Challenges in KNN Classification

Shichao Zhang, *Senior Member, IEEE*

Abstract—The KNN algorithm is one of the most popular data mining algorithms. It has been widely and successfully applied to data analysis applications across a variety of research topics in computer science. This paper illustrates that, despite its success, there remain many challenges in KNN classification, including K computation, nearest neighbor selection, nearest neighbor search and classification rules. Having established these issues, recent approaches to their resolution are examined in more detail, thereby providing a potential roadmap for ongoing KNN-related research, as well as some new classification rules regarding how to tackle the issue of training sample imbalance. To evaluate the proposed approaches, some experiments were conducted with 15 UCI benchmark datasets.

Index Terms—Data mining, lazy learning, KNN classification, classification rule

1 Introduction

NN (nearest neighbor) classification is an efficient solution to approximation, which was first proposed as a nonparametric discrimination in statistics [1]. However, it has long suffered from the key issue of overfitting [2], [3]. k nearest neighbor (KNN) classification was also advocated by Fix and Hodges [1] as a possible solution to this issue. Here, K objects are found in a training dataset that are closest to a test object/data. A label is then assigned according to the predominance of the majority class in this neighborhood. This is the standard prediction approach to KNN classification, known as the majority rule.

KNN classification has the remarkable property that, under very mild conditions, the error rate of a KNN algorithm tends towards being Bayes optimal as the sample size tends towards infinity [4]. For any data analysis application, if establishing a model with some training dataset is proving troublesome, it is likely that a KNN algorithm will provide the best solution [5]. As a result, KNN algorithms have been widely used in research and are considered to being one of the top-10 data mining algorithms [6]. In this era of big data, KNN approaches provide a particularly efficient way of identifying useful patterns and developing case-based reasoning algorithms for artificial intelligence (AI) [7], [8].

As with other classification algorithms, KNN classification is a two-phase procedure: model training and test data prediction. In the training phase, KNN only involves finding a suitable K for a given training dataset [9]. The most common method for this is the cross-validation. In the prediction phase, the first step is a search for K data points in the training dataset that are most relevant to a query (test data/sample). Without other information, the K most relevant data points are taken to be the K nearest neighbors of the test data within the training dataset. After this, a prediction is made on the basis of the class of test data that most frequently occurs amongst the K neighbors. This is referred to as the majority rule (which is similar to the Bayesian rule). From the above procedure of KNN classification, it indicates that there are mainly four challenging issues, K computation, nearest neighbour selection, nearest neighbour search, and classification rule.

It can be very difficult to set a suitable K for a given training dataset. Training samples often have different distributions in the sample space, which can lead to there being no obviously suitable K for the whole training sample space. This has resulted in two research directions as follows. One is to set different K values to different sample subspaces [10], [11]. Another is to set different K values to different test samples [12]. From Zhang *et al.*, the efficiency is pretty good when clustering the sample space to 3-5 subspaces [11]. It is time-consuming to set different K values to different test samples.

Nearest neighbor selection has been studied extensively. It is really a procedure of determining a proximity measure. Much of the related research has focused on constructing distance functions for measuring the proximity. Song, *et al.* proposed two KNN methods for measures the informativeness of test data, Locally Informative-KNN (LI-KNN) and Globally Informative-KNN (GI-KNN) [13]. Each of them is as a query-based distance metric to measure the closeness between objects. However, no distance function has yet been identified that is suitable for all training samples regardless of their distribution. In other words, there remains a need for distance function for the selection of the K nearest neighbor points that can work effectively across most training samples. On the other hand, feature selection is useful to choosing the K nearest neighbors [14]. This can be a lazy procedure because it depends on data mining tasks.

Nearest neighbor search is particularly challenging and remains unsolved because it is a complete sample space search when looking for all the K nearest neighbors for each test data. As a result, KNN classification is often referred to as a lazy data mining method. There are some efforts devoted to resolving the problem of how to undertake a truly effective nearest neighbor search. From the lately

• *The author is with the College of Computer Science and Technology, Central South University, Changsha 410083, PR China. E-mail: zhangsc@csu.edu.cn.*

Manuscript received 21 Aug. 2019; revised 21 Dec. 2020; accepted 31 Dec. 2020. Date of publication 5 Jan. 2021; date of current version 12 Sept. 2022. (Corresponding author: Shichao Zhang.) Recommended for acceptance by Y. Zhang. Digital Object Identifier no. 10.1109/TKDE.2021.3049250

related reports, most of them were focused on seeking K approximate nearest neighbors. Li, *et al.* examined 16 approximate nearest neighbor search algorithms in different domains [15]. And then, they proposed a nearest neighbor search method that achieves both high query efficiency and high recall empirically on majority of the datasets under a wide range of settings.

The majority rule has been used widely and successfully in real applications as a KNN classification principle. However, if a training dataset has unbalanced classes, the majority rule is unable to work effectively. This is why there is little research that has sought to extend the use of KNN algorithms to cost/risk-sensitive learning.

The rest of this paper is organized as follows: Section 2 briefly introduces the problem of K computation. The methods for nearest neighbor selection are reviewed in Section 3. Nearest neighbor search is the topic of Section 4. An overview of the issues surrounding classification rules is provided in Section 5. Section 6 evaluates the efficiency of some new classification rules that have been developed to improve KNN classification. Some conclusions are put forward in Section 7.

2 K Computation

Setting a suitable K for a given training dataset is a key step in KNN classification. There are two main ways in which this can be accomplished. The first option is for the data analyst employing KNN classification to assume that the users will provide the K for their datasets. However, it is clearly challenging for users to establish effective K values in this way.

The second option is to use all of the samples in the given training dataset, i.e., to attack the issue of K computation with training samples. There are three main approaches to dealing with K computation. We briefly outline them below.

Setting a Single K Value for the Whole Sample Space. Since its inception almost all approaches to KNN classification focused on setting only one suitable K for a given training dataset. A natural way of going about this was to use cross-validation to find a K for the training dataset. This method has been successfully applied in real applications in statistics and data mining [16]. One way of computing an optimal K for a dataset is to use what is known as holdout cross-validation. First of all, a suitable K for each sample in a dataset is searched for. Then, the K that delivers the highest classification efficiency is chosen for the whole sample space. Another technique is to use m-fold cross-validation. This first partitions the training dataset into m mutually-disjoint subsets. Then, cross-validation is used to generate a viable K for each subset. Finally, the K that delivers the best classification efficiency is chosen for the whole sample space.

After KNN method was invented by Fix and Hodges [1], a lot of research has been devoted to setting a single K for the whole sample space. For example, Loftsgaarden and Quesenberry referred the KNN method to a nonparametric solution that was applied to estimate the multivariate density function [17]. Sebestyen took the KNN method as an important tool of decision making and collected it to his monograph "Decision-making Processes in Pattern Recognition" [18]. Cover and Hart designed a KNN algorithm for pattern classification [2]. Wettschereck and Dietterich experimentally compared the nearest-neighbor and nearest-hyperrectangle algorithms [3]. Hastie and Tibshirani proposed a KNN method for classification and regression [19]. They designed a general model based on a local Linear Discriminant Analysis, called the DANN algorithm. Singh, *et al.* applied KNN method to image understanding [20]. They designed a nearest neighbor algorithm, aiming to find the nearest average distance rather than nearest maximum number of neighbors. Peng, *et al.* advocated to use other classification rules to KNN classification. A locally adaptive neighborhood morphing classification method was developed to minimize bias [21]. Chen and Shao applied the KNN method to estimate the value of missing data [22]. A jackknife variance estimation was designed for improving the nearest-neighbor imputation. Domeniconi and Gunopulos presented an adaptive KNN algorithm for pattern classification [23]. The maximum margin boundary found by the SVM is used to determine the most discriminant direction over the neighborhood of test data. Tao, *et al.* proposed a RKNN (reverse k nearest neighbor) method for retrieving dynamic multidimensional datasets [24]. It utilizes a conventional data-partitioning index on the dataset and does not require any pre-computation. Zhu and Basir designed a fuzzy KNN algorithm for remote sensing image classification [25]. In the algorithm, each nearest neighbor provides evidence on the belongingness of the input pattern to be classified, and it is evaluated based on a measure of disapproval to achieve the adaptive capability during the classification process. Zhang and Zhou applied KNN method to multilabel classification [26]. They designed an ML-KNN algorithm, or a multi-label lazy learning approach. Qin, *et al.* applied the KNN method to cost-sensitive classification [27]. The neighborhood in the minor class is set much more cost. Liu, Wu and Zhang designed a KNN algorithm for multilabel classification [28]. A nearest neighbor selection was designed for multilabel classification. The certainty factor is further adopted to well address the problem of unbalanced and uncertain data. Gallego, *et al.* developed a clustering-based KNN classification. The K nearest neighbors are taken as the initial points [29].

