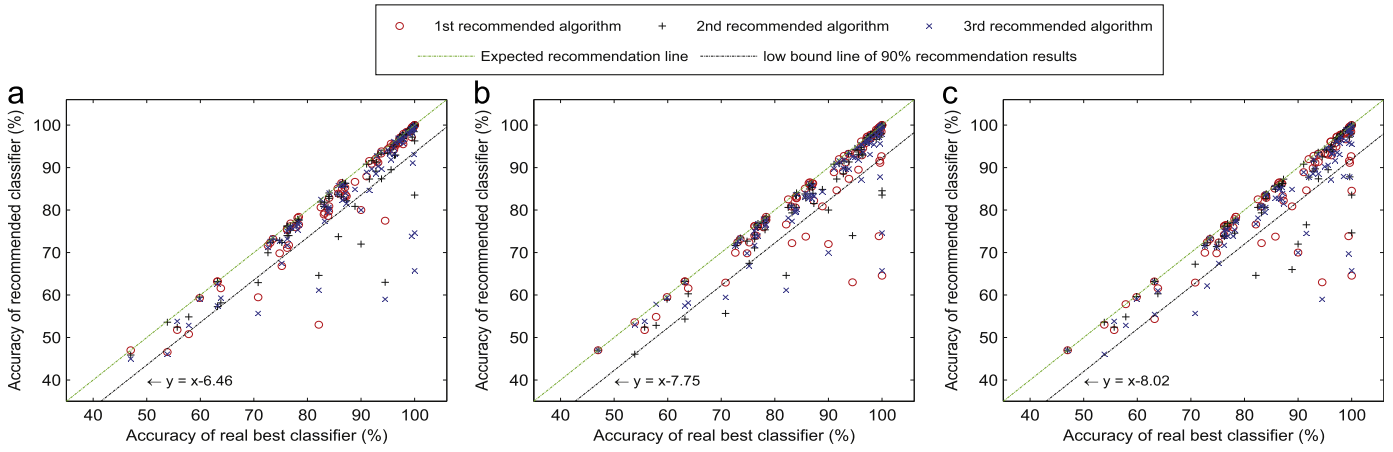

Fig. 4. Classification accuracy of the real best algorithm vs. classification accuracy of the recommended algorithms under the traditional $V_{S\&I}$. (a) α=0%, (b) α=0.05% and (c) α=0.1%.

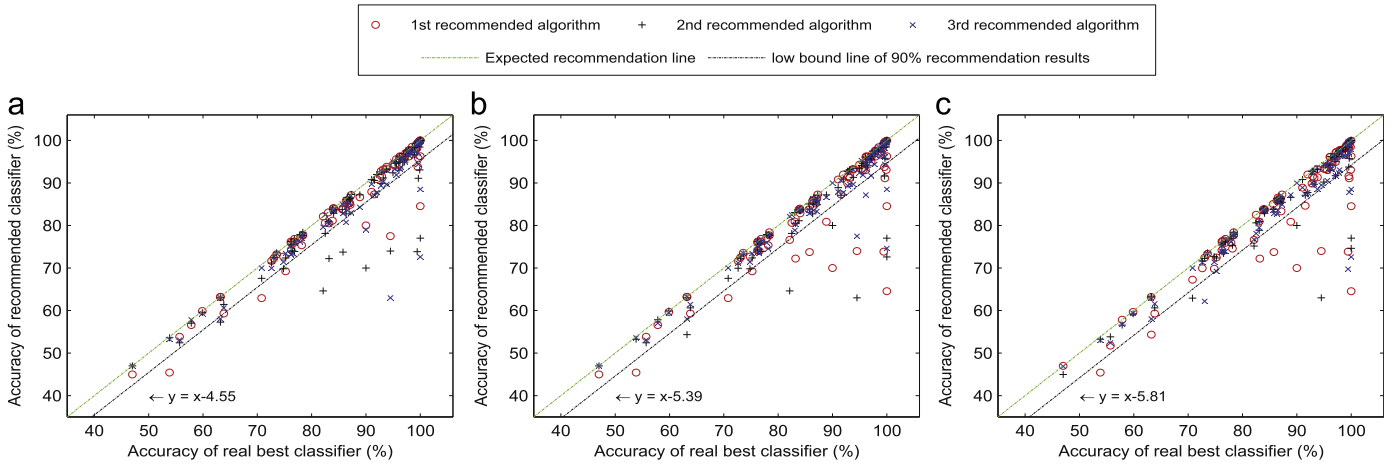

Fig. 5. Classification accuracy of the real best algorithm vs. classification accuracy of the recommended algorithms under the problem complexity based V_C. (a) α=0%, (b) α=0.05% and (c) α=0.1%.

between accuracy and runtime, so a larger α value means users are willing to trade more classification accuracies for shorter runtime. Thus, with the increase of the α values, the accuracy of the recommended algorithm deceases, i.e., becomes far away from the real best accuracy. Thus, the margins between the diagonal and the parallel line increases.

A more detailed description of the recommendation results is presented as well. For each of the 84 data set, its real best classification accuracy and the recommended algorithms' accuracies are plotted on one picture. Figs. 6–8 show the classification accuracies of the recommended algorithms with the data set feature $V_{I\&II}$, the data set characteristics $V_{S\&I}$ and problem complexity based features

