Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7409171, 11 pages
https://doi.org/10.1155/2022/7409171

Research Article

A Large-Scale k-Nearest Neighbor Classification Algorithm Based on Neighbor Relationship Preservation

Yunsheng Song, Xiaohan Kong, and Chao Zhang

College of Information Science and Engineering, Shandong Agricultural University, Tai'an, 271018, China

Correspondence should be addressed to Yunsheng Song; sys_sd@126.com

Received 19 August 2021; Accepted 8 December 2021; Published 7 January 2022

Academic Editor: Petros Nicopolitidis

Owing to the absence of hypotheses of the underlying distributions of the data and the strong generation ability, the k-nearest neighbor (kNN) classification algorithm is widely used to face recognition, text classification, emotional analysis, and other fields. However, kNN needs to compute the similarity between the unlabeled instance and all the training instances during the prediction process; it is difficult to deal with large-scale data. To overcome this difficulty, an increasing number of acceleration algorithms based on data partition are proposed. However, they lack theoretical analysis about the effect of data partition on classification performance. This paper has made a theoretical analysis of the effect using empirical risk minimization and proposed a large-scale k-nearest neighbor classification algorithm based on neighbor relationship preservation. The process of searching the nearest neighbors is converted to a constrained optimization problem. Then, it gives the estimation of the difference on the objective function value under the optimal solution with data partition and without data partition. According to the obtained estimation, minimizing the similarity of the instances in the different divided subsets can largely reduce the effect of data partition. The minibatch k-means clustering algorithm is chosen to perform data partition for its effectiveness and efficiency. Finally, the nearest neighbors of the test instance are continuously searched from the set generated by successively merging the candidate subsets until they do not change anymore, where the candidate subsets are selected based on the similarity between the test instance and cluster centers. Experiment results on public datasets show that the proposed algorithm can largely keep the same nearest neighbors and no significant difference in classification accuracy as the original kNN classification algorithm and better results than two state-of-the-art algorithms.

1. Introduction

K-nearest neighbor classification algorithm is a lazy learning method that does not require a training process but simply stores training instances [1]. When given a test instance, kNN classification algorithm first calculates the similarity between the given instance and all instances in the training set, then finds k-nearest instances according to the similarity, finally predicts its label by the majority voting based on the category of these instances. Owing to its advantages of substantial theoretical foundation, strong generalization performance, and no assumptions on data distribution, the kNN classification algorithm has been widely used in many fields [2–6]. It is selected as one of the top 10 classic algorithms in data mining [7].

With the rapid development of sensing and Internet technology, data from all walks of life is increasing by orders of magnitude; big data becomes the focus of government, academia, and industry; and the research results of data analysis and mining have been widely used in the Internet of Things, healthcare, e-commerce, finance, and so on. However, kNN needs to compute the similarity between the aim instances and all the training instances so that its execution efficiency faces a great challenge in the big data environment. An increasing number of acceleration algorithms are proposed to improve the efficiency of kNN classification algorithms to process the large-scale data [8–10]. The existing accelerating algorithms for kNN classification can usually be divided into two categories from the perspective of data preprocessing: kNN classification based on data

partition (DP-kNN) algorithm and kNN classification based on instance selection (IS-kNN) algorithm [11, 12].

The basic ideology DP-kNN algorithm divides the training set into several subsets by feature space partition, then classifies the test instances using some of the divided subsets. Specifically, the feature space of the training set is divided into several subregions, then determines which divided subregions the test instance belongs to, and finally finds k-nearest neighbors in the subset of instances corresponding to that region. These algorithms mainly take advantage of the local learning characteristics of kNN classification algorithms: the label of the test instance in the prediction process is only related to the most similar instances in the training set. Therefore, it tries to ensure that k-nearest neighbors of each instance in its divided subset are consistent with the ones in the original dataset. However, most of the existing data partition algorithms scarcely analyze this consistency from a theoretical point of view, so they are difficult to guarantee that the algorithm has high generalization performance.

Different from the DP-kNN algorithm, the IS-kNN algorithm does not use all the training examples. At the same time, it finds k-nearest neighbors of the test instance from a representative subset of the training set, where the subset is obtained by using the instance selection algorithm. Because the size of the representative subset is smaller than the original training set, it can greatly improve the efficiency of finding neighbors for the test instance. Instance selection is an important data preprocessing method; it removes noisy instances and those instances far away from the classification decision plane from the training set according to the similarity and label differences of the training instances. Since there are more instances far from the classification decision plane in most datasets than those close to the classification decision plane, an instance selection algorithm can greatly reduce the size of the training set and keep the classification accuracy relatively unchanged. However, the time complexity of most existing instance selection algorithms is the square of the training set size, which makes it difficult to effectively process large-scale data. Furthermore, it only uses the information of the part data rather than all the data, so its generalization performance could be negatively affected.