Gou *et al.* proposed a KNN algorithm based on local constraint representation [30]. Specifically, it first finds the K nearest neighbors of the test data in each class of the training data according to the euclidean metric. Then it constructs K local mean vectors in each class according to the K nearest neighbors in the previous step. Finally, it uses the local mean vector to fit the test data and combines local constraints to get the final representation-based distance metric. It predicts class labels of the test data through the representation-based distance metric. In addition, Gou *et al.* also proposed a KNN algorithm based on local mean representation [31]. It is different from the previous algorithmist. Although it is also the first to construct a local mean vector in each class of training data. However, in the following steps, when performing linear representation of the test data, it performs two linear representations to obtain the optimal relationship representation. Finally, it predicts the class label of the test data through a new metric function based on the local mean.

Setting Different K Values for Different Samples. It is well known that instances are nonuniformly distributed in a

sample space. It therefore seems reasonable that different test data should be given different K values. Thus, some recent work has proposed the approach of setting an optimal K for each test sample. For example, Guo *et al.* has presented an approach where a KNN model is constructed for a sample that replaces the sample itself, to serve as the basis of classification [32]. The value of K is automatically determined, can vary according to the sample and is optimal in terms of classification accuracy. Li *et al.* have proposed an improved KNN classification algorithm that uses different numbers of nearest neighbors for different categories, rather than having a fixed number across all categories [33]. More samples (nearest neighbors) are used to decide whether test data should be classified to a category, if there are more samples for that category in the training dataset. Yu *et al.* have put forward a method, called optimally pruned K-nearest neighbors (OP-kNNs), that can compete with other state-of-the-art methods while remaining fast [34]. Setting an optimal K for each sample has also been proposed in the context of graph sparse reconstruction (see [12]), called G-optimal-K. When compared with the preceding three optimal-K methods above in experiments, the G-optimal-K approach produced the best results. We will therefore briefly examine this approach in greater detail.

The idea of Zhang *et al.* was to revise conventional KNN algorithms using a sparse reconstruction framework [12]. This can generate different K values for different test samples and make the best use of prior knowledge in the training dataset. The G-optimal-K approach was designed with three regularization terms. A reconstruction process is adopted to move between training and test samples that obtains a K value for every test sample. In the reconstruction process, a least square loss function is applied to achieve the minimal reconstruction error. An L1-norm is then applied to generate element-wise sparsity for selecting the different K values for the different test samples. To improve the reconstruction performance, an $L_{2,1}$-norm is employed to generate row sparsity, thus removing noisy samples. Finally, an Locality Preserving Projection (LPP) regularization term is suggested to preserve the local sample structure.

K Value Approximation. The G-optimal-K approach has since been extended to approximation computation in an algorithm, called Ktree (see [35]). This is illustrated in Fig. 1. To get an optimal K for a test data, one must compute a K value for each new data item, one by one, before predicting the class of the test data. A major issue with K computation is that it is both expensive and time-consuming if users would like to set different K values to predict different data classifications. Therefore, Zhang *et al.* advocated approximating the optimal K of the test data with its nearest neighbor's optimal K by using what is called a Ktree [35]. A Ktree is built to rapidly search for the nearest neighbor and K value for the data. The K computation proceeds as follows. In the training phase, the KNN method is Usedom to build a Ktree for the training dataset, where each leaf node is a training sample with an optimal K value. In the prediction phase, the KNN method is used to search the Ktree to obtain the nearest neighbor for the test data. The optimal K of the nearest neighbor is assigned to the test data.

The K value approximation delivers three results as follows. One is that different K values can be set with training samples. Another is that the approximation K values work well compared with the real-time computed K. Last one is that the approximation K values can be trained before given a data mining task, whereas the real-time computed K is obtained after given the data mining task which is a lazy procedure.

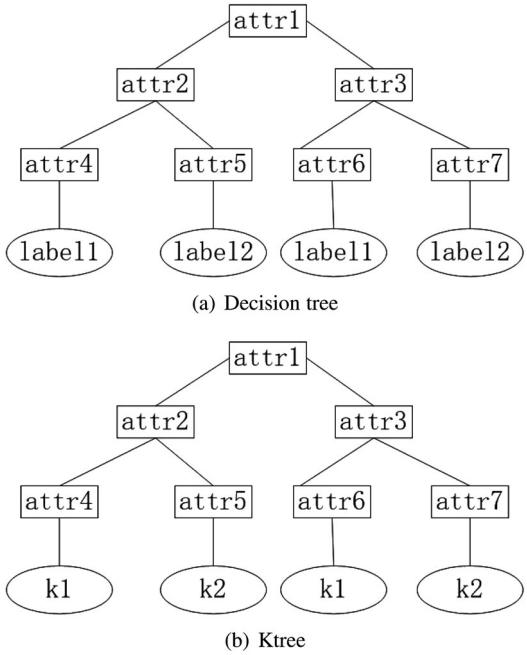

Fig. 1. Decision tree with class leaves and K value leaves.

3 Nearest Neighbor Selection

As having mentioned, nearest neighbor selection is really a procedure of determining a proximity measure. Most of the research relating to nearest neighbor selection involves studying the distance function or similarity metrics for measuring the proximity between KNN classification objects. Numerous techniques have been developed for modifying KNN classification in terms of distance measurement selection/construction and this has become a hot topic in KNN algorithm research [19], [21], [36], [37]. Currently a variety of distance measures are available, such as euclidean, Hamming, Minkowsky, Mahalanobis, Camberra, Chebychev, Quadratic, Correlation, Chi-square, and hyperrectangle [38], Value Difference Metrics [39] and Minimal Risk Metrics [40], with an additional option being grey distance [41].

However, distance functions generally do not perform consistently well, even under specified conditions [42]. This makes the use of a KNN approach highly experience dependent. Various attempts have been made to remedy this situation. Amongst these, Discriminant Adaptive Nearest Neighbor (DANN) is notable for carrying out a local linear discriminant analysis to deform the distance metric based on the 50 nearest neighbors [19]. Local Flexible Metric based on Support Vector Machine (LFM-SVM) also deforms the metric by feature weighting. Here, the weights are inferred by training an SVM on the entire dataset [23]. Klocal Hyperplane distance Nearest Neighbor (HkNN) uses a collection of 15-70 nearest neighbors from each class to span a linear subspace for that class. Classification is then based not on

the distance to prototypes but on the distance to linear subspaces [37].

There are other kinds of distance defined by the data properties. Examples here include: tangent distance using the USPS zip code dataset [43], shape context-based distance using the MNIST digit dataset [44], distances between histograms of texts using the CUReT dataset [45] and geometric blur-based distances using Caltech-101 [46]. These measures can be extended by kernel techniques so as to estimate a curved local neighborhood [47]. This makes the space around the samples nearer or further from the test data, depending on class-conditional probability distributions. There are also many other efforts to measuring the proximity between samples. For example, Blanzieri and Ricci presented a minimum risk metric (MRM) for classification tasks that exploits estimates of the posterior probabilities [40]. The MRM is optimal, in the sense that it optimizes the finite misclassification risk, whereas the Short and Fukunaga Metric minimize the difference between finite risk and asymptotic risk. Domeniconi, *et al.* built a locally adaptive nearest-neighbor classification method to try to minimize bias [36]. a chi-squared distance analysis was employed to compute a flexible metric for producing neighborhoods that are highly adaptive to query locations. Neighborhoods are elongated along less relevant feature dimensions and constricted along most influential ones. Peng, *et al.* advocated an adaptive KNN classification method for minimizing bias [47]. A quasiconformal transformed kernels was applied to compute neighborhoods over which the class probabilities tend to be more homogeneous. Athitsos, *et al.* applied the KNN method to information retrieval [48]. A method, BoostMap, was designed for efficient nearest neighbor retrieval under computationally expensive distance measures. Chen, *et al.* developed a KNN search by utilizing the distance lower bound to avoid the calculation of the distance itself if the lower bound is already larger than the global minimum distance [49]. They constructed a lower bound tree (LB-tree) by agglomeratively clustering all the sample points to be searched. Li, *et al.* proposed a KNN algorithm with local probability centers of each class [33]. It can reduce the number of negative contributing points which are the known samples falling on the wrong side of the ideal decision boundary, in a training set and by restricting their influence regions. Liu, Wu and Zhang presented a nearest neighbor selection was designed for multilabel classification [28]. The target labels of test data are predicted with the help of those relevant and reliable data, which explored by the concept of shelly nearest neighbor. Song, *et al.* proposed two KNN methods for measures the informativeness of test data [13]. That is, Locally Informative-KNN and Globally Informative-KNN were constructed as a query-based distance metric to measure the closeness between objects. Zhang, Cao and Wang developed a weighted heterogeneous distance Metric [50]. With the WHDM, the reduced random subspace-based Bagging (RRSB) algorithm is proposed for construct ensemble classifier, which can increase the diversity of component classifiers without damaging the accuracy of the component classifiers. Gou, *et al.* designed a generalized mean distance-based KNN classifier (GMDKNN) [51]. The multi-local mean vectors of a test data in each class are calculated by adopting its class-specific K nearest neighbors.