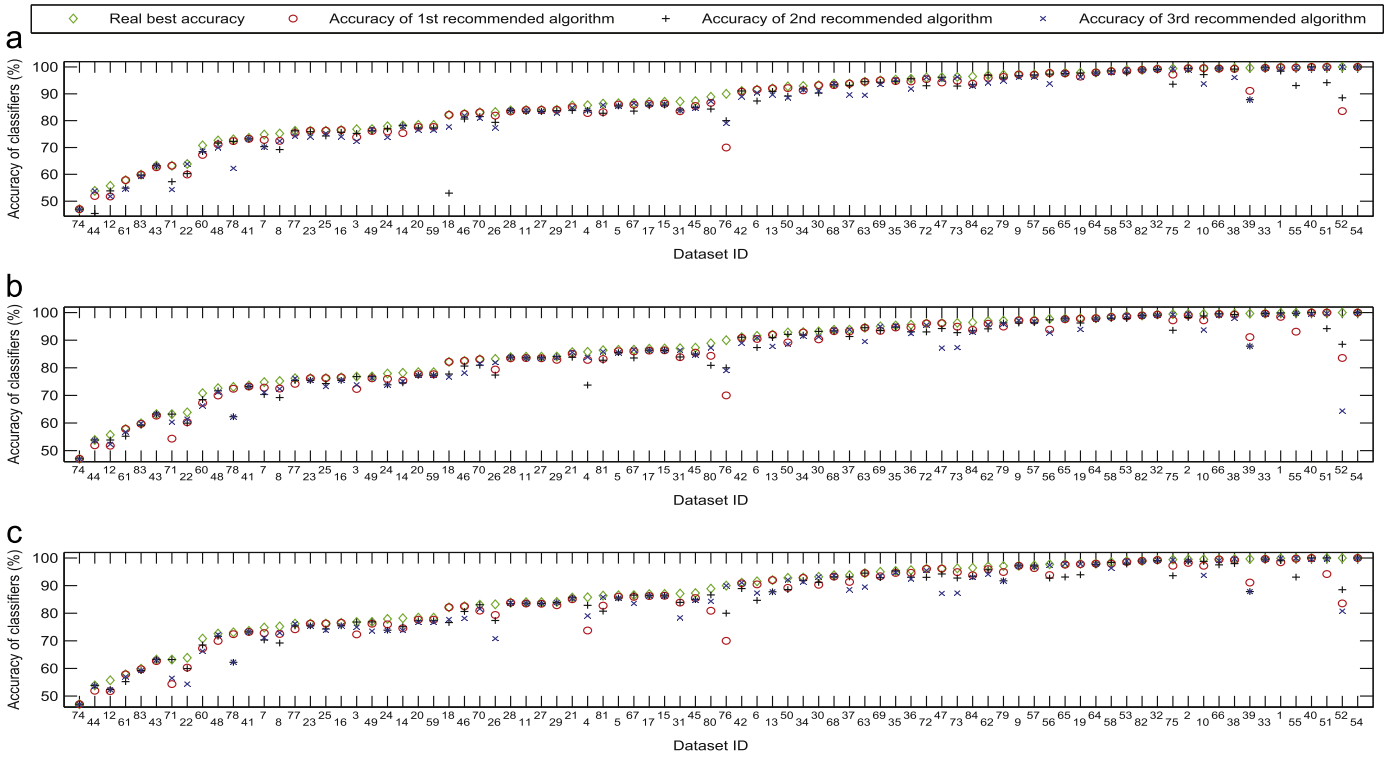

Fig. 6. Classification accuracies of the recommended algorithms under our proposed $V_{I\&II}$ and the real best algorithm over the 84 data sets. (a) α=0%, (b) α=0.05% and (c) α=0.1%.

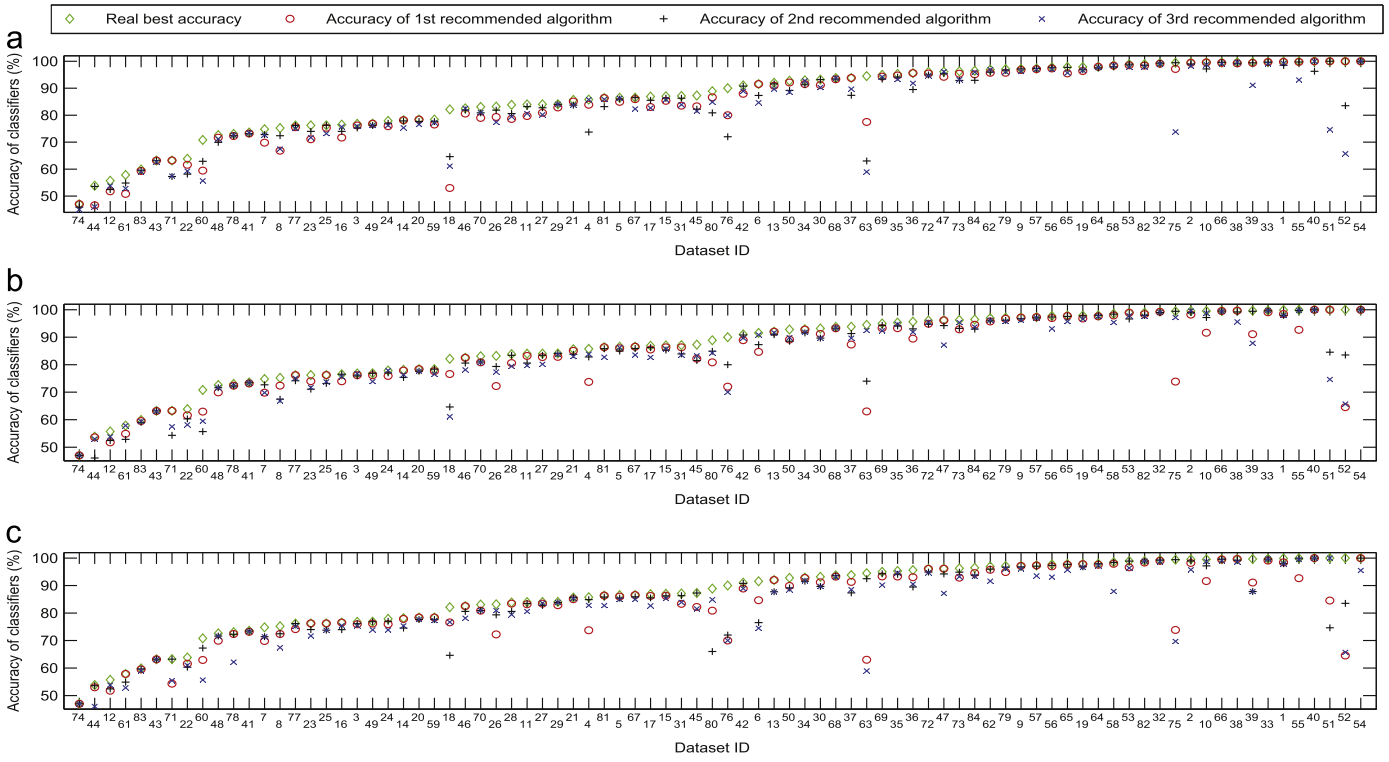

Fig. 7. Classification accuracies of the recommended algorithms under the traditional $V_{S\&I}$ and the real best algorithm over the 84 data sets. (a) α=0%, (b) α=0.05% and (c) α=0.1%.

V_C, respectively. On each plot, the accuracies are sorted in the ascending order of the real best accuracy to make the results clear.

From Figs. 6–8 we observe that:

- For all the three different types of data set characteristics $V_{I\&II}$, $V_{S\&I}$ and V_C, the recommendations of our proposed recommendation method are better "acceptable". The accuracies of the most recommended algorithms are very close to or just the same as those of the real best algorithm for each α. This means that both our proposed classification algorithm recommendation method and data set feature extraction method are effective.

