Neurocomputing xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neurocomputing

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

Cost-sensitive KNN classification

Shichao Zhang

School of Information Science and Engineering, Central South University, Changsha 410083, China

ARTICLE INFO

Article history:
Received 30 April 2018
Revised 27 October 2018
Accepted 12 November 2018
Available online xxx

Keywords:
Cost-sensitive
KNN

ABSTRACT

KNN (*K* Nearest Neighbors) classification is one of top-10 data mining algorithms. It is significant to extend KNN classifiers sensitive to costs for imbalanced data classification applications. This paper designs two efficient cost-sensitive KNN classification models, referred to Direct-CS-KNN classifier and Distance-CS-KNN classifier. The two CS-KNN classifiers are further improved with extant strategies, such as smoothing, minimum-cost *k*-value selection, feature selection and ensemble selection. We evaluate our methods with real data sets, to show that our CS-KNN classifiers can significantly reduce misclassification cost.

1. Introduction

Cost-sensitive methods perform optimization in domain-driven learning/mining algorithms by means of balancing the cost of class prediction. It was motivated by classifying clinical diagnosis data that contains skewed class distribution. After that, most of extant classifiers had been improved to addressing the issue of minimizing misclassification cost (Bradford et al., 1998; Turney, 1995); [4,5,7,30,47]. There are also some efforts to minimize multiple costs, such as misclassification cost, test cost, waiting cost and imputation cost [11,35,22,43,44]. From these methods, misclassification cost is incorporated with other costs together for real data classification applications. This provides a practical research direction of cost-sensitive classification under multiple cost environments. To our knowledge, until date, there is no work on improving the KNN (*K* Nearest Neighbors) classification to deal with the cost-sensitive issue, although KNN classification is one of Top-10 KDD algorithms [26,34,45]. In this paper we study the issue of KNN classification sensitive to costs by extending our early research [21].

KNN classifier is a lazy learner without building any models. Unlike the existing model-based classification algorithms (building a model with a given training dataset and predicting any test samples with the built model), KNN classifier needs to keep all the training examples in memory, so as to search for all the *K* nearest neighbours for a test sample. Despite of the KNN classifier for imbalanced data in [31], there is no work on constructing KNN classifier in the field of cost-sensitive learning (called as CSL). The main challenge is how to make the KNN classifier sensitive to costs, because the *K* nearest neighbours is only a small subset of the whole training sample space. In this paper we attack this challenging issue by designing two simple yet efficient classifiers, referred to Direct-CS-KNN classifier and Distance-CS-KNN classifier, for classifying imbalanced data in applications. For efficiency, several enhancement strategies, such as smoothing, minimum-cost *K* value selection, feature selection and ensemble selection, are incorporated with our KNN classifiers. For experimental comparison, the smoothing method is also incorporated with the cost-sensitive C4.5.

Several groups of experiments were conducted to evaluate the efficiency with real datasets, and demonstrated that our cost-sensitive KNN classifiers significantly reduce misclassification cost, compared with extant methods. In particular, the proposed KNN classifiers consistently outperform the cost-sensitive C4.5 (CS-C4.5) on some datasets downloaded from UCI.

In the following sections, the cost-sensitive learning and KNN classification are briefly recalled in Section 2. We present our two new cost-sensitive KNN classifiers in Section 3. The enhancement methods, such as feature selection, probability calibration and ensemble selection, are introduced in Section 4. Experimental study is illustrated in Section 5. We conclude this research in Section 6.

2. Preliminary

2.1. Research into cost-sensitive learning

Cost-sensitive learning is an application-driven data mining method, where training datasets are with imbalanced classes. Consider a training dataset is with two classes, saying positive and negative. Accordingly, misclassification refers to two cases, false positive and false negative. The costs of false positive and false negative are often different in real applications. In many medical diagnosis applications, a false positive misclassification can cause

E-mail address: zhangsc@csu.edu.cn

https://doi.org/10.1016/j.neucom.2018.11.101
0925-2312/

Table 1
Two-class cost matrix.

	Actual negative	Actual positive
Predict negative	C(0, 0) = C_{00}	C(0, 1) = C_{01}
Predict positive	C(1, 0) = C_{10}	C(1, 1) = C_{11}

a life risk cost. And a false negative misclassification leads to a patient spends only some money for medical treatment. Generally, a cost matrix for training datasets with two classes is illustrated as Table 1.

In the above cost matrix, C_{10} stands for the cost of a false positive misclassification (incorrect prediction), and C_{01} is the cost of a false negative misclassification (incorrect prediction). Conceptually, the cost of incorrectly labeling an example should be greater than the cost of incorrectly labeling it. Mathematically, this can be expressed as the cases, $C_{10} > C_{00}$ and $C_{01} > C_{11}$ [5].

According to Elkan (2011), given a cost matrix C, by denoting $C(i,j)$ as the cost of predicting the true class j as class i. If $i \neq j$, then it is a wrong prediction. Otherwise, $i = j$. Formally, we optimize the following total cost to predict the instance or sample x.

$$L(x, i) = \sum_{j=1}^{n} p(j|x)C(i, j) \tag{1}$$

Where $p(j|x)$ is the probability of j given x.

From the literature, there are mainly three research directions to make a classifier cost-sensitive. The first one is to change the distribution of class of training samples, so as to reduce the misclassification cost. The second advocates improving extant learning algorithms sensitive to multiple costs [35]. And the third is to incorporate extant learning methods with the boosting approach [10]. Different from the three direction, a direct cost-sensitive learning algorithm wad designed based on the conditional probability estimation that directly computes the optimal class label for each test sample using cost function [27,28].

More efforts attacking this issue are as follows. Chao et al. [1] designed a data classification algorithm for prognosis of cardiac resynchronization therapy with the speckle-tracking echocardiograms. Domingos [4] designed a general method for making classifiers cost-sensitive, referred to MetaCost. Greiner et al. [6] presented active cost-sensitive classifiers. Li et al. [10] studied a cost-sensitive classification with Genetic programming. Li et al. (2004) proposed a decision tree sensitive to multiple costs. Qin et al. (2005), [18–21] studied some cost-sensitive decision trees for applications, such as training datasets with missing data, semi-supervised classification, multiple-scale costs, and KNN classification. Other cost-sensitive decision trees include, such as [9,22,27–29,38,46].

Most of cost-sensitive learning methods are based on the following assumption. For example, the cost matrix C (i.e., an $M \times M$ matrix) should be available for the issue m-class classification for the training process. And this matrix does not change during the model training or decision tree building. This means that the matrix is generally static in the classification process. An entry $C(i, j)$ is considered when a test sample is predicted to be the class i, but it actually belongs to the class j.