More recently, a new measure named neighborhood counting has been proposed that can define the similarity between two data points by using the number of neighborhoods [42]. To measure the similarity between two data points, all neighborhoods of covering both the two data points are counted and the number of such neighborhoods as a measure of similarity. As the features of high-dimensional data are often correlated the above kinds of measures can easily become meaningless. Some approaches have been designed to deal with this issue, e.g., by applying variable aggregation to define the measure [36], [52], [53]. Aside from the above kinds of measures, another strategy that can be applied is to consider the geometrical placement of the neighbors rather than their actual distances [54]. This approach is effective in some cases, but is in conflict with human intuition when the data is manifold. López *et al.* proposed a nearest neighbor classification for high-dimensional data [55]. Specifically, it first sorts all the features through the FR strategy. Then it selects the first r features with larger weights. Finally, a new distance function is constructed to predict the class label based on the selected features and the test data. Feng *et al.* proposed a new distance measurement function to solve the problem of class imbalance [56]. It first reconstructs the data with a projection matrix, and then calculates the KL divergence between different classes. Finally, it gets a distance metric matrix by solving the proposed objective function. In order to solve the problem of outliers, Mehta *et al.* proposed a KNN classification through harmonic mean distance [57]. It looks for K nearest centroid neighbors of the test data in each class. In addition, it also calculates a local centroid mean vector in each class, and uses the nearest harmonic mean distance between the test data and the local centroid mean to predict the class label of the test data. Syaliman *et al.* proposed a KNN algorithm based on local mean and distance weight [58]. It not only finds the local mean vector in each class, but also applies weights based on the distance. The class weights farther from the test data are smaller, on the contrary, the class weights closer to the test data are larger. Jiao *et al.* proposed a paired distance metric for KNN [59]. It avoids the traditional method using only one distance metric for global data. It also sets weights on features and has resolved the uncertainty in the output of the classifier. Nguyen *et al.* proposed a large-margin distance metric learning approach [60]. It can maximize the margin of each training example. It also solves the optimization problem that the proposed formula is non-convex. Weinberger *et al.* proposed a distance measurement function for KNN [61]. It can make K nearest neighbors always belong to the same class and maximize the distance between different classes. Goldberger *et al.* proposed a new mahalanobis distance measure, it can learn the low-dimensional linear embedding of a data [62]. In this way, it can reduce the computational complexity of the algorithm and speed up the KNN classification. Mensink *et al.* proposed two distance-based classifiers (i.e., KNN and nearest class mean (NCM)) [63]. In addition, it also introduces a new distance measurement function to improve their performance. In the experiment, it is verified that the NCM algorithm has better performance. V. Davis*et al.* used information theory to learn a mahalanobis distance function [64]. It minimizes the KL divergence between two Gaussian

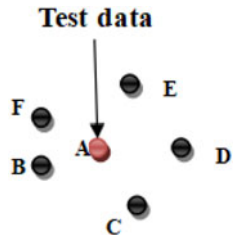

Fig. 2. Ideal nearest neighbors of test data A.

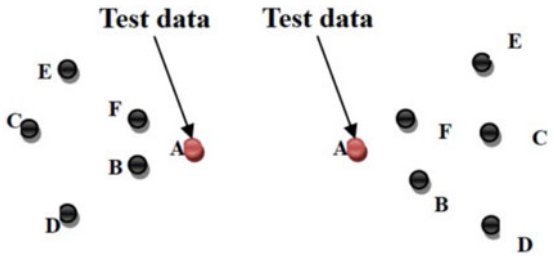

Fig. 3. Nearest neighbors to the left or right of test data A.

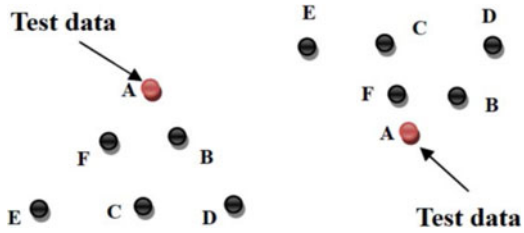

Fig. 4. Nearest neighbors below or above test data A.

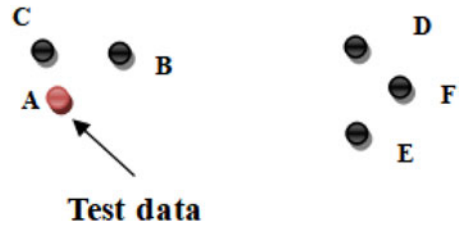

Fig. 5. Some nearest neighbors further away from test data A.

distributions by constraining the distance function. Finally, it learns a mahalanobis matrix A by optimizing the objective function. Nguyen *et al.* proposed a new metric learning method through maximization of the Jeffrey divergence [65]. Specifically, it first generates two multivariate Gaussian distributions from local pairwise constraints. Then it maximizes the Jeffrey divergence of these two distributions to get a linear transformation. Finally, it turns the problem into an unconstrained optimization problem and solves it. Globerson *et al.* proposed a new metric learning for classification tasks [66]. It can make the points of the same class be close to each other and the points of different classes are far away. It optimizes the equivalent convex dual form of the proposed function to solve the metric matrix. FISHER *et al.* discussed the application of multiple measures to the same principles in classification problems [67]. Wang *et al.* proposed a feature extraction algorithm [68]. Its core idea is to draw the data of same class in its neighbors, and push data of different classes away from it as much as possible. It avoids the problem of small sample size in traditional Linear Discriminant Analysis (LDA). In addition, it has also been extended to nonlinear feature extraction through a kernel function.

Lately a very different approach to nearest neighbor selection has been adopted, called the shell-KNN algorithm, as mentioned in Section 2 [16], [69]. It first selects the K nearest neighbors using distance functions. Then, the shell nearest neighbors are chosen from the K nearest neighbors. This has therefore been described as quadratic-selection.

Generally, there is an expectation that all the selected nearest neighbors for a test data will be ideally distributed around the test data, as illustrated in Fig. 2.

Fig. 2 shows an ideal nearest neighbor distribution for test data A. However, the collection of training samples for real applications is effectively random and the training samples often have different distributions. Consequently, the nearest neighbors of a test data will not have an ideal distribution. Other potential cases are shown in Figs. 3, 4, and 5.

According to experiments conducted by Zhang [16], [69], it is much more efficient to take nearest neighbors closely distributed around test data A. It takes quadratic-selection to identify the nearest neighbors within the shell surrounding test data A. First of all, one searches for the K nearest neighbors of test data A in the training dataset. Each feature/attribute is then treated as an axis and the left and right nearest neighbors are selected from the K nearest neighbors for test data A. Finally, the nearest neighbors within the shell of test data A are selected from the left and right nearest neighbors on all axes.

Quadratic-selection can only identify those nearest neighbors that are distributed around test data A. It is therefore worth taking note of the following cases.

Point 1 Some nearest neighbors amongst the K nearest neighbors will be selected many times;

Point 2 The number of selected shell-nearest neighbors of an item of test data, S, is less or equal to K, i.e., $S \leq K$;

Point 3 Different test data will produce different S values.

In the case of point 1 above, the number of times a nearest neighbor is selected can be used to compute the weight of the nearest neighbor. This is a new way of setting weights for samples. We will discuss how to make use of this case in detail in Section 5.

In relation to point 2, the set of selected shell nearest neighbors of an item of test data is a subset of the K nearest neighbors. Note that, when the nearest neighbors of the test data are further away, as shown in Fig. 5, the quadratic-selection approach may fail to locate them.

For point 3, it delivers a fact that different test data can be set different K values. In the shell-KNN algorithm, all nearest neighbors are expected to distribute around the test data. Some points therefore need to be discarded from the selected K nearest neighbors. In other words, the number of shell-nearest neighbors should be less or equal to K for each item of test data.

4 Nearest Neighbor Search

It is often suggested in the literature that KNN classification is a lazy form of learning. It does not need to train any model to fit the given training samples, beyond setting the K values. This means that, for each item of test data, KNN classification

has to search the whole training sample space to obtain the K nearest neighbors. This is a time-consuming procedure that weakens the range of possible applications for KNN algorithms. Therefore, Samet designed an algorithm for finding K nearest neighbors of a test data [70]. It adopted a pruning technique with the maxnearestdist as distance upper bound.