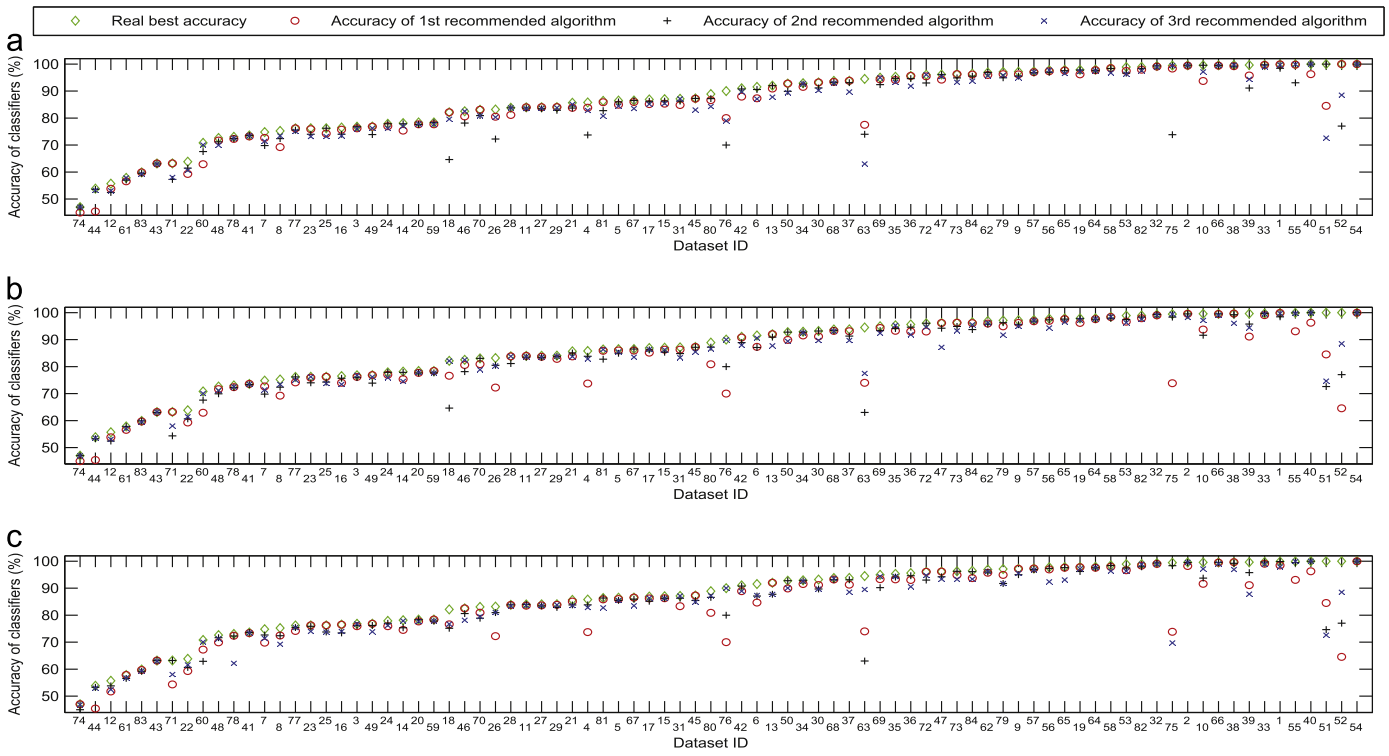

Fig. 8. Classification accuracies of the recommended algorithms under the problem complexity based V_C and the real best algorithm over the 84 data sets. (a) α=0%, (b) α=0.05% and (c) α=0.1%.

Table 6
Average classification accuracy of the recommended algorithms under $V_{I\&II}$, $V_{S\&I}$ and V_C for three different α values.

Recommended classifier	$\alpha = 0\%$			$\alpha = 0.05\%$			$\alpha = 0.1\%$		
	$V_{I\&II}$	$V_{S\&I}$	V_C	$V_{I\&II}$	$V_{S\&I}$	V_C	$V_{I\&II}$	$V_{S\&I}$	V_C
1st recommendation	**85.36**	84.31	85.09	**85.12**	83.85	84.51	**84.67**	84.04	84.05
2nd recommendation	84.58	83.84	83.43	84.18	83.34	84.13	84.26	82.79	84.17
3rd recommendation	84.59	82.77	83.70	84.41	82.50	84.31	83.74	82.26	84.15

The average accuracy of the real best algorithm is 86.66%.

- The best accuracy of the recommended algorithms for $V_{I\&II}$ is closer to the real best algorithm than those for both $V_{S\&I}$ and V_C, and the distribution of the accuracies for $V_{I\&II}$ is tighter than those for $V_{S\&I}$ and V_C as well. This indicates the recommendations of our proposed recommendation method with $V_{I\&II}$ are better than those of with $V_{S\&I}$ and V_C, and further reveals $V_{I\&II}$ is superior to both $V_{S\&I}$ and V_C.
- The first recommended algorithm may not be the most accurate one, for example, the data set 52 in Fig. 6(a), 71 in Fig. 6(b) and 4 in Fig. 6(c). However, for these data sets, the second and the third recommendations perform better than the first one. A similar situation could also be observed in Figs. 7 and 8.

Table 6 shows the summarized accuracies of the recommended algorithms under $V_{S\&I}$, $V_{I\&II}$ and V_C for the three different α values over the 84 data sets.

From Table 6 we observe that:

- Given the average classification accuracy 86.66% of the real best algorithm, the average classification accuracies of the recommended algorithms for $V_{I\&II}$ and V_C under $\alpha = 0\%$ is over 85%. This indeed is a very good approximation to the best accuracy.
- For each of the three different α values, the 1st recommended algorithm has the better classification accuracy. Thus, in practice, the 1st recommendation is usually preferred.
- For each of the three different types of data set characteristics, the average classification accuracies of the recommended algorithms are slightly decreasing with the increase of the α value from 0% through 0.05% to 0.1%. Recall that α represents the amount of accuracies users are willing to trade for a 10 times speedup or slowdown. Thus, a bigger α value can lead to more accuracies are traded for shorter running time, and this will results in accurate but more time-consuming algorithms may not be selected. Due to this reason, the average classification accuracies may decease with the increase of the α value.
- Compared with both $V_{S\&I}$ and V_C, our proposed data set feature $V_{I\&II}$ obtains better classification accuracy for each α value. This means our proposed $V_{I\&II}$ has its advantage in characterizing data sets.

To summarize, for each of the three different data set characteristics, the classification accuracies of the recommended algorithms are very close to those of the real best algorithms. This means that our proposed classification algorithm recommendation method is well enough. Moreover, the recommendation method with our proposed data set feature vector performs

better than with either the traditional data set characteristics or the problem complexity based characteristics. This reveals that our proposed data set feature is superior to both the traditional and the complexity-based ones.

5.3.2. RA and HR of the recommendations

In this section, both the individual and the summarized recommendation results are presented in terms of RA and HR, respectively.