2.2. Research into KNN classification

KNN classification aims at classifying imbalanced data and was selected as top 10 data mining algorithms [26,36,37,39]. There are two main research directions. One is to set a proper K value. Another is the distance function for identifying K nearest neighbors.

For setting K value, a usually-used method is the cross validation in probability theory. It is useful for identifying a proper K value when a training dataset is given. However, training samples are distributed with different densities in the training sample space. This raises a new challenging issue that different samples need different K values for class prediction. Recently, Cheng et al. [2] studied the computation of parameter K for KNN classification, which is an optimal value for each new data. Zhang et al. [37] designed a KNN algorithm with data-driven K parameter computation. Zhang et al. [36] designed an algorithm to efficiently learn K for KNN Classification.

Although there are many distance functions, most KNN classification algorithms use Euclidean distance which is defined as follows.

$$Dist(X, Y) = \sqrt{\sum_{i=1}^{D} (Xi - Yi)^2}$$

Where Xi and Yi ($i = 1 \ldots N$), respectively, are an attribute of two samples/instances X and Y.

Given the selected nearest-neighbours, we use the following voting approaches to classify the test sample:

1. Majority voting: $y' = \arg\max_v \sum_{(xi,yi)\in Dz} \delta(v, yi)$
2. Distance-Weighted Voting: $y' = \arg\max_v \sum_{(xi,yi)\in Dz} w_i \delta(v, yi)$

Where $\delta()$ is an indicator function, D_Z is the set of nearest neighbors of the test sample, and the weight is $w_i = 1/d(x', xi)^2$.

Some lately research reports, for example, Deng et al. [3] pioneered KNN method to classify big data. It first conducts a k-means clustering to separate the whole dataset into several parts. And then, each subset is classified with KNN method. Liu et al. [12] proposed a neighbor selection for multilabel classification. Liu and Zhang [13] studied a noisy data elimination using the mutual KNN for data classification. Zhang [31] proposed to replace the majority rule with CF measure for KNN classification. This leads to a minor class can be become a winner. Zhang [32] studied a shell-neighbor method for KNN classification. It assists in learning from datasets with missing values. Zhang et al. [33] proposed a self-representation nearest neighbor search for data classification. The authors proposed representing each sample by other samples with a new self-reconstruction method. The obtained coefficient is used to compute the value of K for every sample, rather than all samples used in the traditional methods [9,39,45]. Finally, this literature proposed build a decision tree with the obtained value of K in the leaf to output the labels of training samples.

KNN classifiers are lazy learners, which is time consuming since the distance between every test sample and other samples should be calculated. To deal with this issue, Zhang et al. (2018) pioneered a kTree and a k*Tree to use different numbers of nearest neighbors for KNN classification. The kTree method needs less running cost but achieves similar classification accuracy, compared with those KNN methods that assign different K values to different test samples. The k*Tree method is an extension of the kTree. It speeds its test stage by extra storing the information of the training samples in the leaf nodes of kTree, such as the training samples located in the leaf nodes, their KNNs, and the nearest neighbor of these KNNs. It makes KNN only using a subset of training samples in the leaf nodes. This is different from previous methods, e.g., [38,42], which use KNN method to visit all samples. Therefore, our proposed method may decrease the computation cost of the test process.

2.3. Research into direct cost-sensitive classification

From Section 2.2, a classifier outputs conditional probability estimation for training samples and test samples as well. Using this

probability estimation, one can directly compute the optimal class label for each test sample with the cost matrix. This method is referred to direct cost-sensitive (CS) classification [27,28].

From the literature, the exiting CS approaches have a common characteristics. In other words, they do not change neither the classifier's behavior nor control the training examples. Based on that the optimized CS decision criterion directly focus on the output outputted by the classifiers, these CSL algorithms outputs optimal CS prediction.

3. Cost-Sensitive KNN classifiers

For simplifying the description, we consider only binary classification issues in this paper. As claimed in the abstract and introduction, two algorithms are designed for making KNN classifier sensitive to costs (mainly considering the misclassification cost). For efficiency, some enhancement methods are incorporated with our classifiers, aiming at improving the performance of CS KNN classifiers.

3.1. Direct CS KNN classifier

The first CS KNN classifier is very simple yet efficient. The KNN algorithm is applied to training a standard classifier. After all the K nearest neighbors are selected from the training samples, the probability of classes can approximately take its ratio computed with the following formulation.

$$\Pr(i|x) = \frac{k_i}{k}$$

Where k_i is the number of K nearest neighbors for class label i. While the above formula can be used to estimate the probability in Eq. (1), it is easy to compute the optimal class label for each test sample. This approach is referred to *Direct-CS-KNN* classifier.

In standard KNN classifier, K is set to either a fixed value, or generated with the cross validation for a test sample. When the value of K is fixed, the K is often quite small, such as an integer in [1, 12]. Our Direct-CS-KNN classifier aims to minimize the misclassification cost. And it is also much important that the probability estimation is often generated by a KNN classifier. This means, it is very important to set a proper K for any test samples when building a statistically stable CS-KNN classification approach, which can deliver much better probability estimation for reducing the misclassification cost.

In this paper we will consider three test ways of setting the value of K for the Direct-CS-KNN classifier as follows.

- Set a fixed value for the training sample space
- Generate with the cross validation
- Choose a proper K by minimizing the misclassification cost in the training procedure

As having known, the Direct-CS-KNN classifier is easy to understand and implement. However, it must face two main challenges as follows.

- If the value of K is too small, it is statistically unstable to estimate the probability with the KNN approach (k_i/k). This may lead to the issue of over-fitting as well as the increasing of the misclassification cost to some test samples.
- In pratical applications, it is evitable to face noise data in the training samples. And KNN classifier is particularly sensitive to the noisy data. This can lead to that the probability estimation is far from original one.

Fortunately, there are some efficient methods developed for obtaining better probability estimation values from standard classifiers in the field of machine learning. Zadrozny and Elkan [27,28] advocated building an un-pruned decision tree, as well as transforming the scores of its leaves by smoothing them, called m-estimation. Certainly, the m-estimation can be combined to the Direct-CS-KNN classifier. To make the Direct-CS-KNN classifier firstly achieves stable and then decreases misclassification cost, our method in this work proposes two variants to improve the existing m-estimation method. Specifically, the first one is to use the cross-valuation method for determining the value of m at different datasets. It is obvious that such a method is practical than the method with a fixed value for every sample. The second solution is the use of the smoothed probability estimation with the cost matrix when setting the value of K.