For supporting the increasing functionalities of smartphones, it is important to fast locate one of the nearest objectives for a customer. Therefore, there are recently some reports on searching K approximate nearest neighbors. For example, Jegou, Douze and Schmid proposed a product quantization for nearest neighbor search [71]. The idea is to decompose the space into a Cartesian product of low-dimensional subspaces and to quantize each subspace separately. And then, many improved models have been developed for spatiotemporal data. Li and Hu applied the product quantization to develop an approximate nearest neighbor search algorithm for high order data [15]. They incorporated the high order structures of data into the process of designing a more effective subspace decomposition way. Pan, *et al.* designed a Product quantization with dual codebooks for approximate nearest neighbor search [72]. It uses dual codebooks simultaneously to reduce the quantization error in each subspace. Also, a grouping strategy was presented to group the database vectors by their encoding modes, and thus the extra memory cost caused by dual codebooks can be reduced. Chiu, *et al.* presented a ranking model with learning neighborhood relationships embedded in the index space [73]. In this model, the nearest neighbor probabilities are estimated by employing neural networks to characterize the neighborhood relationships, i.e., the density function of nearest neighbors with respect to the query. Lu, *et al.* advocated an approximate nearest neighbor search via virtual hypersphere partitioning [74]. The idea is to impose a virtual hypersphere, centered at the query, in the original feature space and only examine points inside the hypersphere. Munoz, *et al.* developed a large scale approximate nearest neighbor search for high dimensional data [75]. They used a nearest neighbor graph created over large collections. This graph is created based on the fusion of multiple hierarchical clustering results, where a minimum-spanning-tree structure is used to connect all elements in a cluster. Malheiros and Walter constructed a data-partitioning error-controlled strategy for approximate nearest neighbor searching [76]. By changing the size of the candidate neighborhood, the precision and performance are balanced when searching for neighbors. Tellez, *et al.* thought, for intrinsically high-dimensional data, the only possible solution is to compromise and use approximate or probabilistic approaches [77]. And then, they proposed a singleton indexes for nearest neighbor search. Ozan, *et al.* designed a vector quantization method for approximate nearest neighbor search which enables faster and more accurate retrieval on publicly available datasets [78]. The vector quantization is defined as a multiple affine subspace learning problem and the quantization centroids are explored on multiple affine subspaces. Dasgupta and Sinha theoretically studied three Randomized Partition Trees for Nearest Neighbor Search [79]. And then, they combined classical k-d tree partitioning with randomization and overlapping cells.

To compare these approximate nearest neighbor search algorithms, Li, *et al.* examined 16 representative algorithms selected from different domains [80]. And then, they proposed a nearest neighbor search method that achieves both high query efficiency and high recall empirically on majority of the datasets under a wide range of settings.

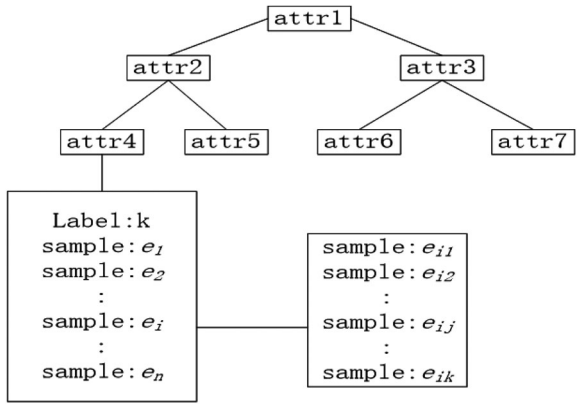

Fig. 6. A K*Tree.

In view of the fact that KNN classification is an approximate solution, effort has recently been undertaken to overcome the lazy aspects of KNN classification with an improved approach to approximation that can achieve good prediction efficiency and accuracy when compared to standard KNN classification algorithms. This is known as K*Tree [35]. K*Tree is not a classifier. It is just a particular kind of tree that has a range of useful information in its leaf nodes, to facilitate obtaining the K nearest neighbors for test data quickly. An example of a K*Tree is provided in Fig. 6.

K*Tree is an extension of the Ktree illustrated in Fig. 1b, with samples added to each leaf node. These samples include the K nearest neighbors of the data in a leaf node and the K nearest neighbors of the nearest neighbor of the data in the leaf node. In other words, in a K*Tree, each of the samples in its leaf nodes can be expressed as $(k; e_1, < e_{11}, e_{12}, \ldots, e_{1k} >; e_2, < e_{21}, e_{22}, \ldots, e_{2k} >; \ldots ; e_i, < e_{i1}, e_{i2}, \ldots, e_{ik} >; \ldots; e_n < e_{n1}, e_{n2}, \ldots, e_{nk} >)$, where, k is an integer value, $< e_1, e_2, \ \ldots, e_i, \ \ldots, e_n >$ is a vector of the training samples that have the same k value, and $< e_{i1}, e_{i2}, \ \ldots, e_{ik} >$ is a vector of the k nearest samples of e_i. However, there are many samples with the same K value, so, information has to be added to the leaf nodes of a K*Tree sample by sample.

K*Tree can be used in the following way. For an item of test data T, the K*Tree can first be searched to obtain the nearest sample, e_i, of T in a leaf node. The K value in this leaf node can then be taken as the K value of T. Finally, the K nearest samples of T can be searched for just amongst the samples attached to the nearest sample e_i, as follows:

$$\begin{aligned} & e_i, < e_{i1}, e_{12}, \ldots, e_{ik} > \\ & < e^1_{i1}, e^2_{i2}, \ldots, e^{k_{i1}}_{i1} > \\ & < e^1_{i2}, e^2_{i2}, \ldots, e^{k_{i2}}_{i2} > \\ & \quad \vdots \\ & < e^1_{ik}, e^2_{ik}, \ldots, e^{k_{ik}}_{ik} >. \end{aligned} \tag{1}$$

This means that KNN classification using the K*Tree approach does not need to search the whole training sample space, significantly enhancing its performance.

Clearly, K*Tree is only an approximation of finding the K nearest neighbors of an item of test data. There are some particular things to note about it:

- P1 The test data's nearest neighbor will definitely be included in the set of K nearest neighbors for the test data. Other K-1 nearest neighbors will be close enough to the test data.
- P2 The prediction efficiency of KNN classification using K*Tree is almost the same as that of KNN classification using Ktree.
- P3 KNN classification with K*Tree is much more robust than KNN classification with Ktree and standard KNN classification.

From the above, it can be seen that K*Tree classification approaches are very different from traditional KNN classification methods. KNN classification with K*Tree is not a lazy learning approach because the K*Tree has to be trained before predicting the test data. Its advantage is that there is no need to search the whole sample space. However, there are still two main limitations in K*Tree classification as follows.

1) K*Tree classification provides only an approximate solution to a query. It is not clear how to estimate the confidence of the answer?
2) It is not clear which tree is the best structure to store the K and the nearest neighbors for the K*Tree classification?

5 Classification Rules

Once the K nearest neighbors of an item of test data have been selected from training samples, the class label of the test data needs to be predicted, using a classification rule/principle. In general, the most popularly-used rules for KNN classification are the majority rule and its various forms of weighting. Recently, two classification rules have been proposed against datasets with imbalanced classes. Apart from recalling these classification rules, some suggestions are advocated to improve the classification rules of shell nearest neighbor classification in this section.

5.1 The Majority Classification Rule

This is a simple yet efficient approach to classification that predicts the class of the test data according to the class of the majority of the K nearest neighbors.

For a training dataset, D, with n features and a decision attribute, let $D = \{(Xi,Yi)\}, Xi = (X_{i1}, X_{i2}, \ldots, X_{in})$, $Dom(Y) = \{c1, c2, \ldots, cm\}$ be the domain of the decision attribute, Y. One can obtain K nearest neighbors of a query (test data), $T = (X, Y)$, from the training dataset. $KNN(T)$ is the set of these K nearest neighbors. The majority rule for KNN classification is as follows:

$$Y = \arg\max_{c \in Dom(Y)} \sum_{Xi \in KNN(T)} I(Yi = c), \tag{2}$$

where $I(\bullet)$ is an indicator function.

5.2 The Weighting Classification Rule

In the majority rule, the K nearest neighbors are implicitly assumed to have equal weight for any decision, regardless of their distance from the test data. This rule therefore adheres to the notion that it is conceptually preferable to give different weights to the K nearest neighbors according to their distance from the test data, with closer neighbors having greater weight. The distance weighting-based classification rule is as follows:

$$Y = \arg\max_{c \in Dom(Y)} \sum_{X_i \in KNN(T)} I(Yi = c) \frac{1}{d(Xi, X)}, \tag{3}$$

where $d(Xi, X)$ is the distance between (Xi, Yi) and the test data T. $d_{MAX} = MAX_{X_i \in KNN(T)}\{d(Xi, X)\}$. The Eq. (2) can be changed to Eq. (3) as follows:

$$Y = \arg\max_{c \in Dom(Y)} \sum_{X_i \in KNN(T)} I(Yi = c)\left(1 - \frac{d(Xi, X)}{d_{MAX}}\right). \tag{4}$$

A general weighting classification rule is as follows:

$$Y = \arg\max_{c \in Dom(Y)} \sum_{X_i \in KNN(T)} \mathrm{w}i * I(Yi = c). \tag{5}$$

A kernel function classification rule is as follows:

$$Y = \sum_{X_i \in KNN(T)} YiK(Xi, X). \tag{6}$$

where $K(Xi, X)$ is a kernel function. Eq. (5) is constructed for a numerical value. If the decision attribute is of a certain character type, Yi can be replaced with the indictor function.