Figs. 9–11 show the recommendation accuracy, average recommendation accuracy and the recommendation hit index for the 84 data sets when $\alpha = 0\%$, 0.05% and 0.1%, respectively. From them we can observe that:

- For each of the three data set characteristics $V_{I\&II}$, $V_{S\&I}$ and V_C, the recommendation accuracies of the most recommendations are greater than 85% and exceed the average recommendation accuracy, and only few recommendations do not hit the real applicable algorithms.
- For each of the three α values, the overall recommendations are similar although the data set characterization methods are different. This indicates that all the three methods can characterize a data set.

  However, the average recommendation accuracy ARA of our proposed method is greater than those of $V_{S\&I}$ and V_C. Moreover, for both the $V_{S\&I}$ and V_C, the recommendation accuracies for some data sets are very small. For example, the data sets 8, 52 and 60 in Figs. 9(b), 10(b) and 11(b) for $V_{S\&I}$, and the data sets 8 and 35 in Figs. 9(c), 10(c) and 11(c) for V_C. Therefore, it is risky to use the traditional method and the problem complexity based method to characterize a data set, and our proposed data set characterization method is suggested.
- Generally, once a recommendation hits the real applicable algorithms, the corresponding recommendation accuracy RA is higher than ARA. However, there are still some exceptions. For example, for data set 14 in Fig. 9(a), data set 4 in Fig. 9(b), and data set 4 in 9(c), the recommendations hit the real applicable algorithms, but the RA is lower than ARA. The reason is that the first recommended algorithm is either the last one of the real applicable algorithms or not matching any of the real applicable algorithms though the other recommended algorithms are among the real applicable algorithms. However, we take the first recommended algorithm as the first-string candidate to evaluate the RA, since the other recommended algorithms may still have better performance than the first recommended algorithm.
- Although the recommendations hit the real applicable algorithms, the recommendation accuracy RA is sometimes very small. For example, the recommendations for data set 30 in Fig. 9(a) and (b) and data set 63 in Fig. 9(c), the three recommendations all hit the real applicable algorithms but the corresponding recommendation accuracy RA is low. The reason is that the first recommended algorithm does not but

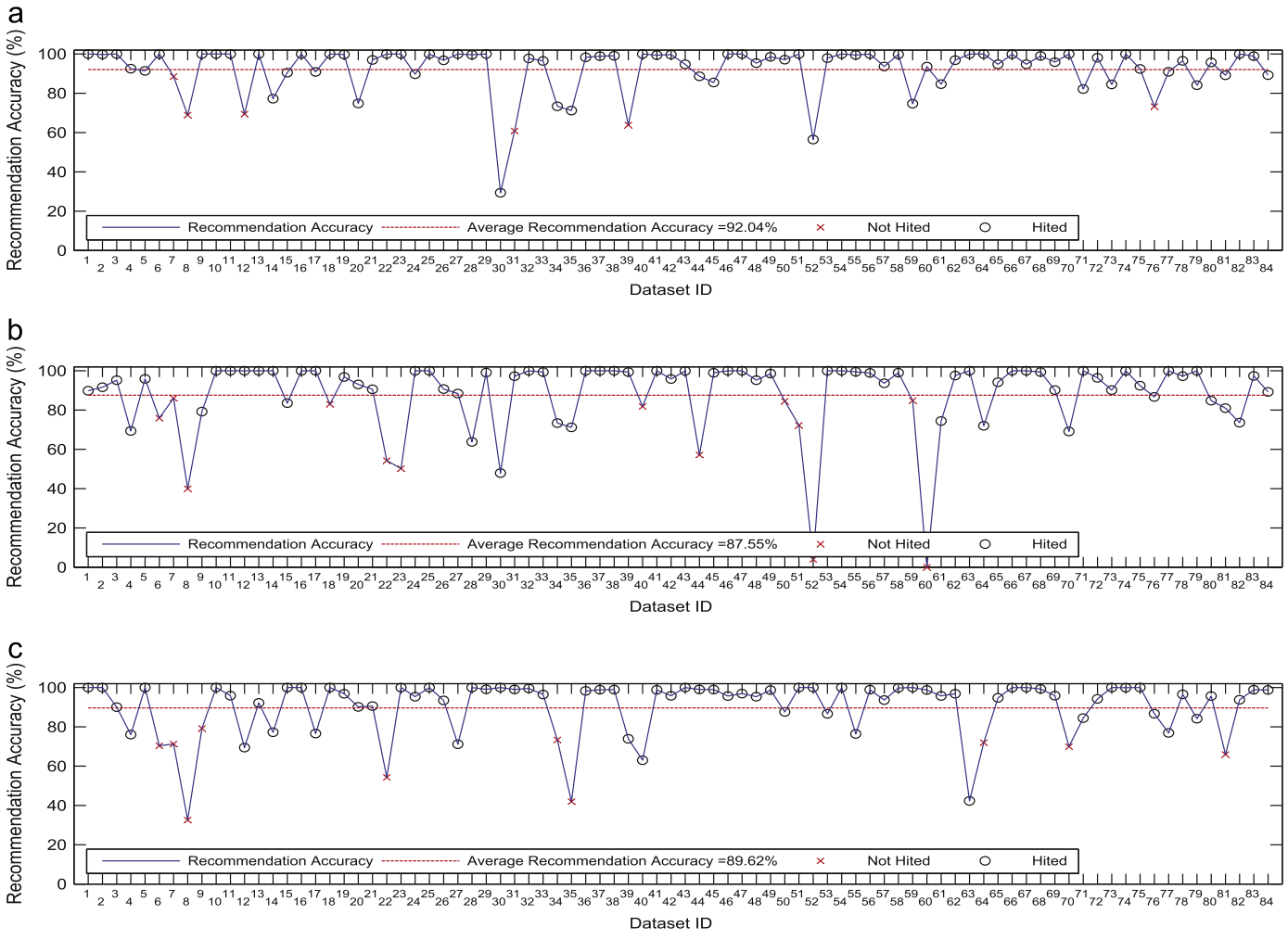

Fig. 9. Recommendation *RA*, *ARA* and *hit marker* for the 84 data sets with $\alpha = 0\%$. (a) The best ARA result of $V_{I\&II}$, (b) The best ARA result of $V_{S\&I}$ and (c) The best ARA result of V_C.