Taking the above improvements, the m-estimation formula is similar to that described in Section 2.3. For illustrating the influence of the m-estimation function on the Direct-CS-KNN classifier, we slightly change the existing m-estimation method as follows:

$$\Pr(i|x) = \left(\frac{k}{k+m}\right) \times \left(\frac{k_i}{k}\right) + \left(\frac{m}{k+m}\right) \times b$$

Where k_i is the number of K nearest neighbors at class label i for a test sample, b is the base rate of class distribution, and m is a key parameter that is used to control this correction's the impact power. From this m-estimation, the value of m can balance the relative frequency and the prior probability. This m-estimation impacts on the probability estimation as follows.

- For noisy data, the value of m can be set somewhat higher, so as to weaken the importance of the noisy value in the final estimation, as well as to reduce the impact of noisy data.
- If the value of K is small, the probability estimation of the K nearest neighbors (k_i/k) can be statistically unstable without smoothing method. If the m-estimation is applied to, we can push $k/(k+m)$ and $m/(k+m)$, respectively, approximate to 0 and 1, so that moving the probability estimation close to the base rate (b). This works well on the dataset with skew-class distribution.

In the experiments, we apply our proposed smoothing method to our proposed methods, i.e., the Direct-CS-KNN classifier and the Distance-CS-KNN classifier (will be designed in Section 3.2), compared to the state-of-the-art CS KNN algorithms.

3.2. KNN classifier with CS distance function

The Distance-CS-KNN classifier, as our second method in this work, is proposed to change the distance function of the standard KNN classification. As having known in Section 2.2, the distance-weighted voting function is as follows.

$$y' = \arg\max_{v} \sum_{(xi,yi)\in Dz} w_i \delta(v, yi)$$

We can easily applied this formulation to a binary decision tree. Let Wp be the distance-weight between a test sample and a positive training sample, and Wn be the distance-weight of between same test sample and a negative training sample. If we do not consider the misclassification cost, the top-K training samples with the highest weights will be taken as the nearest neighbors of a test sample, by ignoring the labels. In the cost-sensitive situations, however, the cost of a false positive (FP) prediction can be very different from the cost of a false negative (FN) prediction. Therefore, the selection of nearest neighbors for different class labels may possible lead to various costs.

In our work, we assumed that both the true positive (TP) prediction and true negative (TN) prediction have no costs. Different from error-rate-based KNN classification, the purpose of CSL is to minimize the misclassification cost instead of the error rate of prediction. Consequently, the potential cost (C_p) to select a positive

nearest neighbor is $FP * Wn$ while the potential cost (C_n) to select a negative nearest neighbor is $FN * Wp$. Therefore, K training samples having the lowest cost are regaded as the nearest neighbors of the test sample.

The distance-weighted voting function is combined to standard KNN classification algorithm to generate this new CSL approach, i.e., Distance-CS-KNN classifier. With the modified KNN classifier, one can predict the class label for all test samples. For a test sample, we use the distance-weighted voting and the cost matrix to predict the class label, so as to minimize the misclassification cost of the test sample.

For a given test sample, let Wpa be the total distance-weight of all positive nearest neighbors, and Wna be the total distance-weight of all negative nearest neighbors. They can be calculated with the following formulas.

$$Wpa = W_1 + W_2 + W_3 + \cdots + W_t$$

$$Wna = W_1 + W_2 + W_3 + \cdots + W_j$$

Where t is the number of positive nearest neighbors, and j is the number of negative nearest neighbors.

Consequently, the probability of labeling a test sample to P can be computed with $P_p = Wpa/(Wpa+Wna)$. The probability labeling the test sample to N can be calculated with $P_n = Wna/(Wpa+Wna)$, where $P_p + P_n = 1$. Hereby, the potential cost (C_p) of labeling the test sample to P is $FP * P_n$. And the potential cost (C_n) of labeling the test sample to N is $FN * P_p$.

If $C_p > C_n$, the test sample can be classified as N, and the probability of this prediction is $C_p/(C_p + C_n)$. Otherwise, the test sample is classified as P, and the probability of this prediction is $C_n/(C_p + C_n)$. This approach is referred to Distance-CS-KNN classifier.

3.3. Some potential and challenges

This section analyzes the potential and challenges for extending the KNN classification sensitive to costs.

A number of literature has conducted experiments to compare the difference between the Direct-CS-KNN classifier and the CS-C4.5 classifier on real datasets. In most of datasets, the best result of the Direct-CS-KNN classifier outperforms the CS-C4.5 classifier on both of the minimal cost and AUC measurements. We reported some of them in Table 4 in Section 5.2, i.e., the results were marked with gray color in Table 4). As aforementioned, KNN classifier does not outperform the CSL.

Unfortunately, it is changeling for applying the KNN classifier to the CS environments. Firstly, setting K value is still an open issue in machine learning and data mining. Secondly, it is the time consuming for tuning the value of K.

There exist many research reports on attacking these challenges in KNN methods, which can be modified for the Direct-CS-KNN classifier and Distance-CS-KNN classifier, as well as for their variations. Moreover, additional enhancements will be incorporated with our Direct-CS-KNN classifier, which make search a good value of K easily, so that decreasing the tuning cost. More detail will be introduced in the following section.

4. Some enhancement methods

4.1. Calibration methods for improving probability estimation

Generally, if a classifier can well estimate the conditional probability of training samples, it can also estimate well the conditional probability estimation of test samples. If a trained model did not calculate the probability explicitly, most of classifiers can be modified to output some values that reflect the internal class probability [14]. With such estimated probability, the optimal class label for every test sample can directly be obtained with the given cost matrix.

In real application, many providers of the probability estimations are error-based learners. That is, they are biased or bad calibrated, such as C4.5 decision tree and KNN. The existing methods include the Laplace correction method and the m-smoothing method [23,27,28,17].

4.1.1. Laplace correction

Provost and Domingos [17] proposed comparing different pruning methods on the C4.5 classifier, and then demonstrated that it is not correct for using the Laplace correction method to approximate the class probability at leaf nodes. Instead, the author suggested to have the process of no pruning the decision tree. The Laplace correction method basically corrects the probability values by shifting them towards 0.5, in a two-class problem. This correction has successfully been applied to deal with the over-fitting issue in [23].

4.1.2. m-Smoothing

Zadrozny and Elkan [27,28] proposed the use of an un-pruned decision tree and transformed the scores of the leave nodes by smoothing them. They thought that the Laplace correction method does not work well for datasets with a skewed class distribution. Also, they suggested using a smoothing method, called m-estimation, to replace the Laplace correction method. According to that method, the class probability values can be calculated with the following formulas.

$$\Pr(i|x) = \frac{Ni + b * m}{N + m}$$

Where N indicates sample size, N_i implies the sample size of the ith class, b represents the frequency of the minority class, and m controls the percentage of correction influence. In the literature, the value of m is often set by satisfying the following constraint $b \times m = 10$. The literature in Zadrozny and Elkan [27,28] demonstrated that the direct CSL including the m-estimation may achieve less misclassification cost, compared to the method in MetaCost [4], while the C4.5 decision tree classifier was used for the base learner at the KDD-1998 datasets. We list an example to illustrate the m-smoothing as follows. We assume that there are 5 training samples in the leaf where include 2 positive samples and 3 negative samples. In this case, the probability of a test sample assigned to the leaf is 0.2 while using original C4.5 decision tree method. However, this can be obtain by the smoothed score as follows (assuming $m = 200$ and $b = 0.05$).

$$P' = (2 + 0.05 \times 200)/(5 + 200) = 0.058$$

By this way, the smoothed score is changes to the base rate of KDD-98 data set. That is, the smoothed score is equivalent to the base rate.