5.3 CF (Certainty Factor) Classification Rule

There are numerous variations upon the above classification rules. However, neither of these classification rules work well for the imbalanced datasets typical of real applications, such as tumor diagnosis in medical tests or investments in the stock market. Generally, this is a challenge when the data mining tasks are sensitive to cost or risk.

To deal with this issue, [81], [82] has proposed ways of increasing the competitiveness of minor classes when undertaking imbalanced data classification, known as Certainty Factor KNN (CF-KNN) classification. These are summarized below.

The CF measure is incorporated into the KNN classification as follows. For a test sample $T = (X, Y)$, assume $p(Y = ci|D)$ is the ratio of ci in the training dataset, D, and $p(Y = ci|KNN(T))$ is the ratio of ci in the set of K nearest neighbors, $KNN(T)$. If $p(Y = ci|KNN(T)) \geq p(Y = ci|D)$, the CF is computed as follows:

$$\begin{aligned} &CF(Y = ci, KNN(T)) \\ &= \frac{p(Y = ci|KNN(T)) - p(Y = ci|D)}{1 - p(Y = ci|D)}. \end{aligned} \tag{7}$$

It means $CF(Y = ci, KNN(T)) > 0$. If $p(Y = ci|KNN(T)) < p(Y = ci|D)$, the CF is computed as follows:

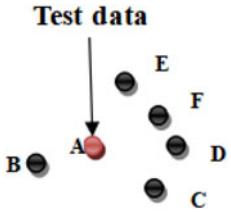

Fig. 7. The shell nearest neighbors are not ideally distributed around test data A.

$$CF(Y = ci, KNN(T)) = \frac{p(Y = ci|KNN(T)) - p(Y = ci|D)}{p(Y = ci|D)}. \quad (8)$$

It means $CF(Y = ci, KNN(T)) < 0$.

The CF strategy for KNN classification can be defined as follows:

$$Y = \arg\max_{1 \le i \le m} CF(Y = ci, KNN(T)). \quad (9)$$

According to the description of CF, $CF(Y = ci, KNN(T))$ will have a value in the range $[-1, 1]$. If $CF(Y = ci, KNN(T)) > 0$, it will increase that the class of the query should be predicted to be $Y = ci$. If $CF(Y = ci, KNN(T)) < 0$, however, it will decrease that the class of the query should be predicted to be $Y = ci$. If $CF(Y = ci, KNN(T)) = 0$, it will be the same as it is for the training set D that the class of the query should be predicted to be. In other words, the class of T is undetermined for binary classification applications.

5.4 Lift Classification Rule

The above CF-KNN classification can be modified by using the lift measure, also known as Lift-KNN classification as follows:

$$Lift(Y = ci, KNN(T)) = \frac{p(C = ci|KNN(T))}{p(C = ci|D)}. \quad (10)$$

Clearly, $Lift(Y = ci, KNN(T)) > 0$. If the lift of a class is less than or equal to 1, the probability of the class is not increased for the K nearest neighbors. However, if $Lift(Y = ci, KNN(T)) > 1$, the probability of the class is increased for the K nearest neighbors. In that case, the Lift-KNN strategy for KNN classification can be defined as follows:

$$Y = \arg\max_{1 \le i \le m} Lift(Y = ci, KNN(T)). \quad (11)$$

If $Lift(Y = ci, KNN(T)) = 1$, the class of T is undetermined for binary classification applications.

5.5 Shell KNN Classification Rule

Recalling Fig. 2 in Section 3, shell nearest neighbors are well distributed around the test data A. In real applications, the shell nearest neighbors may not be ideal, due to the fact that training examples are randomly collected. This case is illustrated in Fig. 7 as follows.

With the quadratic-selection rule, sample B is selected many more times than samples C, D, E and F, although all five nearest neighbors are very close to test data A. This is a very interesting case when using KNN classification in data mining applications. In this paper, we formally discuss this case as follows.

Using the quadratic-selection rule, we can obtain 2n samples, left($X1$), right($X1$), ... , left(Xn), right(Xn) from the set $KNN(T)$, where left(Xi), right(Xi) are the left and right nearest neighbors of the test data, T, with respect to the feature Xi, respectively. The majority rule of KNN classification can be extended as follows:

$$Y = \arg\max_{c \in Dom(Y)} \sum_{Xi \in KNN(T)} (I(left(Xi) = c) + I(right(Xi) = c)). \quad (12)$$

From Eq. (12), if a nearest neighbor is selected many times, it will be the winner. This is another way of addressing imbalanced classes with Case 1 (mentioned in Section 3). For Fig. 7, let the label of B be *c1*, the label of C be *c2*, and the label of D, E and F be *c3*, and n =10. From Eq. (2), the label of T is assigned to *c3*. From Eq. (12), the label of T is voted to *c1*. If we let the label of B be *c1*, the label of C, D, E and F be *c2*. From Eq. (2), the label of T is predicted to be *c2*. In the case of Eq. (12), the label of T is not determined, due to the fact that *c1* and *c2* receive the same votes. To attack this issue when mining imbalanced data, we can modify Eq. (13) as follows:

$$Y = \arg\max_{c \in Dom(Y)} \{count(Xi, Yi)^l I(Yi = c)\}, \quad (13)$$

where the $count(Xi, Yi)$ is the number of selected nearest neighbors (Xi, Yi), and "l" is greater than 1.

TABLE 1
Information About the Downloaded Datasets

Datasets	Number of samples	Number of samples after deleted some	Dimensions	Classes
OCCUDS	1,994	1,194	101	10
Chess	3,196	1,860	36	2
CNAE	1,080	696	856	9
German	1,000	762	20	2
Ionosphere	351	250	34	2
Isolet	1,560	936	617	2
Letter	20,000	11,984	16	26
Segment	2,130	1,518	19	7
USPS	9,298	5,758	256	10
Vehicle	846	512	18	4
Waveform	2,746	1,267	21	3
Yeast	1,484	945	1,470	10
Arcene	200	110	10,000	2
Carcinom	174	99	9,182	11
CLLSUB	111	61	11,340	3

6 Experiments

A large number of new ideas have been presented in this paper. For the sake of simplicity, just a few representatives of the new classification rules in Section 5 will be evaluated in this section. The classification accuracy of the new classification rules was compared with the majority classification rule (referred to here as the standard KNN classification rule) across 15 datasets, as shown in Table 1, below.