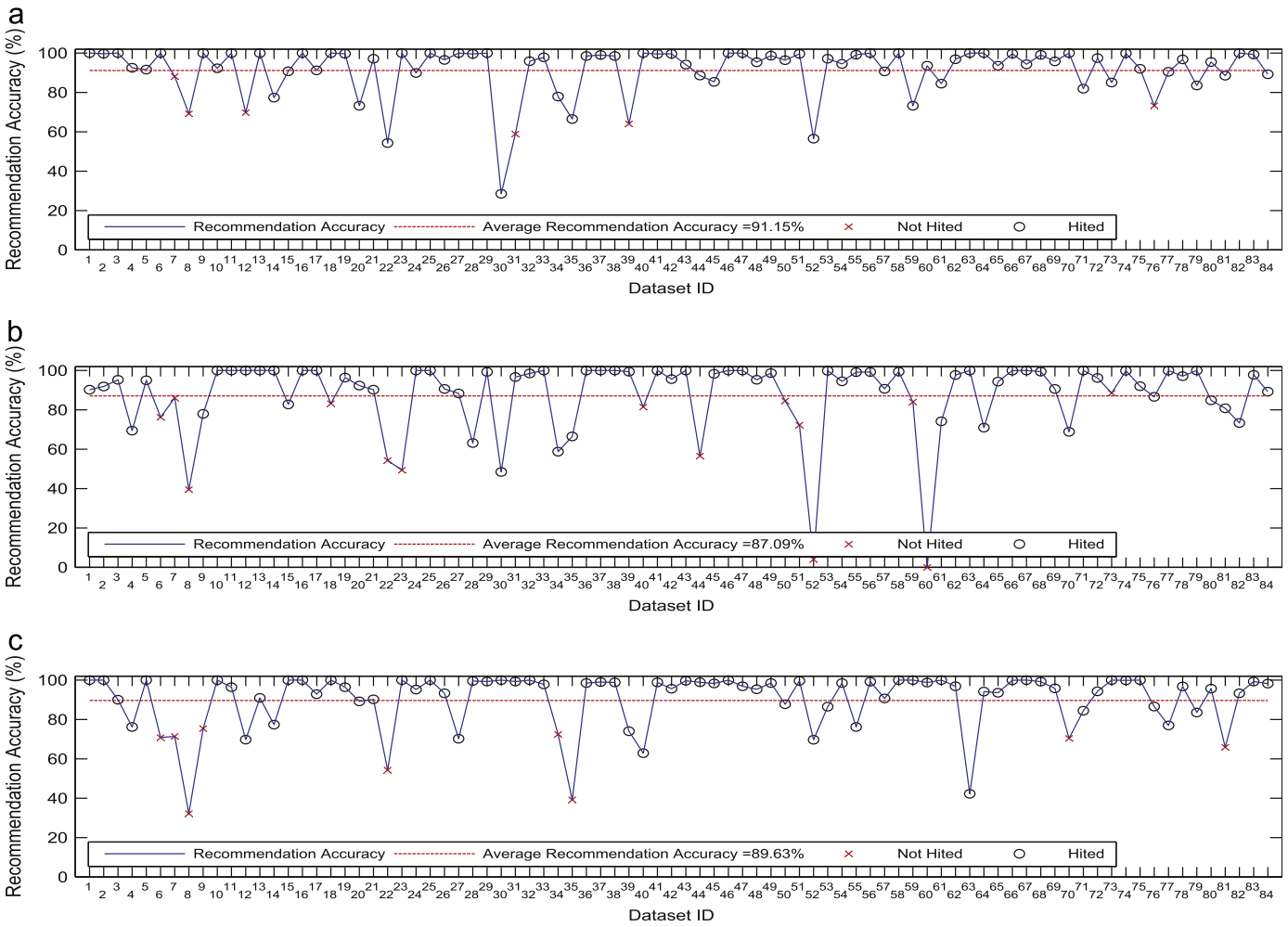

Fig. 10. Recommendation *RA*, *ARA* and *hit marker* for the 84 data sets with $\alpha = 0.05\%$. (a) The best ARA result of $V_{I\&II}$, (b) The best ARA result of $V_{S\&I}$ and (c) The best ARA result of V_C

the other recommended algorithm(s) hit the real applicable algorithms. This means that if the other recommended algorithms are employed to evaluate RA then the recommendation accuracy would be improved.

Table 7 shows the average recommendation accuracy and the average hit rate of the recommended algorithms under three data set characteristics $V_{I\&II}$, $V_{S\&I}$ and V_C over the 84 data sets when $\alpha = 0\%$, 0.05% and 0.1%, respectively.

From Table 7 we can make the following observations: (i) both the average recommendation accuracies and the average recommendation hit rates are well enough, the maximum average recommendation accuracy and maximum recommendation hit rate are 93.75% and 92.86%, respectively, while the minimum average recommendation accuracy and minimum recommendation hit rate are 81.13% and 73.81%, respectively. (ii) For each of the three different α values, our proposed $V_{I\&II}$ has the best performance in terms of the mean, max and min ARA and HR.

In order to explore whether the superiority of our method $V_{I\&II}$ is statistically significant, Wilcoxon signed ranks test [48] was conducted at significance level 0.05. The reason why the non-parametric test is employed lies in that it is difficult for ARA and HR to follow the normal distribution and satisfy variance homogeneous condition. The alternative hypotheses are that $V_{I\&II}$ are better than both $V_{S\&I}$ and V_C in characterizing data sets. The column "*p*-value" in Table 7 shows the test results.

From the column "*p*-value" of Table 7 we can observe that: (i) our proposed $V_{I\&II}$ is significantly superior to $V_{S\&I}$ in terms of both ARA and HR for all the three different α values since the corresponding *p*-values are less than 0.05; (ii) our proposed $V_{I\&II}$ significantly outperforms problem complexity based characteristics V_C in terms of both ARA and HR when $\alpha = 0\%$. When $\alpha = 0.05\%$ or 0.1%, $V_{I\&II}$ is significantly better than V_C in terms of ARA but has no significant difference in terms of HR. This is because that there usually exists a set of classification algorithms that are appropriate for a given data set, and the algorithms recommended by $V_{I\&II}$ and V_C are both in this algorithm set. However, the performance of the algorithms recommended by $V_{I\&II}$ is better than that recommended by V_C, although the performance difference is not significant.

To summarize, although $V_{I\&II}$, $V_{S\&I}$ and V_C are all capable of characterizing data sets, the average recommendation accuracy of $V_{I\&II}$ is the best. At the same time, $V_{I\&II}$ is unified and easy to interpret; while for $V_{S\&I}$, even knowing that some of its indicators are related to algorithms' running time, it is still not quite sure how these indicators interact with each other and work as a whole to characterize a data set. Moreover, $V_{I\&II}$ is adapted to various of data sets; while for V_C, its time complexity is high, so it can be hard to compute or even fail in calculating some of indicators such as length of classification bound and percentage of retained adhere subsets especially for larger data sets. What's more, the nonlinearity of NN (Nearest Neighbors) and linear classifiers are calculated based on Monte–Carlo estimate [49].