After Zadrozny and Elkan [27,28], Wang et al. [23] have successfully applied the m-smoothing to attack the over-fitting issue. There are also many improvements developed with different classifiers. For example, Niculescu-Mizil and Caruana [15] experimented two other ways of correcting the poor probability estimation predicted by decision tree, SVM and other error based classifiers: Platt Scaling [16] and Isotonic Regression. These methods can also be used in directly CSL algorithms.

4.1.3. Platt scaling

Different from the m-smoothing, Platt [16] proposed passing SVM predictions by a sigmoid function to posterior probabilities. Specifically, assuming the output of a learning method as $f(x)$, we

normalize the output with the following sigmoid to obtain their calibrated probabilities:

$$P(y = 1|f) = 1/(1 + exp(Af + B))$$

Where both A and B are used maximum likelihood estimation to conduct fitting from a fitting training set (f_i, y_i). In this process, we use gradient method to search for both A and B. Platt Scaling is effective for sigmoid-shaped distortion in the predicted probabilities [15]. Wang et al. [23] have successfully applied it to overcome the issue of over-fitting.

4.1.4. Isotonic regression

Different from Platt Calibration, isotonic regression may correct monotonic distortion [15]. Zadrozny and Elkan [27–29] have use it to conduct probability estimation from the classifiers, such as SVM, Naive Bayes and decision tree classifiers. The only limitation of isotonic regression is that it mapping function should be monotonically increasing. Specifically, given f_i from a model and the true targets y_i, the basic assumption of isotonic regression can be expressed as.

$$y_i = m(f_i) + \epsilon_i$$

Where $m(.)$ is an isotonic function. Given a train set (f_i, y_i), the issue of Isotonic Regression can be changed to find the isotonic function $m'(.)$ as follows.

$$m' = \arg\min z \sum (yi - z(fi))^2$$

The Isotonic Regression has successfully been applied to deal with the over-fitting issue in [23].

4.2. Feature selection

Feature selection, also called variable subset selection, or attribute selection, has been an efficient method of reducing the complexity for data mining and machine learning applications. It is often taken as a part of data preprocessing stage of KDD. Therefore, many classification algorithms, such as the nearest neighbor classifier and the decision tree classifier, can be benefited from an efficient feature selection procedure. The benefit is apparent because real datasets often contain missing values, referred to noisy data, and some of attributes are irrelevant or distracting. This can lead to the issues of data over-fitting and poor predication accuracy on test sample.

There are many nice feature selection approaches designed for enhancing practical algorithms in data mining and machine learning, as well as in Statistics and Pattern Recognition. Two usually-used approaches are the filter approach and the wrapper method. The filter algorithms select the important features with a preprocessing step, which is independent of the induction algorithm. The drawback of this approach is that it totally ignores the influence of the selected features [8].

Mahy previous feature selection methods [40,41] focused on improving the model accuracy. To our knowledge, no literature focused on the influence of feature selection to improve the performance of cost-sensitive classifiers. Specifically, the feature selection approach should efficiently remove the noisy data. This leads to our CS KNN classifiers can find K much better nearest neighbors, i.e., they are with the most relevant attributes that assists in minimizing the misclassification cost.

4.3. Ensemble learning method

Ensemble is a strategy of improving the efficiency of data mining algorithms. It is also referred to the Committee Method, or the Model Combiner. Ensemble learning integrates a set of models to enhance the prediction accuracy that can win anyone in the set of the data mining algorithms. The procedure of the ensemble learning is as follows.

Some base models are first constructed with improving extant data mining and machine learning algorithms. And then, a certain scoring function is designed for model selection, so as to obtain a subset from these base models. This model selection is formally described as follows.

1. Starting with null ensemble;
2. Adding the best model to the ensemble in the library which maximize the ensemble's performance via a hill-climb set;
3. Repeat Step 2 until all base models have been examined;
4. Return that subset of base models that yields maximum performance on the hill-climb set.

Generally, ensemble classification algorithms can generate many models. Given a test sample, the ensemble method first test every built base models, and then each base model outputs a prediction. Furthermore, all predictive results are synthesized with a majority rule. The ensemble method (including bagging, boosting and stacking) is more accurate and stable, than an individual classifier.

It is time consuming to evaluate the prediction of an ensemble typically, compared to evaluating the prediction of an individual model. Consequently, an ensemble classification algorithm is viewed as a way of compensating the drawback of an individual classifier (e.g., poor learning performance) by conducting a number of computation.

4.4. KNN with CS feature selection

Training datasets from real application are often dirty. There are some examples with missing values, referred to noisy data, and some attributes may be irrelevant, or disturbing. A main shortcoming of lazy learning (KNN classification) algorithm is often sensitive to the noisy data and the irrelevant, or disturbing attributes. Due to the curse of "garbage in and garbage out", low-quality data often delivers poor classification performance on test sample. Therefore, efficient feature selection strategies certainly help in classifying real data.

It is true there have been many efficient feature selection methods developed for data mining. We can roughly divide them into two categories, the filter approach and the wrapper approach. Kohavi and John [8] presented a nice wrapper algorithm that generally performs better than extant filter approaches. And the accuracy was significantly improved.

In [8], the wrapper approach conducts a search in the space of possible parameters. The search requires a state space, an initial state, a search engine and a termination condition. The search aims to find the state with the highest evaluation, using a heuristic function to guide it. To construct any one of the heuristic function and the evaluation function, Kohavi and John advocated adopting the prediction accuracy estimation. Two famous search strategies, the hill-climbing and best-first, were employed and concluded that the best-first search strategy is the winner between the hill-climbing and best-first.

This wrapper approach is applied to enhance the proposed Direct-CS-KNN classifier and Distance-CS-KNN classifier, so as to improve the performance. Because the ultimate goal of CSL is to minimize the misclassification cost, Kohavi and John's wrapper approach cannot simply be applied to our CS KNN classifiers. In this paper, a variation of Kohavi and John's wrapper is designed for feature selection. The main difference is that the new wrapper approach replaces the error rate with misclassification cost in both the heuristic function and the evaluation function.