For the problem of the lack of consistency analysis about nearest neighbors under data partition, this paper analyzes its classification performance theoretically from the perspective of optimization. The contribution of this paper is as follows:

(1) Theoretically analyzing the effect of data partition on the classification performance of the kNN classification algorithm and giving the difference measurement between k-nearest neighbors obtained with data partition and without data partition

(2) Obtaining the fact that minimizing the similarity of the instances in different divided subsets can largely reduce the effect of data partition on the classification based on the theoretical analysis

(3) Adopting the minibatch k-Means clustering algorithm to execute data partition, because it divides the dataset into several subsets with a large difference in similarity

(4) Searching k nearest neighbors from the union of several candidate divided subsets for the test instance, where the candidate divided subsets are selected by the similarity between the test instance and cluster centers

(5) Compared with the two existing typical algorithms, the experimental results on the public dataset show that the proposed algorithm could largely hold k same nearest neighbors and similar classification accuracy of the original kNN classification algorithm

The rest of this paper is organized as follows. Section 2 reviews related methods about kNN classification acceleration algorithm. Section 3 analyzes the effect of data partition on the classification performance of the kNN classification algorithm and proposes a novel algorithm, called the large-scale kNN classification algorithm based on neighbor relationship preservation (NPR-kNN algorithm). Section 4 reports the experimental results through the comparison with existing methods. Section 5 gives the conclusion of this paper and the future work.

2. Related Work

The existing acceleration algorithm for k-nearest neighbor classification from the perspective of data preprocessing can be categorized into the acceleration algorithms based on data partition (DP-kNN algorithm) and the acceleration algorithms based on instance selection (IS-kNN algorithm).

The DP-kNN algorithm is mainly divided into three steps: the feature space of the current training instance is firstly divided into several subregions; then, the divided region where the test instances stay is determined, and finally, k-nearest neighbors are found from the subset of instances within this region. Because the kNN classification algorithm is a local learning algorithm, it is necessary to ensure that the neighboring sequences of the instance before and after data partition are consistent when dividing the training set. Most of the existing data partitioning algorithms for kNN classification algorithms are based on the binary tree structure; the current data is recursively divided into two subsets of similar capacity until the termination condition is met starting from the original set. Friedman et al. [13] firstly have proposed the concept of the KD tree, which uses the attributes of the data to recursively divide the k-dimensional feature space into several subregions and treat the data falling in each region as a subset. However, in the face of high-dimensional complex data, there will be a phenomenon that some attributes with a large amount of information are not used in the process of building a tree. To solve this problem, Verma et al. [14] have proposed a KD tree that maximizes variance (MKD-tree) algorithm, which selects the attribute with the largest variance of the attribute value on the current data as the node for division. The MKD-tree algorithm uses only a certain attribute each time the current data is divided, which will cause partial

information loss. For this reason, a binary tree algorithm based on principal component analysis is proposed [15], which divides the current data based on the score of the first m principal components and the corresponding median value. In addition, there also exist some data partition algorithms based on the structure of the nearest neighbor graph and hash approximation [16–19]. However, most of the existing data partitioning algorithms do not theoretically study the effect of data partitioning on the kNN classification algorithm.

The IS-kNN algorithm mainly searches k-nearest neighbors of the test instance in a representative subset of the training set with a relatively small size. The representative subset is obtained by various instance selection algorithms [12]. Hart [20] has proposed a compressed nearest neighbor based on 1NN (CNN) algorithm, which obtains a subset S of the training set T so that the instances in the set $T - S$ are correctly classified by S. That is, the instances in $T - S$ have the same labels as their neighbors in S. The CNN algorithm first randomly selects an instance from the training set into the set S. Then each time select an instance from $T - S$ and determine whether it is the same as the label of its neighbor in the set S: if it is consistent, put it in S, and repeat the above process until the set $T - S$ is empty. Although the CNN algorithm can obtain a relatively small subset S, this algorithm is very sensitive to the order of reading data, and the time complexity is the square of the number of instances in the training set. To overcome this difficulty, Angiulli [21] has proposed a fast compressed nearest neighbor (FCNN) algorithm. The FCNN algorithm first selects the instances closest to each center. It puts them into the set S and then iteratively selects representative instances from the set of instances in $T - S$ that are not correctly classified by the set S and puts them into the set S and repeats the selection process until S can correctly classify all instances in $T - S$. The FCNN algorithm is independent of the order of reading data, but its time complexity is $O(|T||S|)$, where T is the size of the set T. To improve the efficiency of the FCNN algorithm in processing large-scale data, [22] has proposed an FCNN algorithm based on parallel distributed computing. Although the proposed algorithm can achieve the purpose of greatly reducing the data size, it does not consider the impact of noise instances. To solve this problem, many editing algorithms have been proposed; its main idea is to remove instances that are inconsistent with their nearest neighbor labels. CNN series of algorithms and editing algorithms have achieved the goal of greatly reducing the size of the training set while the training error remains relatively unchanged. However, none of these algorithms consider the local sparsity of training examples in the feature space; it takes a negative effect on the classification performance of the kNN algorithm. For this reason, Nikolaidis et al. [23] have proposed a kind of boundary preservation algorithm, which first uses an editing algorithm to remove the noise instances in the training set and then uses the geometric characteristics of the potential distribution of the training instances in the feature space to divide the training set into border instances and interior instances. Finally, the representative instances from these two kinds of instances are selected and merged into the final instance selection subset. Furthermore, there are lots of improved kNN classification algorithms based on graphics and search algorithms [24]. However, most kNN classification algorithms based on instance selection need to calculate the similarity between all instances, which makes it difficult to process large-scale data [12].