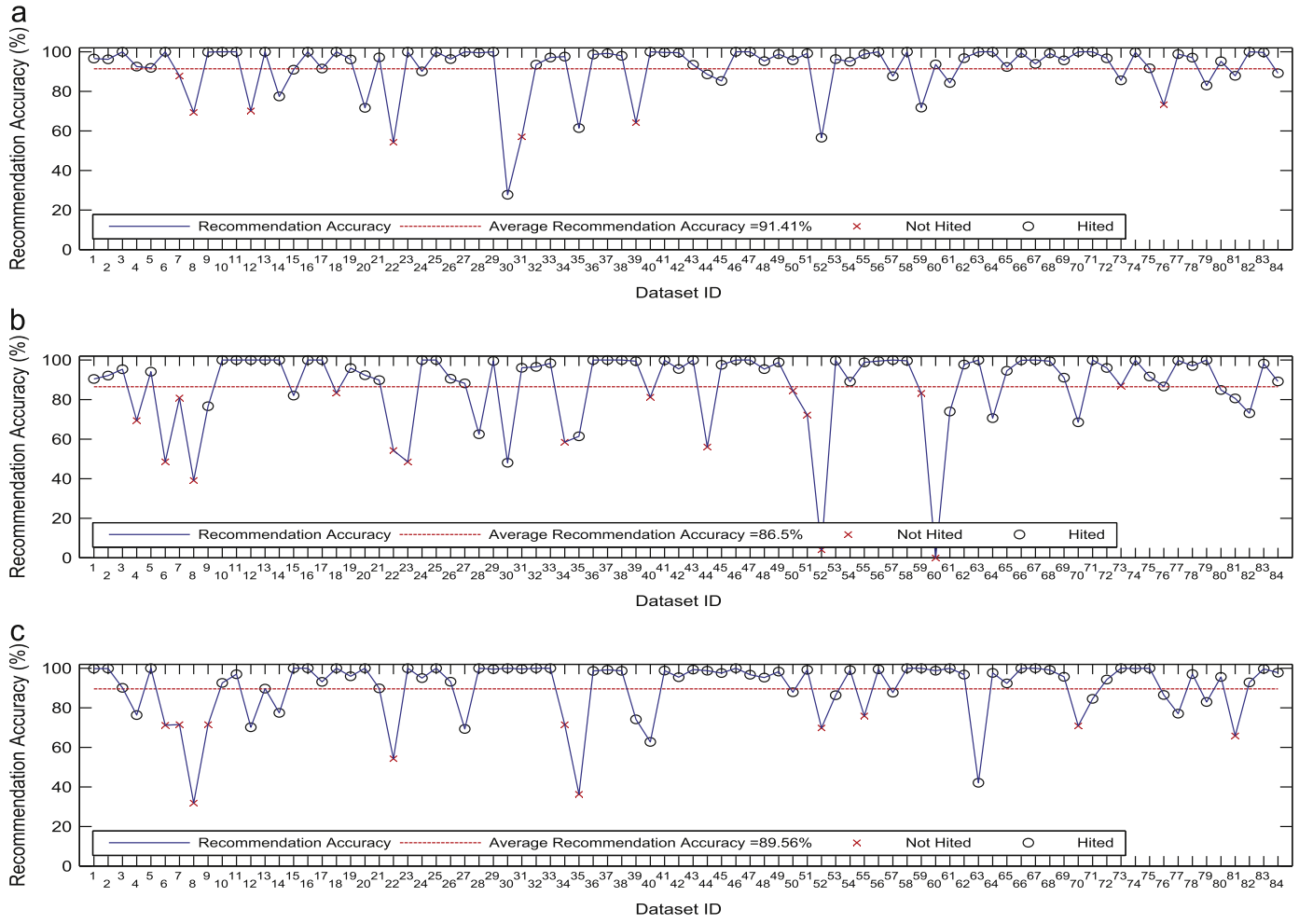

Fig. 11. Recommendation *RA*, *ARA* and *hit marker* for the 84 data sets with $\alpha = 0.1\%$. (a) The best ARA result of $V_{I\&II}$, (b) The best ARA result of $V_{S\&I}$ and (c) The best ARA result of V_C.

Table 7
ARA and HR of the recommended algorithms under three data set characteristics and three α values.

α (%)	Data set characteristics	ARA (%)				HR (%)			
		Mean	Max	Min	p-value	Mean	Max	Min	p-value
0	$V_{I\&II}$	90.46	93.75	88.84	–	89.89	92.86	84.52	–
	$V_{S\&I}$	82.40	86.87	80.60	0	83.26	88.10	75.00	0
	V_C	88.53	89.82	86.04	0	88.97	90.48	78.57	0.001
0.05	$V_{I\&II}$	90.26	92.98	88.20	–	89.50	92.86	83.33	–
	$V_{S\&I}$	81.63	86.35	79.46	0	82.21	86.90	75.00	0
	V_C	87.89	90.01	84.38	0	89.10	90.48	78.57	0.6092
0.1	$V_{I\&II}$	90.08	92.71	88.14	–	89.21	91.67	83.33	–
	$V_{S\&I}$	81.13	86.59	78.84	0	82.10	86.9	73.81	0
	V_C	87.06	89.35	83.36	0	89.17	90.48	78.57	0.5007

It should be run many times for obtaining stable estimates of these indicators, this is very time consumption.

5.3.3. *Effect of the number of the nearest neighbors*

The different number of the nearest neighbors would lead to the different nearest neighbor data sets, and further the different recommendations.

Thus, in this subsection, we analyze the effect of the number of the nearest neighbors on the recommendation results and see if we can obtain some clue on how to determine the number of the nearest neighbors.

For this purpose, firstly, all the possible nearest neighbor settings are tested. That is, when identifying the nearest neighbor data sets for a given data set, the k is set from 1 to the number of the historical data sets minus 1. Then, we summarize the average recommendation accuracies and the average recommendation hit rates with respect to the number of the nearest neighbors (i.e. k) over the 84 data sets under the three different α values, respectively. Fig. 12 shows the details.

From Fig. 12 we observe that:

- For each of the three α values, the number of the nearest neighbors does affect both the average recommendation accuracy *ARA* and the average recommendation hit rate *HR*.