Our experiment settings for the new wrapper approach are similar to that in [8]. The search space is a set of states and each of

states stands for a feature subset. Consequently, each state has n bits for a set of training examples with n attributes. And a bit is used to indicate the selection state, i.e., 1 means the feature is chosen, 0 means no. This setup aims at reducing the computational complexity. This leads to faster search for the K nearest neighbors using lesser features of the training dataset. The best-first search algorithm is taken as our search engine. For a simple dataset with three features, the setting of the proposed CS feature selection is summarized as follows.

State:	a Boolean vector, one bit for each feature
Initial state:	an empty set of features (0,0,0)
Search	(0,0,0) (0,1,0) (1,0,0) (0,0,1)
space:	(1,1,0) (0,1,1) (1,0,1) (1,1,1)
Search engine:	Best first
Evaluation function:	Misclassification Cost

4.5. KNN with CS stacking

Although a dissimilarity measure can simply be constructed for the set of observations, setting the value of K could be a challenging issue, especially in the research topic of CSL. To attack this challenge, we proposes an ensemble method in this paper, referred to CS stacking, so as to obtain a better parameter K.

As well known, the ensemble selection has been well studied and there is a very popular method in data mining and machine learning. It intends to obtain those better features with integrating multiple feature selection algorithms. A famous ensemble technique is the Stacking method that is designed for generalizing the accuracy, aiming to improve the performance. Both the in-sample and out-of-sample strategies are applied to identify the best guesser in this paper. This paper uses the single generalization method in [25] for combining it to our Direct-CS-KNN classifier and Distance-CS-KNN classifier.

Our CS stacking method contains four steps as follows.

1. Stack ← CS KNN classifiers, the K can be with different values
2. Training a model on each classifier;
3. Computing the probability of class for each sample with its votes in these classifiers;
4. *Re*-labeling each sample with the class with the above computed results;
5. Applying the classification algorithm to the above relabeled samples.

The above procedure is a variant of the MetaCost developed in [4], which is a bagging approach.

5. Experiments and analysis

This section examines our Direct-CS-KNN classifier and Distance-CS-KNN classifier with a set of experiments. And the performance is simply analyzed.

5.1. Experiment setup

The experiments are conducted for evaluating the performance of both the Direct-CS-KNN classifier and Distance-CS-KNN classifier with the feature selection and stacking. Their misclassification costs are compared with other key performance measurements, such as Area Under the Curve (AUC) over different cost ratios (FN/FP) is compared to other classification method (e.g., C4.5 and its invariants). All these classification methods are implemented using the public software Weka [24]. We summarize them in the following Table 2.

Note that three experiments will be conducted with the above algorithms, and six datasets are downloaded from UCI repository. The details of these datasets are listed in Table 3.

The criteria for selecting the datasets is listed as follows.

1. We selected the datasets with two classes due to our assumption in this paper. Moreover, the CS KNN classification algorithms in this paper designed for only handling the cost issue in the minority class. This condition is hard to satisfy and we resorted to converting several multi-class datasets into two-class datasets by choosing the minority class as the positive class and union all other classes as the negative class. For two-class datasets, the minority class is always assigned as the positive class, and the majority class is assigned as the negative class.
2. Our experiments contain complete data. We remove the missing data for the dataset with missing values. To do this, we employed the "Replacing-Missing-Values" filter in Weka [35].
3. The class distribution of the used datasets is either balanced or unbalanced. The ratio of major class size to minor class size (called imbalance level) in the used datasets varied from 1.02 (Waveform-5000) to 8.8 (Page-blocks).

We conduct all the experiments by compared our proposed methods with the comparison methods on the used datasets.

We used a UCI dataset Statlog (heart) in our first experiment, where the classifier performance is evaluated by two evaluation metrics, such as misclassification cost and AUC generated. Specifically, Statlog (heart) is a dataset containing recommended cost matrix. In our experiment, we normalized the cost matrix is normalized and set the value of the cost ratio (FN/FP) and TP and TN, respectively, as 5 and 0.

We also used the Statlog (heart) dataset as the testing samples in the second experiment, is still used to conduct the test. We employed misclassification cost and AUC as the evaluation metrics of our proposed two CS KNN classifiers, i.e., Direct-CS-KNN and Distance-CS-KNN.

We used five UCI datasets in the third experiment, where the misclassification cost FP, FN, respectively, are set to 1, and an integer in the set of (2, 5, 10, 20). Specifically, we assumed that a high cost incurs for the misclassification of the minority class, which can be often in real applications in which the less frequent class is more important than the major class. Moreover, we also set the cost of TP and TN as 0. We evaluated our Distance-CS-KNN classifier (and its variations), compared to the CS-C4.5 classifier in terms of average misclassification cost.

All of the three experiments are repeated for 10 times and ten-folder cross validation method is used in all tests to prevent the over-fitting data.

5.2. Experimental results and analysis

In this section we experimentally compare our CS KNN classifiers with other competing algorithms. For the sake of easy reading, understanding and discussing, all the results without enhancement methods are marked with gray background on all tables.

Experiment 1. This experiment is designed to examine the impact of the enhancement measures on our CS KNN classifiers. Table 4 lists the summary results. The average misclassification cost and AUC are criterion factors. And the UCI dataset Statlog(heart) is the training examples. The Direct-CS-KNN classifier is improved with incorporated with the smoothing (simply denoted as Direct-CS-KNN-SM classifier) and setting K with minimum-cost (called as Direct-CS-KNN–CSK classifier) approach. In Experiment 1, a fixed K value ($K = 5$) is set for both the Direct-CS-KNN classifier and the Direct-CS-KNN-SM classifier, and automatically setting K with minimum-cost (on training examples) is for the Direct-CS-KNN–CSK classifier.

As having seen in the results of the first experiment, to reduce the misclassification cost of classifiers, our cost-sensitive KNN

Table 2
List of CS algorithms and abbreviations.

#	Method	Abbreviation	Base classifier
1	Direct CS KNN	Direct-CS-*KNN*	*KNN*
2	Direct CS KNN with Smoothing	Direct-CS-*KNN*-SM	*KNN*
3	Direct CS KNN with *K* value selection	Direct-CS-*KNN*-CSK	*KNN*
4	Distance CS KNN	Distance-CS-*KNN*	*KNN*
5	Distance CS KNN with Feature Selection	Distance-CS-*KNN*-FS	*KNN*
6	Distance CS KNN with Stacking	Distance-CS-*KNN*-STK	*KNN*
7	C4.5 with Minimum Expected Cost	CS-C4.5	C4.5
8	C4.5 with Minimum Expected cost and smoothing	CS-C4.5-SM	C4.5

Table 3
Summary of the data set characteristics.