3. Main Content

3.1. Related Concepts. Let $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ be the labeled training set of instances from l different classes, where each instance x_i is expressed by m-dimensional feature vector $(x_{i1}, x_{i2}, \cdots, x_{im})$ and x_{ij} is its jth feature value, y_i is the label of the instance x_i, m, N are the number of features and instances, $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, m$.

KNN classification algorithm is learned by comparing the similarity between the unlabeled instance and all the training instances. When given a test instance x, kNN first calculates the similarity d_i with each instance x_i in the training set T and then sorts all the training instances according to the order of the similarity d_i and takes the first k instance as k nearest neighbors of x_i. Finally, the class of the k neighbors with the largest number of instances will be determined to be the label of the instance x.

The basic idea of the existing kNN classification acceleration algorithm based on data partition is quite similar. After dividing the training set into several subsets of approximately equal size, it determines which of the divided subsets the test instance is most similar to and finds its k neighbors in this subset. The kNN algorithm is also a local learning algorithm, and the predicted label of the test instance is only related to the label of its nearest neighbors in the training set. To obtain a similar classification performance with the one using all the training instances, the DP-kNN algorithm needs to guarantee the k-nearest neighbors in the divided subset of the test instance to be consistent with the original training set as much as possible. Specifically, it is ensured that the test instance and its k-nearest neighbors in the training set are still in the same divided subset. Therefore, the data partition should be carefully studied.

The test instance has randomness and is unknown before executing prediction, and its location in the feature space is difficult to be decided. This difficulty takes the trouble to perform a good data partition. Fortunately, empirical risk minimization takes an effective way to solve this problem in statistical learning theory. Minimizing the empirical risk $1/N\sum_{i=1}^{N} l(y_i, \widehat{y}_i)$ can obtain the optimal solution, where $l(y_i, \widehat{y}_i)$ is the loss function between the true label y_i and the predicted label $\widehat{y}_i$ predicted by k-nearest neighbors of the instance x_i. In this way, it should ensure that each training instance and its k-nearest neighbors in the training set are still in the same divided subset. To this end, we analyze the effect of data partitioning on the neighbor relationship from the perspective of optimization.

3.2. Transformation of the Problem. For a given training set T, kNN classification algorithm finds k nearest neighbors for each instance in T transformed into solving the following

optimization problem:

$$\max_{A\in R^{N\times N}} tr(A\times D)$$
$$\text{s.t.}\sum_{j=1}^{N} A_{ij}=k, A_{ij}\in\{0,1\}, i,j=1,2,\cdots,N, \tag{1}$$

where $A=[A_{ij}]_{N\times N}\in R^{N\times N}$ is a boolean matrix, $A_{ij}=1$ when instance x_i is one of k nearest neighbors of the instance x_j, and $A_{ij}=0$ when instance x_i is not one of k nearest neighbors of the instance x_j; D is the similarity matrix and each element $D_{ij}>0$ is the similarity between the instance x_i and the instance x_j; and $tr(A\times D)$ is the trace of the matrix $A\times D$ which is the product of the matrix A and the matrix D, $i,j=1,2,3,\cdots,N$. The optimization problem (1) has only one optimal solution under the assumption that there exist different similarities for different instances. Let A^* be the optimal solution to the optimization problem (1).

Suppose the training set T is divided into n disjoint subsets T_l, where $l=1,2,3\cdots,n$. The KNN classification algorithm based on data partition is aimed at finding k-nearest neighbors of each instance within its divided subset. For each divided instance subset T_l, each element $(x_i,y_i)\in T_l$ searching its k-nearest neighbors in T_l can be transformed into solving the following optimization problem:

$$\max_{A^l\in R^{n_l\times n_l}} tr\left(A^lD^l\right)$$
$$\text{s.t.}\sum_{j=1}^{n_l} A_{ij}^l=k, A_{ij}^l\in\{0,1\}, i,j=1,2,\cdots,n_l, \tag{2}$$

where $A^l\in R^{n_l\times n_l}$, A^l is a boolean matrix, $A_{ij}^l=1$ if and only if the instance x_i is one of k-nearest neighbors of the instance $x_j\in T_l$, otherwise $A_{ij}^l=0$; the matrix $D^l\in R^{n_l\times n_l}$ is a submatrix of D with row and column indexes V_l, $V_l=\{i\in N^*:(x_i,y_i)\in T_l\}$, and $n_l=|T_l|$ is the size of the set T_l. Let $\bar{A}^l$ be the optimal solution for solving the optimization problem (2) for the instance subset T_l, where $l=1,2,\cdots,n$.