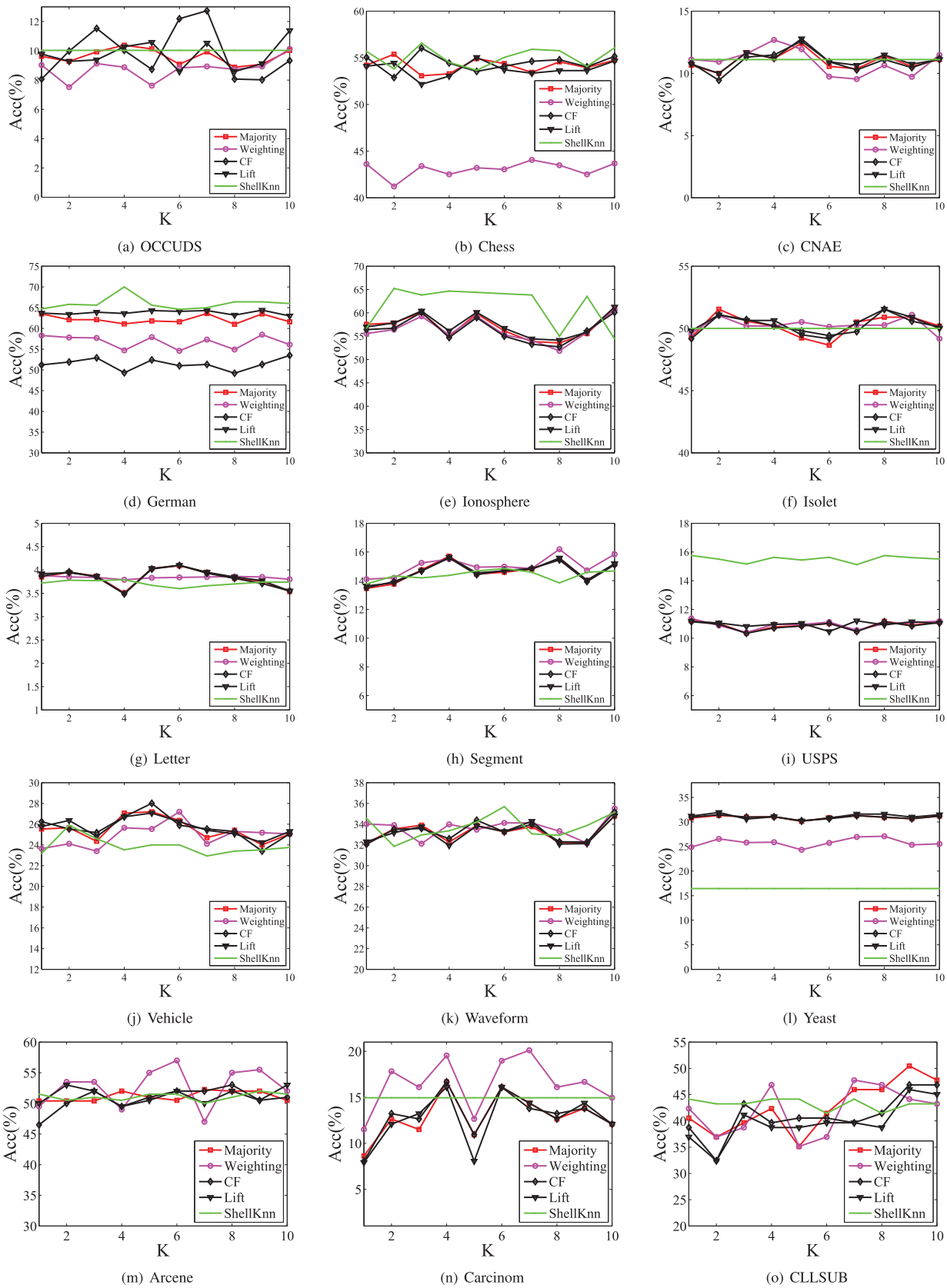

Fig. 8. Classification accuracy for the original datasets using different K values.

6.1 Experimental Setting

The 15 datasets above were downloaded from the UCI machine learning library. They include 5 binary datasets and 10 multi-class datasets. The 15 datasets needed to be slightly modified to meet the requirements of the evaluation. So, 80 percent of the samples belonging to a certain class in the binary datasets were deleted, with all of the remaining samples forming the new datasets. In the multi-class datasets, if the

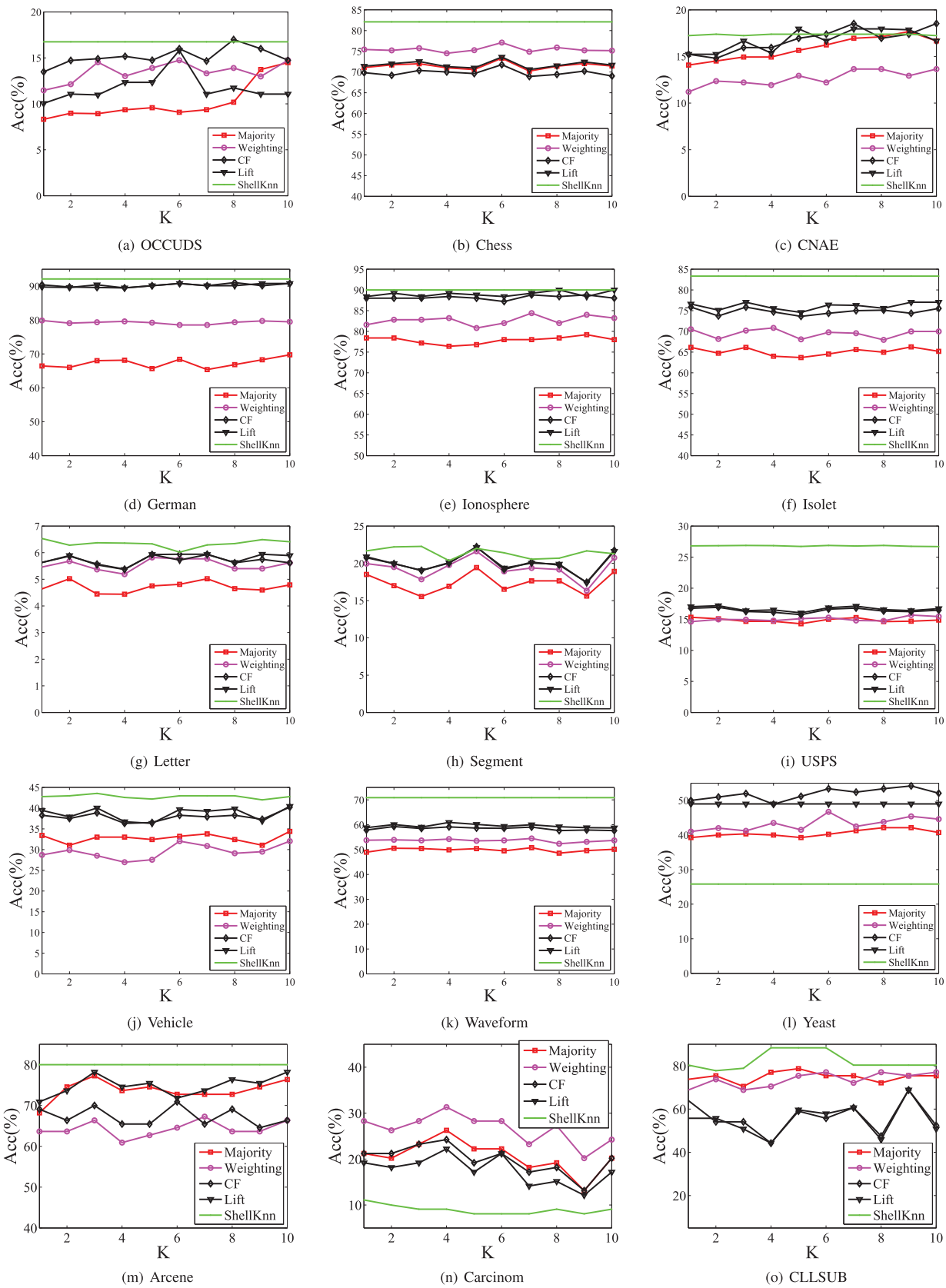

Fig. 9. Classification accuracy for the unbalanced datasets with different K values.

number of classifications was odd, we removed 80 percent of the samples belonging to the odd-numbered classes (i.e., 1, 3, 5...). If the number of classifications was even, we removed 80 percent of the samples belonging to the even class (i.e., 2, 4, 6...).

A set of experiments was conducted on the above datasets using the classification rules mentioned in Section 5. The primary goal was to compare their performance with that of the standard classification rule, but a further objective was to test

TABLE 2
Binary Results on Imbalanced Data

Datasets	Chess			German			Ionosphere			Isolet			Arcene		
	ACC	SEN	SPE	ACC	SEN	SPE	ACC	SEN	SPE	ACC	SEN	SPE	ACC	SEN	SPE
Majority	0.690	0.261	0.772	0.662	0.684	0.297	0.836	0.904	0.040	0.681	0.731	0.305	0.750	0.905	0.132
Weighting	0.729	0.174	0.853	0.897	0.972	0.027	0.875	0.971	0.012	0.745	0.858	0.179	0.741	0.881	0.181
CF	0.720	0.174	0.840	0.916	0.995	0.003	0.891	0.989	0.040	0.771	0.899	0.130	0.671	0.767	0.290
Lift	0.727	0.144	0.842	0.918	0.993	0.003	0.900	0.987	0.012	0.778	0.905	0.141	0.671	0.767	0.290
ShellKNN	0.821	0	1	0.921	1	0	0.900	1	0	0.833	1	0	0.800	1	0

TABLE 3
Average Classification Accuracy for Multiple Classifications in the Unbalanced Datasets

Datasets	CCUDS	CNAE	Letter	Segment	USPS	Vehicle	Waveform	Yeast	Carcinom	CLLSUB
Majority	13.02	13.65	4.71	19.43	15.32	31.05	49.86	48.99	20.61	74.92
Weighting	14.41	14.51	5.38	20.03	16.71	36.33	58.39	42.12	26.57	73.61
CF	15.13	18.68	5.70	20.01	17.16	38.65	54.46	48.99	19.90	56.07
Lift	15.69	18.53	5.82	21.61	16.92	39.45	60.93	49.00	17.58	54.92
ShellKNN	16.75	17.26	6.54	21.74	26.80	42.77	70.86	25.82	10.07	80.33

TABLE 4
Standard Deviation of the Classification Accuracy

Datasets	CCUDS	CNAE	Letter	Segment	USPS	Vehicle	Waveform	Yeast	Carcinom	CLLSUB
Majority	0.01	0.12	0.01	0.01	0.15	0.01	0.01	0	0.03	0.02
Weighting	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.03
CF	0.01	0.16	0.01	0.01	0.01	0.02	0.01	0	0.03	0.07
Lift	0.01	0.16	0.01	0.01	0.01	0.01	0.01	0	0.02	0.07
ShellKNN	0	0.04	0.01	0	0	0	0	0	0.01	0.01

their effect on the classification imbalances in the data. Each dataset was first divided into a test set and a training set using 10-fold cross-validation. Then, all the classification rules were examined using the original dataset (no sample deletion, i.e., unclassified and unbalanced data). This was to ensure the experiment was using different K values. After this, all the classification rules were tested using different K values on the unbalanced datasets. Finally, K = 5 was selected to perform 10 experiments on the unbalanced datasets, to examine the average and variance of the classification accuracy.