Dataset	No. of attributes	No. of samples	Class distribution (P/N)
Statlog (heart)	14	270	120/150
Credit-g	21	1000	300/700
Diabetes	9	768	268/500
Page-blocks	11	5473	560/4913
Spambase	58	4601	1813/2788
Waveform-5000	41	3347	1655/1692

Table 4
Key performance measurements on Statlog(heart).

(a) Average misclassification cost

Data Set	Direct-CS-KNN	Direct-CS-KNN-SM	Direct-CS-KNN-CSK	CS-C4.5	CS-C4.5-SM
Statlog(heart)	0.3815	0.3605	0.3556	0.6704	0.4938

(b) Area under ROC (AUC)

Data Set	Direct-CS-KNN	Direct-CS-KNN-SM	Direct-CS-KNN-CSK	CS-C4.5	CS-C4.5-SM
Statlog(heart)	0.744	0.763	0.768	0.759	0.776

classifiers all are winners, compared with the CS-C4.5 classifier. Consider the CS-C4.5 classifier as the benchmark, the Direct-CS-KNN classifier has reduced the misclassification cost by 43%. In particular, the Direct-CS-KNN-SM classifier and Direct-CS-KNN–CSK classifier have both reduced the misclassification cost by more than 46%.

Considering the AUC measure, the proposed Direct-CS-KNN-SM classifier and Direct-CS-KNN–CSK classifier have the AUC much higher than the benchmark, CS-C4.5 classifier. However, our Direct-CS-KNN classifier has the AUC slightly lower than the CS-C4.5 classifier.

In addition, the Direct-CS-KNN-SM classifier and Direct-CS-KNN–CSK classifier always perform better among the four examined classification algorithms, in terms of carrying out a lower misclassification cost, as well as a higher AUC. Consequently, the Direct-CS-KNN–CSK classifier is the best among the four examined classification algorithms.

Experiment 2. The Statlog(heart) dataset is still adopted in the second experiment with the cost-matrix. The main task is to examine the efficiency of our Direct-CS-KNN classifier and Distance-CS-KNN classifier. The misclassification cost and AUC will be the test objects. The Direct-CS-KNN classifier with the smoothing and setting *K* with minimum-cost methods has well reduced the misclassification cost in our first experiment. In this experiment we also apply the smoothing and setting *K* with minimum-cost methods to the Direct-CS-KNN classifier and Distance-CS-KNN classifier, so as to further improve the efficiency. Table 5 lists the examined results as follows.

Experiment 3. We design the second experiment in a simple yet straightforward way, so as to examine the modified CS KNN classification, i.e., the Distance-CS-KNN classifier, whether or not it is more efficient than its original version, the Direct-CS-KNN classifier. Based on Experiment 2, the third experiment is focused on examining the Distance-CS-KNN classifier.

Table 5
Key performance measurements on Statlog(heart).

(a) Average misclassification cost

Dataset	Direct-CS-KNN	Distance-CS-KNN
Statlog(heart)	0.3512	0.344

(b) ROC (AUC) measure

Dataset	Direct-CS-KNN	Distance-CS-KNN
Statlog(heart)	0.7642	0.7688

We reported the results of the test samples in our third experiment Tables 6 and 7. Tables 6 and 7, respectively, listed the average misclassification cost on five UCI data sets, and the corresponding results on the *t*-test.

Based on the experimental results, our CS KNN classifiers outperformed the CS-C4.5 classifier at all the datasets. Moreover, the improvement is larger with the increase of the cost ratio. The reason is that the CS-C4.5 classifier builds the decision tree classifier by taking the cost at the stage of classification into account and ignoring the misclassification cost. On the contrary, our proposed method considered the misclassification cost at both the classification stage and the stage of calculating distance weight. Also, on most of the selected UCI datasets across different cost ratios, it is important that Distance-CS-KNN-FS classifier and Distance-CS-KNN-STK classifier outperform their original Distance-CS-KNN classifier. In addition, from the examined four classification algorithms, the Distance-CS-KNN-STK classifier is the best one. The Distance-CS-KNN-STK classifier is also very stable and performs better than other competing algorithms across different cost ratios.

Table 6
Averaging the misclassification cost on the selected UCI datasets.

(a) Cost ratio with FP = 1 and FN = 2

Dataset	Distance-CS-KNN	Distance-CS-KNN-FS	Distance-CS-KNN-STK	CS-C4.5	CS-C4.5-SM
Diabetes	0.3758	0.3596	0.3633	0.3828	0.3722
Credit-g	0.428	0.417	0.402	0.435	0.408
Page-blocks	0.0607	0.0592	0.0585	0.0422	0.0397
Spambase	0.1091	0.1044	0.0993	0.1052	0.0973
Waveform-5000	0.1341	0.1315	0.1298	0.1951	0.1933

(b) Cost ratio with FP = 1 and FN = 5

Dataset	Distance-CS-KNN	Distance-CS-KNN-FS	Distance-CS-KNN-STK	CS-C4.5	CS-C4.5-SM
Diabetes	0.5573	0.536	0.5352	0.6003	0.5789
Credit-g	0.598	0.5815	0.582	0.77	0.681
Page-blocks	0.0965	0.0896	0.0838	0.0846	0.0767
Spambase	0.2006	0.1937	0.1864	0.2121	0.1905
Waveform-5000	0.1637	0.1596	0.1562	0.3756	0.3254

(c) Cost ratio with FP = 1 and FN = 10

Dataset	Distance-CS-KNN	Distance-CS-KNN-FS	Distance-CS-KNN-STK	CS-C4.5	CS-C4.5-SM
Diabetes	0.5898	0.5832	0.5869	0.8268	0.7642
Credit-g	0.717	0.6756	0.62	1.043	0.832
Page-blocks	0.1214	0.1163	0.1031	0.1297	0.1108
Spambase	0.3019	0.2836	0.2712	0.3512	0.3122
Waveform-5000	0.1933	0.1896	0.1815	0.611	0.5752

(c) Cost ratio with FP = 1 and FN = 20

Dataset	Distance-CS-KNN	Distance-CS-KNN-FS	Distance-CS-KNN-STK	CS-C4.5	CS-C4.5-SM
Diabetes	0.7591	0.725	0.717	0.9635	0.8281
Credit-g	0.939	0.822	0.8161	1.258	1.035
Page-blocks	0.1782	0.1665	0.1546	0.1838	0.1633
Spambase	0.4027	0.3817	0.3552	0.5781	0.4842
Waveform-5000	0.1963	0.1915	0.1848	1.0678	0.9912

Table 7
T-test summary.