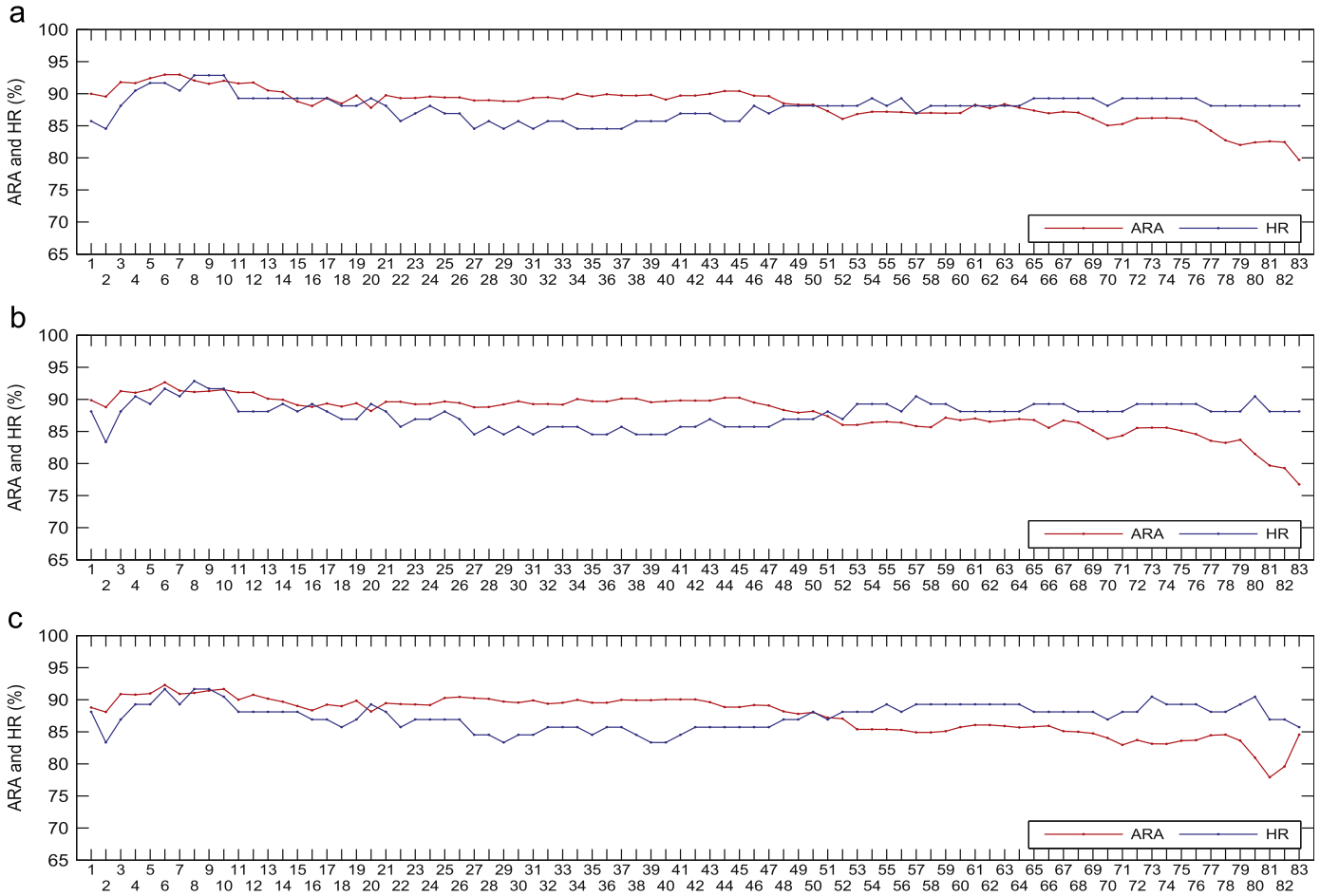

Fig. 12. *ARA* and *HR* under the different *k* values. (a) ARA and HR for each *k* with α=0%, (b) ARA and HR for each *k* with α=0.05% and (c) ARA and HR for each *k* with α=0.1%.

- For all the three α values, the average recommendation hit rate *HR* reaches the highest point quickly with the increase of the number of the nearest neighbors *k* although there are fluctuations, and a greater *k* can not surely improve the *HR* very much. This reveals that there is not a bigger *k* by which the average recommendation hit rate *HR* obtains its maximum.
- For all the three α values, the average recommendation accuracy *ARA* climbs up the top most point very quickly with the increase of the number of the nearest neighbors *k*, although there are fluctuations before and after that point, and a greater *k* cannot improve *HR* but seems to make it worse. This reveals that there is a small *k* by which the average recommendation accuracy *ARA* achieves its maximum.
- Although the average recommendation hit rate *HR* and the average recommendation accuracy *ARA* obtain their maxima at the different *k* values, these *k* values all fall into the range of 8–13. This means if we set *k* to around 10%–15% of the number of historical data sets, a better recommendation accuracy and recommendation hit rate can be obtained. This is a subset of the 10%–25% that was suggested by Soares and Brazdil [45] and Brazdil et al. [27].

6. Conclusion

In this paper, we have presented a recommendation method for classification algorithms based on the novel data set characteristics, with an aim to assist people in selecting algorithms among a large number of candidates for a new classification problem.

In this method, the data set features are first extracted, the nearest neighbors of a new data set are next recognized, and their applicable classification algorithms are identified. Then the relationship between the neighbor data sets and the classification algorithms' performance are built up, upon which the appropriate classification algorithms can be recommended to a new classification problem.

In order to facilitate the recommendation, we have also presented a data set feature extraction method. This method uses structural and statistical information based feature vectors to characterize each data set, which is quite different from the traditional data set characterization methods.

In order to validate our proposed recommendation method, 84 data sets and 17 different types of classifiers have been used in the experiment. For a comparison, we also conducted the recommendation with the traditional data set characteristics and the problem complexity based characteristics. The experimental results show that (i) our proposed classification algorithm recommendation method is effective and works well, and (ii) our proposed data set characterization method is superior to both the traditional and the problem complexity based methods. Moreover, we have further explored the impact of the number of the nearest neighbors on the recommendation results, and recommend to set it to the 10–15% of the number of the historical data sets.

For the future work, we plan to explore further the possible relationships between data mining algorithm parameters and data set features. With the help of the proposed data set extraction method, we envision the task being feasible to help user choose proper algorithms and associated parameters for any specific data mining problem.

Acknowledgments

The authors would like to thank Prof. Geoff Webb (Monash University, Australia), the editors and the anonymous reviewers for their insightful and helpful comments and suggestions, which resulted in substantial improvements to this work. This work is supported by the National Natural Science Foundation of China under grant 61070006.