3.3. The Estimation of the Effect of Data Partition. Let $h(x_i)\in\{1,2,\cdots,l\}$ be the index of the divided subset which the instance x_i belongs to, e.g., $h(x_i)=j$ when $(x_i,y_i)\in T_j$, where $i=1,2,3,\cdots,N$, $l=1,2,3,\cdots,n$. In fact, the nearest neighbor algorithm based on data partitioning approximately decomposes the optimization problem (1) into n suboptimization problems (2) and independently solves this separate suboptimization problem. Combine the optimal solutions of these n subproblems (2) into a new matrix $\bar{A}=[\bar{A}_{ij}]_{N\times N}\in R^{N\times N}$, each element is $\bar{A}_{ij}=I_{(h(x_i),h(x_j))}\bar{A}_{ij}^l$, where binary function $I_{(a,b)}=1$ if and only if $a=b$; otherwise, $I_{(a,b)}=0$. The matrix $\bar{A}$ is an approximation of the optimal solution matrix A^* of the optimization problem (1). In order to ensure the performance of the algorithm, the difference between $f(A^*)$ and $f(\bar{A})$ should be minimized. In order to measure the difference between the two, we introduce the following lemmas and theorems.

Lemma 1. *$\bar{A}$ is the optimal solution to the following problem:*

$$\max_{A\in R^{n\times n}} tr\left(A\bar{D}\right)$$
$$s.t.\sum_{j=1}^{N} A_{ij}=k, A_{ij}\in\{0,1\}, i=1,2,\cdots,N, \tag{3}$$

where the matrix $\bar{D}=[\bar{D}_{ij}]_{N\times N}\in R^{N\times N}$, $\bar{D}_{ij}=I_{(h(x_i),h(x_j))}D_{ij}$.

Proof. According to the definition $\bar{D}$, each element satisfies the following rule:

$$\bar{D}_{ij}=\begin{cases} D_{ij}, & h(x_i)=h(x_j),\\ 0, & h(x_i)\neq h(x_j).\end{cases} \tag{4}$$

Moreover, the result can be seen from the calculation properties of the block matrix

$$tr\left(A\bar{D}\right)=\sum_{l=1}^{n} tr\left(A^lD^l\right). \tag{5}$$

Therefore, the original optimization problem (3) can be decomposed into n suboptimization problems (2), i.e.,

$$\max_{A\in R^{n\times n}} tr\left(A\bar{D}\right)=\sum_{l=1}^{n}\max_{A^l\in R^{n_l\times n_l}} tr\left(A^lD^l\right). \tag{6}$$

On the other hand, all suboptimization problems are independent and the matrix $\bar{A}^l$ is the optimal solution of the suboptimization problem (2), so the matrix $\bar{A}$ is the optimal solution of the problem $\max_{A\in R^{n\times n}} tr(A\bar{D})$. □

Theorem 2. *For the given training set $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$ and its partition index set $\{h(x_1),h(x_2),\cdots,h(x_N)\}$, then*

$$f(A^*)-f\left(\bar{A}\right)\leq\sum_{i=1}^{N}\sum_{j\notin\wedge^i} D_{ji}, \tag{7}$$

where $f(A)=tr(AD)$, $\wedge^i=\{l\in N^*:h(x_l)=h(x_i),1\leq l\leq N\}$.

Proof. Let $\bar{f}(A) = tr(A\bar{D})$. According to the definition of the matrix $\bar{A}$ and A^*, we have

$$\begin{aligned}\bar{f}(\bar{A}) &= \sum_{i=1}^{N}\sum_{j\in\wedge^i}\bar{A}_{ij}D_{ji} = \left(\sum_{i=1}^{N}\sum_{j\in\wedge^i}\bar{A}_{ij}D_{ji} + \sum_{i=1}^{N}\sum_{j\notin\wedge^i}\bar{A}_{ij}D_{ji}\right) \\ &\quad - \sum_{i=1}^{N}\sum_{j\notin\wedge^i}\bar{A}_{ij}D_{ji} = \sum_{i=1}^{N}\sum_{j=1}^{N}\bar{A}_{ij}D_{ji} - \sum_{i=1}^{N}\sum_{j\notin\wedge^i}\bar{A}_{ij}D_{ji} \\ &= f(\bar{A}) - \sum_{i=1}^{N}\sum_{j\notin\wedge^i}\bar{A}_{ij}D_{ji},\end{aligned} \tag{8}$$