Cost Ratio (FP:FN)		CS-4.5-CS
1:2	Distance-CS-KNN	3/0/2
	Distance-CS-KNN-FS	4/0/1
	Distance-CS-KNN-STK	4/0/1
1:5	Distance-CS-KNN	4/0/1
	Distance-CS-KNN-FS	4/0/1
	Distance-CS-KNN-STK	5/0/0
1:10	Distance-CS-KNN	5/0/0
	Distance-CS-KNN-FS	5/0/0
	Distance-CS-KNN-STK	5/0/0
1:20	Distance-CS-KNN	5/0/0
	Distance-CS-KNN-FS	5/0/0
	Distance-CS-KNN-STK	5/0/0

6. Conclusion and future work

This paper has presented two approaches, named as Direct-CS-KNN classifier and Distance-CS-KNN classifier aim at making KNN classification sensitive to costs, so as to minimize the misclassification cost. For efficiency, several useful methods, including the smoothing, setting K with minimum-cost, CS feature selection and CS stacking, are significantly combined to our CS KNN classifiers. Sets of experiments have been conducted to evaluate the efficiency, and demonstrate that the proposed CS KNN classifiers can minimize the misclassification cost very well, compared to the CS-C4.5 classifier on the selected UCI data across different cost ratios.

Our future work is mainly focused on examining the Direct-CS-KNN classifier and Distance-CS-KNN classifier with real datasets. Also, some calibration methods, such as Platt Scaling and Isotonic Regression, may be applied to obtain better probability estimation for class memberships. And the Genetic algorithm may be employed to be with CS fitness function for improving the proposed feature selection wrapper.

Acknowledgement

This research is partly supported by the China “1000-Plan” National Distinguished Professorship; the China Key Research Program (Grant no: 2016YFB1000905); the Key Program of the National Natural Science Foundation of China (Grant no. 61836016), the Natural Science Foundation of China (Grant no: 61672177); the Project of Guangxi Science and Technology (GuiKeAD17195062); and the Guangxi Collaborative Innovation Center of Multi-Source Information Integration and Intelligent Processing.

References

[1] P. Chao, et al., An intelligent classifier for prognosis of cardiac resynchronization therapy based on speckle-tracking echocardiograms, Artif. Intell. Med. 54 (3) (2012) 181–188.
[2] D. Cheng, S. Zhang, Z. Deng, Y. Zhu, M. Zong, KNN algorithm with data–driven k value, in: Proceedings of International Conference on Advanced Data Mining and Applications, ADMA-2014, Guilin, China, Dec 19–21, 2014, 2014, pp. 499–512.
[3] Z. Deng, X. Zhu, D. Cheng, M. Zong, S. Zhang, Efficient KNN classification algorithm for big data, Neurocomputing 195 (2016) 143–148.
[4] P. Domingos, MetaCost: a general method for making classifiers cost-sensitive, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 155–164.
[5] C. Elkan, The foundations of cost-sensitive learning, in: Proceeding of the Seventeenth International Joint Conference of Artificial Intelligence, 2001, pp. 973–978.
[6] R. Greiner, A.J. Grove, D. Roth, Learning cost-sensitive active classifiers, Artif. Intell. 139 (2) (2002) 137–174.
[7] R. Hu, X. Zhu, D. Cheng, W. He, Y. Yan, J. Song, S. Zhang, Graph self-representation method for unsupervised feature selection, Neurocomputing 220 (2017) 130–137.
[8] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. 97 (1–2) (1997) 273–324.