It should be noted that, for the binary dataset, we not only obtained the classification accuracy (ACC), but also the sensitivity (SEN) and specificity (SPE) for the unbalanced dataset.

6.2 KNN Classification of the Original Datasets

The first set of experiments were conducted using the downloaded datasets without any changes. The results are shown in Fig. 8, which presents the classification accuracy of the classification rules for the 15 original datasets with different K values. It can be seen that the accuracy of these classification rules does not differ greatly for most of the datasets, especially in relation to the OCCUDS, CNAE, Isolet, Letter, Segments, Vehicle and Waveform datasets. The ShellKnn classification rule performs the worst on the Yeast dataset, but the best on the German, Ionosphere and USPS datasets. Overall, the ShellKnn classification rule performs pretty well. For other classification rules, there was little difference in their performance on the original datasets, though the weighted classification rules performed particularly poorly on some datasets, such as Chess and Yeast.

6.3 Unbalanced Datasets With Binary Classes

The second set of experiments was conducted with the modified datasets where there were unbalanced binary classes. The results are presented in Fig. 9.

Fig. 9 shows the classification accuracy for all of the classification rules for a class-unbalanced dataset, with Chess, German, Ionosphere and Isolet being the binary datasets. For the German, Ionosphere and Isolet datasets, the performance of the Majority classification rules and Weighted classification rules was not very good because these two classification rules do not consider the importance of small sample classes when dealing with class imbalances. The CF, Lift and ShellKNN classification rules slightly increased the competitiveness of the small classes in the unbalanced classifications. Thus, in most cases, their performance was much better than the Majority and Weighted classification rules.

Table 2 shows the ACC, SEN, and SPE results for the binary dataset. Here, it can be seen that the ShellKnn classification rule performed the best for the binary-class datasets, with the Majority classification rule being the worst.

6.4 Unbalanced Multi-Class Datasets

The third set of experiments were conducted using the modified datasets with multi-class imbalances. The results for the average classification accuracy and variance across the 10 experiments are presented in Tables 3 and 4.

With regard to the variance, all of the classification rules were stable, the variance was small and the results were robust. For the average classification accuracy, the ShellKNN. Lift, and CF classification rules improved the average classification accuracy by 2.74, 0.89, and 0.32 percent, respectively, in relation to the Majority classification rule. In particular, t on For the USPS and Waveform datasets, the ShellKNN classification rule improved the classification accuracy by 11.48 and 21 percent, respectively.

Across the various unbalanced datasets, the ShellKnn classification rule generally performed well in comparison to the other classification rules.

7 Conclusion

This paper has systemically reviewed the latest research regarding KNN classification that is addressed to its four most challenging issues. This paper has focused on introducing the main results recently established within our research group. These can be distinguished from other extant approaches by their interest in delivering new research directions for KNN classification.

Although the approaches presented here are both efficient and promising, there are still some open issues that require further research:

1) How to set different K values for different kinds of test data so that the results delivered by the KNN classification algorithm will offer the best possible performance whilst remaining robust.
2) How to establish the best tree structures for building KTree and K*Tree when a decision tree is not the best data structure for saving or rapidly searching for the K values and the nearest neighbors of a leaf node.
3) How to make KNN classification efficient when mining big data.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grants 61836016 and 61672177.

References

[1] E. Fix and J. L. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: Small sample performance," California Univ. Berkeley, Tech. Rep., 1952.
[2] T. M. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.
[3] D. Wettschereck and T. G. Dietterich, "An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms," *Mach. Learn.*, vol. 19, no. 1, pp. 5–27, 1995.
[4] Z. Lei, Y. Jiang, P. Zhao, and J. Wang, "News event tracking using an improved hybrid of kNN and SVM," in *Proc. Int. Conf. Future Gener. Commun. Netw.*, 2009, pp. 431–438.
[5] "30 questions to test a data scientist on k-nearest neighbors (kNN) algorithm," 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/30-questions-test-k-nearest-neighbors-algorithm/
[6] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.
[7] X. Zhu, S. Zhang, W. He, R. Hu, C. Lei, and P. Zhu, "One-step multi-view spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 10, pp. 2022–2034, Oct. 2019.
[8] X. Zhu, S. Zhang, Y. Li, J. Zhang, L. Yang, and Y. Fang, "Low-rank sparse subspace for spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1532–1543, Aug. 2019.
[9] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, "A novel kNN algorithm with data-driven k parameter computation," *Pattern Recognit. Lett.*, vol. 109, pp. 44–54, 2018.
[10] M. Tan, S. Zhang, and L. Wu, "Mutual kNN based spectral clustering," *Neural Comput. Appl.*, vol. 32, no. 11, pp. 6435–6442, 2020.
[11] S. Zhang, J. Zhang, X. Zhu, Y. Qin, and C. Zhang, "Missing value imputation based on data clustering," in *Transactions on Computational Science I*. Berlin, Germany: Springer, 2008, pp. 128–138.
[12] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for kNN classification," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 3, 2017, Art. no. 43.
[13] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, "IKNN: Informative K-nearest neighbor pattern classification," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discov.*, 2007, pp. 248–264.
[14] X. Zhu, S. Zhang, R. Hu, Y. Zhu, and J. Song, "Local and global structure preservation for robust unsupervised spectral feature selection," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 3, pp. 517–529, Mar. 2018.
[15] L. Li and Q. Hu, "Optimized high order product quantization for approximate nearest neighbors search," *Front. Comput. Sci.*, vol. 14, no. 2, pp. 259–272, 2020.
[16] S. Zhang, "Parimputation: From imputation and null-imputation to partially imputation," *IEEE Intell. Informat. Bull.*, vol. 9, no. 1, pp. 32–38, Nov. 2008.
[17] D. O. Loftsgaarden *et al.*, "A nonparametric estimate of a multivariate density function," *The Ann. Math. Statist.*, vol. 36, no. 3, pp. 1049–1051, 1965.
[18] G. S. Sebestyen, *Decision-Making Processes in Pattern Recognition*. Indianapolis, IN, USA: Macmillan Publishing Co. 1962.
[19] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification and regression," in *Proc. 8th Int. Conf. Neural Inf. Process. Syst.*, 1996, pp. 409–415.
[20] S. Singh, J. Haddon, and M. Markou, "Nearest neighbour strategies for image understanding," in *Proc. Workshop Adv. Concepts Intell. Vis. Syst.*, 1999, pp. 2–7.
[21] J. Peng, D. R. Heisterkamp, and H. Dai, "LDA/SVM driven nearest neighbor classification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2001, vol. 1, pp. I–I.
[22] J. Chen and J. Shao, "Jackknife variance estimation for nearest-neighbor imputation," *J. Amer. Statist. Assoc.*, vol. 96, no. 453, pp. 260–269, 2001.
[23] C. Domeniconi and D. Gunopulos, "Adaptive nearest neighbor classification using support vector machines," in *Proc. 14th Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 665–672.
[24] Y. Tao, D. Papadias, and X. Lian, "Reverse kNN search in arbitrary dimensionality," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, pp. 744–755.
[25] H. Zhu and O. Basir, "An adaptive fuzzy evidential nearest neighbor formulation for classifying remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 8, pp. 1874–1889, Aug. 2005.
[26] Z. Minling and Z. Zhihua, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, 2007.
[27] Z. Qin, A. T. Wang, C. Zhang, and S. Zhang, "Cost-sensitive classification with k-nearest neighbors," in *Proc. Int. Conf. Knowl. Sci. Eng. Manage.*, 2013, pp. 112–131.
[28] H. Liu, X. Wu, and S. Zhang, "Neighbor selection for multilabel classification," *Neurocomputing*, vol. 182, pp. 187–196, 2016.
[29] A.-J. Gallego, J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan, "Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation," *Pattern Recognit.*, vol. 74, pp. 531–543, 2018.
[30] J. Gou, W. Qiu, Z. Yi, X. Shen, Y. Zhan, and W. Ou, "Locality constrained representation-based k-nearest neighbor classification," *Knowl.-Based Syst.*, vol. 167, pp. 38–52, 2019.
[31] J. Gou, W. Qiu, Z. Yi, Y. Xu, Q. Mao, and Y. Zhan, "A local mean representation-based k-nearest neighbor classifier," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 3, pp. 1–25, 2019.
[32] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *Proc. OTM Confederated Int. Conf. "On Move Meaningful Internet Syst."*, 2003, pp. 986–996.