References

[1] J.R. Quinlan, Discovering rules by induction from large collections of examples, in: D. Michie (Ed.), Expert Systems in the Micro-Electronic Age, Edinburgh University Press, Edinburgh, 1979, pp. 168–201.
[2] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[3] L. Breiman, Classification and Regression Trees, Chapman & Hall, CRC, 1984.
[4] A.W. Moore, D. Zuev, Internet traffic classification using Bayesian analysis techniques, in: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, ACM, 2005, pp. 50–60.
[5] G.I. Webb, J.R. Boughton, Z. Wang, Not so naive Bayes: aggregating one-dependence estimators, Machine Learning 58 (1) (2005) 5–24.
[6] W.W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, 1995, pp. 115–123.
[7] P. Clark, T. Niblett, The CN2 induction algorithm, Machine Learning 3 (4) (1989) 261–283.
[8] E. Frank, I.H. Witten, Generating accurate rule sets without global optimization, in: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., 1998, pp. 144–151.
[9] B. Liu, W. Hsu, Y. Ma, Integrating classification and association rule mining, Knowledge Discovery and Data Mining (1998) 80–86.
[10] W. Li, J. Han, J. Pei, CMAR: accurate and efficient classification based on multiple class-association rules, in: Proceedings IEEE International Conference on Data Mining, IEEE Computer Society, 2001, pp. 369–376.
[11] F. Thabtah, P. Cowling, Y. Peng, MCAR: multi-class classification based on association rule, in: Proceedings of the ACS/IEEE 2005 International Conference on Computer Systems and Applications, IEEE, 2005, pp. 1–7.
[12] X. Zhu, Q. Song, Z. Jia, A weighted voting-based associative classification algorithm, The Computer Journal 53 (6) (2010) 786–801.
[13] S.M. Weiss, I. Kapouleas, An empirical comparison of pattern recognition. in: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, vol. 1, Morgan Kaufmann Publishers Inc., 1989, pp. 781–787.
[14] J.W. Shavlik, R.J. Mooney, G.G. Towell, Symbolic and neural learning algorithms: an experimental comparison, Machine Learning 6 (2) (1991) 111–143.
[15] R.P.W. Duin, A note on comparing classifiers * 1, Pattern Recognition Letters 17 (5) (1996) 529–536.
[16] S. Ali, K.A. Smith, On learning algorithm selection for classification, Applied Soft Computing 6 (2) (2006) 119–138.
[17] D.H. Wolpert, W.G. Macready, No free lunch theorems for search, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.
[18] T. Mitchell, Machine Learning, McGraw-Hill, 1997.
[19] C.E. Brodley, Addressing the selective superiority problem: automatic algorithm/model class selection, in: Proceedings of the Tenth International Conference on Machine Learning, Citeseer, 1993, pp. 17–24.
[20] D. Michie, D.J. Spiegelhalter, C.C. Taylor, J. Campbell, Machine Learning, Neural and Statistical Classification, Ellis Horwood, 1995.
[21] J.R. Quinlan, Comparing connectionist and symbolic learning methods. in: Proceedings of a Workshop on Computational Learning Theory and Natural Learning Systems: Constraints and Prospects: Constraints and Prospects, vol. 1, MIT Press, 1994, pp. 445–456.
[22] D.W. Aha, Generalizing from case studies: a case study, in: Proceedings of the Ninth International Conference on Machine Learning, Citeseer, 1992, pp. 1–10.
[23] P. Brazdil, J. Gama, B. Henery, Characterizing the applicability of classification algorithms using meta-level learning, in: Proceedings of European Conference on Machine Learning, Springer, 1994, pp. 83–102.
[24] J. Gama, P. Brazdil, Characterization of classification algorithms, Progress in Artificial Intelligence (1995) 189–200.
[25] K.A. Smith, F. Woo, V. Ciesielski, R. Ibrahim, Modelling the relationship between problem characteristics and data mining algorithm performance using neural networks, Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, and Complex Systems 11 (2001) 356–362.
[26] K.A. Smith, F. Woo, V. Ciesielski, R. Ibrahim, Matching data mining algorithm suitability to data characteristics using a self-organising map, Hybrid Information Systems (2002) 169–180.
[27] P.B. Brazdil, C. Soares, J.P. Da Costa, Ranking learning algorithms: using IBL and meta-learning on accuracy and time results, Machine Learning 50 (3) (2003) 251–277.
[28] A. Kalousis, J. Gama, M. Hilario, On data and algorithms: understanding inductive performance, Machine Learning 54 (3) (2004) 275–312.
[29] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 289–300.
[30] E. Bernadó-Mansilla, T.K. Ho, Domain of competence of xcs classifier system in complexity measurement space, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 82–104.
[31] N. Tatti, Distances between data sets based on summary statistics, Journal of Machine Learning Research 8 (2007) 131–154.
[32] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, Irvine, CA, 2007, 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉.
[33] D.H. Fisher, L. Xu, N. Zard, Ordering effects in clustering, in: Proceedings of the Ninth International Workshop on Machine Learning, Morgan Kaufmann Publishers Inc., 1992, pp. 162–168.
[34] G.I. Webb, Multiboosting: a technique for combining boosting and wagging, Machine Learning 40 (2) (2000) 159–196.
[35] D.C. Hoaglin, F. Mosteller, J.W. Tukey, Understanding Robust and Exploratory Data Analysis, vol. 3, Wiley, New York, 1983.
[36] U. Fayyad, K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: R. Bajcsy, (Ed.), Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France, 1993, pp. 1022–1027.
[37] M.A. Hall, Correlation-Based Feature Selection for Machine Learning, Ph.D. Thesis, The University of Waikato, 1999.
[38] Z. Zhao, H. Liu, Searching for interacting features in subset selection, Intelligent Data Analysis 13 (2) (2009) 207–228.
[39] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation. in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, vol. 29, ACM, 2000, pp. 1–12.
[40] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, Machine Learning 11 (1) (1993) 63–90.
[41] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Machine Learning 6 (1) (1991) 37–66.
[42] J. Platt, Sequential minimal optimization: a fast algorithm for training support vector machines, Advances in Kernel Methods: Support Vector Learning 208 (1999) 98–112.
[43] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[44] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings of International Conference on Machine Learning, 1996, pp. 148–156.
[45] C. Soares, P. Brazdil, Zoomed ranking: Selection of classification algorithms based on relevant performance information, Principles of Data Mining and Knowledge Discovery (2000) 160–181.
[46] R.D. King, C. Feng, A. Sutherland, Statlog: comparison of classification algorithms on large real-world problems, Applied Artificial Intelligence 9 (3) (1995) 289–333.
[47] K.A. Smith, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Computing Surveys 41 (1) (2008) 1–25.
[48] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.
[49] A. Hoekstra, R.P.W. Duin, On the nonlinearity of pattern classifiers. in: Proceedings of the 13th International Conference on Pattern Recognition, vol. 4, IEEE, 1996, pp. 271–275.

Qinbao Song received the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2001. He is currently a Professor of software technology in the Department of Computer Science and Technology, Xi'an Jiaotong University, where he is also the Deputy Director of the Department of Computer Science and Technology. He is also with the State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China. He has authored or coauthored more than 80 referred papers in the areas of machine learning and software engineering. He is a board member of the Open Software Engineering Journal. His current research interests include data mining/machine learning, empirical software engineering, and trustworthy software.

Guangtao Wang received the BS degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2007. He is currently a PhD student in the Department of Computer Science and Technology, Xi'an Jiaotong University. His research focuses on feature subset selection and meta-learning.

Chao Wang received the BS degree and master degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2008 and 2011, respectively. His research focuses on classification.