$$\begin{aligned}\bar{f}(A^*) &= \sum_{i=1}^{N}\sum_{j\in\wedge^i}{A^*}_{ij}D_{ji} = \left(\sum_{i=1}^{N}\sum_{j\in\wedge^i}{A^*}_{ij}D_{ji} + \sum_{i=1}^{N}\sum_{j\notin\wedge^i}{A^*}_{ij}D_{ji}\right) \\ &\quad - \sum_{i=1}^{N}\sum_{j\notin\wedge^i}{A^*}_{ij}D_{ji} = \sum_{i=1}^{N}\sum_{j=1}^{N}{A^*}_{ij}D_{ji} - \sum_{i=1}^{N}\sum_{j\notin\wedge^i}{A^*}_{ij}D_{ji} \\ &= f(A^*) - \sum_{i=1}^{N}\sum_{j\notin\wedge^i}{A^*}_{ij}D_{ji}.\end{aligned} \tag{9}$$

Combining equations (8) and (9) and the result $\bar{f}(\bar{A}) \geq \bar{f}(A^*)$ according to Lemma 1, we have

$$\begin{aligned}f(A^*) &= \bar{f}(A^*) + \sum_{i=1}^{N}\sum_{j\notin\wedge^i}{A^*}_{ij}D_{ji} \leq \bar{f}(\bar{A}) + \sum_{i=1}^{N}\sum_{j\notin\wedge^i}{A^*}_{ij}D_{ji} \\ &= f(\bar{A}) + \sum_{i=1}^{N}\sum_{j\notin\wedge^i}{A^*}_{ij}D_{ji} - \sum_{i=1}^{N}\sum_{j\notin\wedge^i}\bar{A}_{ij}D_{ji} \\ &= f(\bar{A}) + \sum_{i=1}^{N}\sum_{j\notin\wedge^i}\left({A^*}_{ij} - \bar{A}_{ij}\right)D_{ji} \leq f(\bar{A}) + \sum_{i=1}^{N}\sum_{j\notin\wedge^i}D_{ji}.\end{aligned} \tag{10}$$

□

It is often assumed that each training instance and other instances in the training set have different similarity values in the kNN classification task. This assumption ensures that each training instance has k fixed nearest neighbors without considering the order of reading data, and the optimal problem (1) has a unique solution. Combined with the above theorem, reducing the difference between the objective function $f(A^*)$ and $f(\bar{A})$ can help reduce the difference between the approximate solution $\bar{A}$ and the optimal solution A^*. Therefore, we need to minimize the estimated difference $\sum_{i=1}^{N}\sum_{j\notin\wedge^i}D_{ji}$, i.e., the similarity between instances that are not in the same partitioned subset should be promoted to decrease as far as possible.

To achieve this aim, the minibatch k-Means clustering (MKC) algorithm is adopted to perform data partition for efficiently and effectively dealing with large data [25–27]. The MKC algorithm is one kind of the two-step k-Means clustering algorithm; it first performs k-Means algorithms on the randomly sampled instances from the original data to obtain the cluster center; then, the rest of the instances decide which cluster they belong to according to the similarity to the cluster centers. Meanwhile, the MKC algorithm is efficient because its time complexity is $O(Nm)$, where m is the size of the sampled subset. An additional advantage of this algorithm is that the maximum number of clusters often does not exceed $\sqrt{N}$ and the size of divided subsets has the uniformity effect, which provides us with an important reference basis for determining the number of divided subsets [28].

3.4. NPR-kNN Algorithm. Suppose the training set is divided into n disjoint clusters $T_1, T_2, \cdots, T_n$ using the MKC algorithm, and each cluster is to be a subset after division. It is an important step to decide k nearest neighbors of the given test instance after the training set partition. For the given test instance x, the traditional way decides which divided subsets the instance x belongs to according to the similarity between the instance x and the cluster centers, then finds k nearest neighbors within this aim subset. However, this way could not be effective for those instances which are far from the aim cluster center because it is difficult to guarantee that these instances and their neighbors in the training set are still in the same cluster. These instances and their k nearest neighbors are very likely in several adjacent clusters because they have higher similarity in the small local region of the feature space than other instances. Therefore, the method of cluster fusion is used to solve this problem. The aim cluster where the test instance x search its k nearest neighbors extends to be the union of λ candidate clusters, where the cluster centers of these candidate clusters are the first λ most similar with the instance x among all the clusters, and λ is an integer greater than 1. In this way, it largely increases the possibility that the test instance x can find the same k nearest neighbors as the original training set. The fixed value of the parameter λ for different datasets is not desirable for the large difference in the sparseness of data distribution. We adopt the early stopping rule to adaptively determine the value of λ for different test instances. It successively merges λ candidate clusters from $\lambda = 1$ to n until k nearest neighbors in the merged set of the test instance x does not change. The following algorithm shows the detailed procedure of the NPR-kNN algorithm.