[33] B. Li, Y. W. Chen, and Y. Q. Chen, "The nearest neighbor algorithm of local probability centers," *IEEE Trans. Syst., Man, Cybern., B Cybern.*, vol. 38, no. 1, pp. 141–154, Feb. 2008.
[34] Q. Yu, Y. Miche, A. Sorjamaa, A. Guillen, A. Lendasse, and E. Séverin, "OP-KNN: Method and applications," *Advances Artif. Neural Syst.*, vol. 2010, 2010, Art. no. 1.
[35] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1774–1785, May 2018.
[36] C. Domeniconi, J. Peng, and D. Gunopulos, "Locally adaptive metric nearest-neighbor classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1281–1285, Sep. 2002.
[37] P. Vincent and Y. Bengio, "K-local hyperplane and convex distance nearest neighbor algorithms," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 985–992.
[38] S. Salzberg, "A nearest hyperrectangle learning method," *Machi. Learn.*, vol. 6, no. 3, pp. 251–276, 1991.
[39] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Commun. ACM*, vol. 29, no. 12, pp. 1213–1228, 1986.
[40] E. Blanzieri and F. Ricci, "Probability based metrics for nearest neighbor classification and case-based reasoning," in *Proc. Int. Conf. Case-Based Reasoning*, 1999, pp. 14–28.
[41] S. Zhang, "Nearest neighbor selection for iteratively kNN imputation," *J. Syst. Softw.*, vol. 85, no. 11, pp. 2541–2552, 2012.
[42] H. Wang, "Nearest neighbors by neighborhood counting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 942–953, Jun. 2006.
[43] P. Simard, Y. LeCun, and J. S. Denker, "Efficient pattern recognition using a new transformation distance," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1993, pp. 50–58.
[44] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
[45] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *Int. J. Comput. Vis.*, vol. 62, no. 1/2, pp. 61–81, 2005.
[46] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, vol. 1, pp. 26–33.
[47] J. Peng, D. R. Heisterkamp, and H. Dai, "Adaptive quasiconformal kernel nearest neighbor classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 656–661, May 2004.
[48] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios, "BoostMap: An embedding method for efficient nearest neighbor retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 89–104, Jan. 2008.
[49] Y.-S. Chen, Y.-P. Hung, T.-F. Yen, and C.-S. Fuh, "Fast and versatile algorithm for nearest neighbor search based on a lower bound tree," *Pattern Recognit.*, vol. 40, no. 2, pp. 360–375, 2007.
[50] Y. Zhang, G. Cao, B. Wang, and X. Li, "A novel ensemble method for k-nearest neighbor," *Pattern Recognit.*, vol. 85, pp. 13–25, 2019.
[51] J. Gou, H. Ma, W. Ou, S. Zeng, Y. Rao, and H. Yang, "A generalized mean distance-based k-nearest neighbor classifier," *Expert Syst. Appl.*, vol. 115, pp. 356–372, 2019.
[52] T. Mary-Huard and S. Robin, "Tailored aggregation for classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2098–2105, Nov. 2009.
[53] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, no. 1/3, pp. 37–52, 1987.
[54] J. S. Sánchez, F. Pla, and F. J. Ferri, "On the use of neighbourhood-based non-parametric classifiers," *Pattern Recognit. Lett.*, vol. 18, no. 11/13, pp. 1179–1186, 1997.
[55] J. López and S. Maldonado, "Redefining nearest neighbor classification in high-dimensional settings," *Pattern Recognit. Lett.*, vol. 110, pp. 36–43, 2018.
[56] L. Feng, H. Wang, B. Jin, H. Li, M. Xue, and L. Wang, "Learning a distance metric by balancing KL-divergence for imbalanced datasets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 12, pp. 2384–2395, Dec. 2019.
[57] S. Mehta, X. Shen, J. Gou, and D. Niu, "A new nearest centroid neighbor classifier based on k local means using harmonic mean distance," *Inf.*, vol. 9, no. 9, 2018, Art. no. 234.
[58] K. Syaliman, E. Nababan, and O. Sitompul, "Improving the accuracy of k-nearest neighbor using local mean based and distance weight," *J. Phys., Conf. Series*, vol. 978, no. 1, 2018, Art. no. 012047.
[59] L. Jiao, X. Geng, and Q. Pan, "BP k nn: k-nearest neighbor classifier with pairwise distance metrics and belief function theory," *IEEE Access*, vol. 7, pp. 48 935–48 947, 2019.
[60] B. Nguyen and B. De Baets, "An approach to supervised distance metric learning based on difference of convex functions programming," *Pattern Recognit.*, vol. 81, pp. 562–574, 2018.
[61] K. Q. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. 18th Int. Conf. Neural Inf. Process. Syst.*, 2005, vol. 18, pp. 1473–1480.
[62] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Proc. 17th Int. Conf. Neural Inf. Process. Syst.*, 2004, vol. 17, pp. 513–520.
[63] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2624–2637, Nov. 2013.
[64] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 209–216.
[65] B. Nguyen, C. Morell, and B. De Baets, "Supervised distance metric learning through maximization of the jeffrey divergence," *Pattern Recognit.*, vol. 64, pp. 215–225, 2017.
[66] A. Globerson and S. Roweis, "Metric learning by collapsing classes," in *Proc. 18th Int. Conf. Neural Inf. Process. Syst.*, 2005, vol. 18, pp. 451–458.
[67] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[68] F. Wang and C. Zhang, "Feature extraction by maximizing the average neighborhood margin," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[69] S. Zhang, "Shell-neighbor method and its application in missing data imputation," *Appl. Intell.*, vol. 35, no. 1, pp. 123–133, 2011.
[70] H. Samet, "K-nearest neighbor finding using maxnearestdist," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 243–252, Feb. 2008.
[71] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.
[72] Z. Pan, L. Wang, Y. Wang, and Y. Liu, "Product quantization with dual codebooks for approximate nearest neighbor search," *Neurocomputing*, vol. 401, pp. 59–68, 2020.
[73] C.-Y. Chiu, A. Prayoonwong, and Y.-C. Liao, "Learning to index for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 1942–1956, Aug. 2020.
[74] K. Lu, H. Wang, W. Wang, and M. Kudo, "VHP: Approximate nearest neighbor search via virtual hypersphere partitioning," *Proc. VLDB Endowment*, vol. 13, no. 9, pp. 1443–1455, 2020.
[75] J. V. Munoz, M. A. Gonçalves, Z. Dias, and R. D. S. Torres, "Hierarchical clustering-based graphs for large scale approximate nearest neighbor search," *Pattern Recognit.*, vol. 96, 2019, Art. no. 106970.
[76] M. de Gomensoro Malheiros and M. Walter, "Spatial sorting: An efficient strategy for approximate nearest neighbor searching," *Comput. Graph.*, vol. 57, pp. 112–126, 2016.
[77] E. S. Tellez, G. Ruiz, and E. Chavez, "Singleton indexes for nearest neighbor search," *Inf. Syst.*, vol. 60, pp. 50–68, 2016.
[78] E. C. Ozan, S. Kiranyaz, and M. Gabbouj, "K-subspaces quantization for approximate nearest neighbor search," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1722–1733, Jul. 2016.
[79] S. Dasgupta and K. Sinha, "Randomized partition trees for exact nearest neighbor search," in *Proc. 26th Annu. Conf. Learn. Theory*, 2013, pp. 317–337.
[80] W. Li *et al.*, "Approximate nearest neighbor search on high dimensional data-experiments, analyses, and improvement," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1475–1488, Aug. 2020.
[81] S. Zhang, "KNN-CF approach: Incorporating certainty factor to kNN classification," *IEEE Intell. Informat. Bull.*, vol. 11, no. 1, pp. 24–33, Dec. 2010.
[82] S. Zhang, "Cost-sensitive kNN classification," *Neurocomputing*, vol. 391, pp. 234–242, 2020.

Shichao Zhang (Senior Member, IEEE) received the PhD degree from Deakin University, Australia. He is currently a china national distinguished professor with the Central South University, China. His research interests include data mining and big data. He has published 90 international journal papers and more than 70 international conference papers. He is a CI for 18 competitive national grants. He is a member of the ACM; and serves/served as an associate editor for four journals.