[9] C. Lei, X. Zhu, Unsupervised feature selection via local structure learning and sparse learning, Multimed. Tools Appl. (2017), doi:10.1007/s11042-017-5381-7.
[10] J. Li, X. Li, X. Yao, Cost-sensitive classification with genetic programming, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, 3, 2005.
[11] C. Ling, Q. Yang, J. Wang, S. Zhang, Decision trees with minimal costs, in: Proceeding of the Twenty First International Conference on Machine Learning, 4–8 July 2004, 2004, pp. 69–76.
[12] H. Liu, X. Li, S. Zhang, Learning instance correlation functions for multi-label classification, IEEE Trans. Cybern. 47 (2) (2016) 499–510.
[13] H. Liu, S. Zhang, Noisy data elimination using mutual k-nearest neighbor for classification mining, J. Syst. Softw. 85 (2012) (2012) 1067–1074.
[14] D. Margineantu, Methods for Cost-Sensitive Learning, Oregon State University, 2001.
[15] A. Niculescu-Mizil, R. Caruana, Predicting Good Probabilities with Supervised Learning, Association for Computing Machinery, Inc, One Astor Plaza, 2005.
[16] J. Platt, Probabilities for SV machines, Adv. Neural Inf. Process. Syst. (1999) 61–74.
[17] F. Provost, P. Domingos, Tree induction for probability-based ranking, Mach. Learn. 52 (2003) 199–215.
[18] Z. Qin, S. Zhang, L. Liu, T. Wang, Cost-sensitive semi-supervised classification using CS-EM, in: Proceedings of the IEEE Eighth International Conference on Computer and Information Technology, 2008, pp. 131–136.
[19] Z. Qin, C. Zhang, T. Wang, S. Zhang, Cost sensitive classification in data mining, in: Proceedings of International Conference on Advanced Data Mining and Applications (ADMA-2010), 2010, pp. 1–11.
[20] Z. Qin, T. Wang, S. Zhang, Incorporating medical history to cost sensitive classification with lazy learning strategy, in: Proceedings of International Conference on Progress in Informatics and Computing conference (PIC-2010), 2010, pp. 19–23.
[21] Z. Qin, T. Wang, C. Zhang, S. Zhang, Cost-sensitive classification with k-nearest neighbors, in: Proceedings of International Conference on Knowledge Science, Engineering and Management, 2013, pp. 112–131.
[22] V. Sheng, C. Ling, A. Ni, S. Zhang, Cost-sensitive test strategies, in: Proceedings of Twenty-first National Conference on Artificial Intelligence (AAAI-06), 2011, pp. 482–487.
[23] T. Wang, Z. Qin, Z. Jin, S. Zhang, Handling over-fitting in test cost-sensitive decision tree learning by feature selection, smoothing and pruning, J. Syst. Softw. (JSS) 83 (7) (2010) 1137–1147.
[24] I. Witten, E. Frank, Data Mining: Practical Machine Learning Techniques With Java Implementations, 2nd edition, Morgan Kaufmann Publishers, 2000.
[25] D. Wolpert, Stacked generalization, Neural Netw. 5 (1992) 241–259.
[26] X. Wu, V. Kumar, J. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[27] B. Zadrozny, C. Elkan, Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers, in: Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 609–616.
[28] B. Zadrozny, C. Elkan, Learning and making decisions when costs and probabilities are both unknown, in: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 204–213.
[29] B. Zadrozny, C. Elkan, Transforming Classifier Scores into Accurate Multiclass Probability Estimates, ACM Press, New York, NY, USA, 2002, pp. 694–699.
[30] B. Zadrozny, One-benefit learning: cost-sensitive learning with restricted cost information, in: Proceedings of the First international Workshop on Utility-based Data Mining, ACM Press, Chicago, Illinois, 2005, pp. 53–58.
[31] S. Zhang, KNN-CF approach: incorporating certainty factor to KNN classification, IEEE Intell. Inf. Bull. 11 (1) (2010) 24–33.
[32] S. Zhang, Shell-neighbor method and its application in missing data imputation, Appl. Intell. 35 (1) (2011) 123–133.
[33] S. Zhang, D. Cheng, M. Zong, L Gao, Self-representation nearest neighbor search for classification, Neurocomputing 195 (2016) 137–142.
[34] S. Zhang, D. Cheng, Z. Deng, M. Zong, X. Deng, A novel KNN algorithm with data-driven k parameter computation, Patt. Recognit. Lett. 109 (2018) 44–54.
[35] S. Zhang, Z. Qin, C. Ling, S. Sheng, "Missing is useful": missing values in cost-sensitive decision trees, IEEE Trans. Knowl. Data Eng. 17 (12) (2005) 1689–1693.
[36] S. Zhang, X. Li, M. Zong, X. Zhu, D. Cheng, Learning k for KNN classification, ACM Trans. Intell. Syst. Technol. 8 ((3) 43) (2017) 1–19.
[37] S. Zhang, X. Li, M. Zong, X. Zhu, R. Wang, Efficient KNN classification with different numbers of nearest neighbors, IEEE Trans. Neural Netw. Learn. Syst. 29 (5) (2018) 1774–1785.
[38] W. Zheng, X. Zhu, G. Wen, Y. Zhu, H. Yu, J. Gan, Unsupervised feature selection by self-paced learning regularization, Patt. Recognit. Lett. (2018), doi:10.1016/j.patrec.2018.06.029.
[39] W. Zheng, X. Zhu, Y. Zhu, R. Hu, C. Lei, Dynamic graph learning for spectral feature selection, Multimed. Tools Appl. (2017), doi:10.1007/s11042-017-5272-y.
[40] X. Zhu, Z. Huang, Y. Yang, H.T. Shen, C. Xu, J. Luo, Self-taught dimensionality reduction on the high-dimensional small-sized data, Patt. Recognit. 46 (1) (2013) 215–229.
[41] X. Zhu, Z. Huang, H.T. Shen, J. Cheng, C. Xu, Dimensionality reduction by mixed kernel canonical correlation analysis, Patt. Recognit. 45 (8) (2012) 3003–3016.
[42] X. Zhu, L. Zhang, Z. Huang, A sparse embedding and least variance encoding approach to hashing, IEEE Trans. Image Process. 23 (9) (2014) 3737–3750 (2014).
[43] X. Zhu, X. Li, S. Zhang, Z. Xu, L. Yu, C. Wang, Graph PCA hashing for similarity search, IEEE Trans. Multimed. 19 (9) (2017) 2033–2044.
[44] X. Zhu, X. Li, S. Zhang, C. Ju, X. Wu, Robust joint graph sparse coding for unsupervised spectral feature selection, IEEE Trans. Neural Netw. Learn. Syst. 28 (6) (2017) 1263–1275.
[45] X. Zhu, S. Zhang, Y. Li, Y. Zhu, Local and Global structure preservation for robust unsupervised spectral feature selection, IEEE Trans. Knowl. Data Eng. 30 (3) (2018) 517–529.
[46] X. Zhu, S. Zhang, Y. Li, L. Yang, Y. Fang, Low-rank sparse subspace for spectral clustering, IEEE Trans. Knowl. Data Eng. (2018) Accepted in 14 July 2018, doi:10.1109/TKDE.2018.2858782.
[47] X. Zhu, S. Zhang, W. He, R. Hu, C. Lei, P. Zhu, One-step multi-view spectral clustering, IEEE Trans. Knowl. Data Eng. 2018 (2018), doi:10.1109/TKDE.2018.2873378.

Further reading

H. Escalante, et al., Acute leukemia classification by ensemble particle swarm model selection, Artif. Intell. Med. 55 (3) (2012) 163–175.
S. Kotsiantis, P. Pintelas, A cost sensitive technique for ordinal classification problems, Methods Appl. Artif. Intel. (2004) 220–229.
S. Kotsiantis, D. Kanellopoulos, P. Pintelas, Handling imbalanced datasets: a review, GESTS Int. Trans. Comput. Sci. Eng. 30 (1) (2006) 25–36.
C. Ling, S. Sheng, T. Bruckhaus, NH. Madhavji, Predicting software escalations with maximum ROI, in: Proceedings of the Fifth IEEE International Conference on Data Mining, 2005, pp. 717–720.
H. Liu, X. Wu, S. Zhang, Neighbor selection for multilabel classification, Neurocomputing 182 (2016) 187–196.
N. Oza, Ensemble Data Mining Methods, NASA Ame Research Center, 2000.
Z. Qin, C. Zhang, S. Zhang, Cost-sensitive decision trees with multiple cost scales, in: Proceedings of the Seventeenth Australian Joint Conference on Artificial Intelligence (AI 2004), 2004, pp. 380–390.
Z. Qin, C. Zhang, S. Zhang, Missing or absent? A question in cost-sensitive decision tree, in: Proceedings of the Fourth International Conference on Active Media Technology (AMT006), 2006, pp. 118–125.
J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, 1993.
Q. Sun, B. Pfahringer, Bagging ensemble selection, AI 2011: Adv. Artif. Intell. (2011) 251–260.
P. Turney, Types of cost in inductive concept learning, in: Proceedings of Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning, 2000, p. 1511.
D. Wettschereck, D.W. Aha, T. Mohri, A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, Artif. Intell. Rev. 11 (1) (1997) 273–314.
J. Zhang, I. Mani, KNN approach to unbalanced data distributions: a case study involving information extraction, in: Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets, 2003.

Shichao Zhang is a Distinguished Professor and the director of Institute of School of Computer Science and Information Technology at the Guangxi Normal University, Guilin, China. He holds a Ph.D. degree in Computer Science from Deakin University, Australia. His research interests include data analysis and smart pattern discovery. He has published over 50 international journal papers and over 60 international conference papers. He has won over 10 nation-class grants, such as the China NSF, China 863 Program, China 973 Program, and Australia Large ARC. He is an Editor-in-Chief for International Journal of Information Quality and Computing, and is served as an associate editor for IEEE Transactions on Knowledge and Data Engineering, Knowledge and Information Systems, and IEEE Intelligent Informatics Bulletin.