3.5. Complexity Analysis of the Proposed Algorithm. Besides the classification performance, execution efficiency is another important evaluation. The NPR-kNN algorithm includes data partition stage and prediction stage. The minibatch k-Means cluster algorithm is adopted to perform data partition in the first stage, and it is designed for dealing with big data and gets several times more efficient than the traditional k-Means cluster algorithm [29]. The test instance searches its k-nearest neighbors using the aim divided subset rather than all the training instances in the prediction stage. The aim divided subset is obtained by only computing the similarity between the cluster centers and the test instance. Moreover, the NPR-kNN algorithm has the additional advantage of allowing distributed storage of large-scale training data. The

Input: The training set $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$, the test instance x, number of subsets n.
Output: The predicted label $\hat{y}$.
1 Initialization: $\Delta = \{0\}$, k-nearest neighbor set $NN_k = \varnothing$
2 Divide the set T into n disjoint $T_1, T_2, \cdots, T_n$ using Mini-batch k-Means clustering algorithm and get n cluster centers set $C = \{c_1, c_2, \cdots, c_n\}$;
while$(\Delta \neq \varnothing \text{or } C \neq \varnothing)$**do**
3 Find the set T_v according to the similarity between x and each instance of C, where $v = \arg \max_{c_j \in C} d(x, c_j)$ $1C = C - \{c_v\}$
4 $C = C - \{c_v\}$;
5 Update k-nearest neighbor set NN_k of x to generate a new set NN_{new} by comparing the similarity between x and each instance of T_v
6 $\Delta = (NN_k - NN_{new}) \bigcup (NN_{new} - NN_k)$;
7 $NN_k = NN_{new}$;
end
8 Obtain the predicted label $\hat{y}$ based on the majority class on NN_k
9 **Return**$\hat{y}$.

Algorithm 1: NPR-kNN algorithm.

training data is divided into several disjoint subsets because there is no intersection among these divided subsets. Therefore, the proposed algorithm can effectively deal with large-scale data.

4. Experiments

To test the proposed algorithm, an extensive experiment comparison has been carried out on the real datasets with two representative kNN classification acceleration algorithms based on data partition.

4.1. Experiment Setup. Two representative algorithms are selected in this paper: kNN classification algorithm based on KD-tree (MKD-kNN) and kNN classification algorithm based on PCA tree (PCA-kNN) [15]. Meanwhile, ten large-scale public datasets are chosen to make a fair comparison with other algorithms to verify the effectiveness of the proposed algorithm [30, 31], where the scale of each dataset is greater than 90000. Information of the ten selected datasets is shown in Table 1.

The NPR-kNN algorithm, MKD-kNN algorithm, and PCA-kNN algorithms are all approximations of the kNN classification algorithm. To evaluate the degree of consistency of k-nearest neighbors of each training instance before and after the training set is divided, the training matching ratio $R_{tr} = N_{tr}/N$, where N_{tr} is the number of instances whose k-nearest neighbors in the divided subsets are the same as the training set and N is the size of the training set. The larger the value of R_{tr}, the stronger the locality of the data maintained by the algorithm; otherwise, the weaker the locality of the data maintained by the algorithm. The test accuracy is an important index to evaluate the performance of the classifier; it mainly characterizes whether the label of the test instance is consistent with the predicted label. However, it does not reflect whether the nearest neighbor sequence of the test instance obtained by the approximate nearest neighbor algorithm is consistent with that obtained by the original kNN algorithm. To this end, we also calculated the test matching ratio $R_{tr} = N_{ts}/N_{\text{test}}$, where N_{ts} is the number of test instances whose k-nearest neighbors obtained by the approximate nearest neighbor algorithm are the same as the ones obtained by the original kNN algorithm, and N_{test} is the number of all test instances. A tenfold cross-validation method is used to estimate three performance index values on different datasets. In addition, the signed-rank test [32, 33] is adopted to test whether there is a significant difference in performance between the NPR-kNN algorithm and other algorithms.

Table 1: Summary of datasets.

Dataset	Size	Features	Class
Acoustic	98528	50	3
Aloi	108000	128	1000
Cifa	60000	3072	2
Combined	98528	100	3
MiniBooNE	94158	50	2
Mnist	350000	95	2
Poker	1025010	10	10
Seismic	98528	50	3
Skin-noskin	245057	3	2
Webspam	350000	254	2

In the following experiments, all attribute values of the used datasets are normalized to the interval $[0, 1]$ to avoid the influence of dimensions between different attributes. The Euclidean distance is used to measure the similarity between instances. The performance of the approximate nearest neighbor algorithm based on the data partition is affected by the size of the subset, so they need to be compared under different numbers of divided subsets. We choose these four different values $s = 500,1000,2000,5000$ as the threshold of the divided subset size according to the suggestion of the paper [17]. It determines the number of divided subsets using the formula $\lceil n/s \rceil$, where $\lceil n/s \rceil$ is the minimum positive integer large than n/s. Moreover, $k = 7$ is chosen based on the experiment result of the paper [34]. The significant level is $\alpha = 0.05$.

Table 2: R_{tr} of different algorithms under $s = 500$.

Dataset	NPR-kNN	MKD-kNN	PCA-kNN
Acoustic	0.357	0.117	0.205
Aloi	0.582	0.386	0.396
Cifa	0.068	0.005	0.022
Combined	0.287	0.079	0.135
MiniBooNE	0.382	0.126	0.123
Mnist	0.665	0.321	0.35
Poker	0.132	0.081	0.076
Seismic	0.342	0.17	0.2
Skin-noskin	0.956	0.907	0.915
Webspam	0.752	0.327	0.354
Mean	0.460	0.252	0.278
Median	0.370	0.148	0.203
Wilcoxon p		0.002	0.002

4.2. Experiment Analysis. The following experiment analysis is made from the three indicators of training matching ratio, test matching ratio, and test accuracy.

4.2.1. Training Matching Ratio. The training matching ratio R_{tr} is the measurement to evaluate the consistency of k -nearest neighbors of all the training instances searched using data partition. To make a fair comparison with other algorithms, each training instance finds its k-nearest neighbors in the single divided subset rather than the union of some divided subsets. Tables 2–5 list the training matching ratio R_{tr} of different algorithms under different s, where two descriptive statistics and the p value of the Wilcoxon signed-rank test between the NPR-kNN algorithm and one of the other algorithms are also listed in the last three lines of the tables.

The following experiment analysis is made from the value of s, because it takes a great effect to the measurement R_{tr}. From the results of Tables 2–4 under three different smaller values of s, the value of R_{tr} of the NPR-kNN algorithm is serval times than the MKD-kNN algorithm and PCA-kNN algorithm on most datasets except the Skin-noskin dataset. Besides the value of s, the sparseness of the data distribution also takes an effect on the value R_{tr}. If most instances of the dataset are distributed densely in the input space, then data partition takes a small effect on the nearest neighbor relationship preservation, and the value R_{tr} could be large. The dataset Skin-noskin is relatively densely distributed and the divided subsets with hundreds of data so that the value R_{tr} of all three algorithms have larger than 0.9 under different values of s on it. Meanwhile, the mean and median of the NPR-kNN algorithm on different datasets are close to or greater than 0.5 and have the largest value among the three algorithms under $s = 500,1000,2000$. Finally, the p value of the Wilcoxon signed-rank test between the NPR-kNN algorithm and one of the other algorithms is smaller than the given significant level $\alpha = 0.05$. Therefore, the NPR-kNN algorithm obtains the best result of the measurement R_{tr} compared with the MKD-kNN algorithm and PCA-kNN algorithm under the smaller s.

Table 3: R_{tr} of different algorithms under $s = 1000$.

Dataset	NPR-kNN	MKD-kNN	PCA-kNN
Acoustic	0.445	0.164	0.293
Aloi	0.626	0.378	0.412
Cifa	0.078	0.005	0.042
Combined	0.412	0.119	0.206
MiniBooNE	0.481	0.24	0.173
Mnist	0.728	0.335	0.404
Poker	0.126	0.111	0.076
Seismic	0.405	0.227	0.136
Skin-noskin	0.985	0.907	0.923
Webspam	0.756	0.617	0.444
Mean	0.504	0.310	0.311
Median	0.463	0.234	0.250
Wilcoxon p		0.002	0.002

Table 4: R_{tr} of different algorithms under $s = 2000$.

Dataset	NPR-kNN	MKD-kNN	PCA-kNN
Acoustic	0.551	0.230	0.395
Aloi	0.723	0.456	0.543
Cifa	0.254	0.013	0.085
Combined	0.525	0.167	0.306
MiniBooNE	0.584	0.279	0.236
Mnist	0.821	0.120	0.508
Poker	0.232	0.180	0.118
Seismic	0.492	0.300	0.357
Skin-noskin	0.961	0.944	0.942
Webspam	0.812	0.657	0.484
Mean	0.596	0.335	0.397
Median	0.568	0.255	0.376
Wilcoxon p		0.002	0.002

Table 5: R_{tr} of different algorithms under $s = 5000$.

Dataset	NPR-kNN	MKD-kNN	PCA-kNN
Acoustic	0.755	0.745	0.749
Aloi	0.921	0.876	0.887
Cifa	0.719	0.715	0.714
Combined	0.828	0.824	0.825
MiniBooNE	0.880	0.878	0.876
Mnist	0.981	0.981	0.976
Poker	0.538	0.539	0.536
Seismic	0.717	0.716	0.714
Skin-noskin	1.000	1.000	1.000
Webspam	0.982	0.975	0.972
Mean	0.832	0.825	0.825
Median	0.854	0.850	0.851
Wilcoxon p		0.023	0.004

Table 6: R_{ts} of different algorithms under $s = 500$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN
Acoustic	0.755	0.113	0.200
Aloi	0.908	0.389	0.401
Cifa	0.368	0.006	0.023
Combined	0.828	0.076	0.135
MiniBooNE	0.879	0.130	0.124
Mnist	0.979	0.321	0.346
Poker	0.536	0.088	0.076
Seismic	0.716	0.164	0.199
Skin-noskin	0.998	0.907	0.913
Webspam	0.937	0.331	0.350
Mean	0.790	0.256	0.278
Median	0.854	0.147	0.200
Wilcoxon p		0.002	0.002

Table 7: R_{ts} of different algorithms under $s = 1000$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN
Acoustic	0.755	0.164	0.290
Aloi	0.914	0.378	0.412
Cifa	0.398	0.005	0.043
Combined	0.827	0.119	0.206
MiniBooNE	0.880	0.240	0.172
Mnist	0.980	0.335	0.401
Poker	0.534	0.111	0.075
Seismic	0.718	0.227	0.136
Skin-noskin	0.995	0.907	0.924
Webspam	0.979	0.617	0.440
Mean	0.798	0.310	0.310
Median	0.854	0.234	0.248
Wilcoxon p		0.002	0.002

For the result in Table 5 under the large value $s = 5000$, three algorithms have the similarity value of R_{tr} on all the datasets except the Acoustic and Aloi datasets. And the mean of R_{tr} of these algorithms is 0.832, 0.825, and 0.825, and their median values are 0.854, 0.850, and 0.851. Moreover, these algorithms get larger values of R_{tr} under $s = 5000$ than the results under $s = 500, 1000, 2000$. The reason for this issue is that there are lots of elements in the divided subset under the large value of s which largely increases the probability that each element and k-nearest neighbors are still in the same divided subset. However, the p value of the Wilcoxon signed-rank test between the NPR-kNN algorithm and one of the other algorithms is smaller than the given significant level $\alpha = 0.05$. Therefore, there exists a significant difference between the NPR-kNN algorithm and one of the other algorithms, and the NPR-kNN algorithm also obtains the best result under $s = 5000$. In conclusion, the experiment result shows that the NPR-kNN algorithm largely keeps the instances and their k-nearest neighbors still in the same divided subsets, and it also verifies the correctness of Theorem 2.

4.2.2. Test Matching Ratio. Besides the training performance measured by the training matching ratio R_{tr}, we pay more attention to the test performance of the algorithm. Test matching ratio R_{ts} is adopted to measure the extent that the test instances and their k-nearest neighbors are still in the same divided subset, and it also evaluates whether the improved algorithm using data partition can obtain a similar performance of the original k-nearest neighbor classification algorithm. Tables 6–9 list the R_{ts} results of three algorithms under different values of s, and the statistical results are also listed in the last three lines of the tables.

The results of Tables 6–9 show that the value of R_{ts} of the NPR-KNN algorithm is several times larger than that of other algorithms on all the datasets except the Skin-noskin dataset under different values of s. The values of R_{ts} on the Skin-noskin dataset of three algorithms are larger than 0.9; this is because its elements are distributed densely in the input space, and each divided subset has hundreds of ele-

Table 8: R_{ts} of different algorithms under $s = 2000$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN
Acoustic	0.755	0.388	0.227
Aloi	0.917	0.547	0.456
Cifa	0.781	0.088	0.013
Combined	0.828	0.302	0.161
MiniBooNE	0.880	0.231	0.280
Mnist	0.980	0.505	0.452
Poker	0.537	0.116	0.180
Seismic	0.716	0.354	0.294
Skin-noskin	0.999	0.941	0.944
Webspam	0.980	0.480	0.651
Mean	0.837	0.395	0.366
Median	0.854	0.371	0.287
Wilcoxon p		0.002	0.002

Table 9: R_{ts} of different algorithms under $s = 2000$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN
Acoustic	0.754	0.297	0.498
Aloi	0.919	0.517	0.647
Cifa	0.851	0.033	0.108
Combined	0.827	0.223	0.398
MiniBooNE	0.879	0.385	0.300
Mnist	0.981	0.512	0.614
Poker	0.534	0.418	0.305
Seismic	0.717	0.379	0.450
Skin-noskin	0.999	0.960	0.958
Webspam	0.997	0.711	0.670
Mean	0.846	0.444	0.495
Median	0.865	0.402	0.474
Wilcoxon p		0.002	0.002

Table 10: The test accuracy of different algorithms under $s = 500$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN	kNN
Acoustic	0.756	0.723	0.736	0.756
Aloi	0.913	0.842	0.838	0.916
Cifa	0.722	0.709	0.718	0.728
Combined	0.828	0.805	0.802	0.828
MiniBooNE	0.880	0.865	0.861	0.880
Mnist	0.979	0.969	0.971	0.981
Poker	0.542	0.531	0.535	0.546
Seismic	0.721	0.712	0.716	0.718
Skin-noskin	0.998	0.996	0.998	0.999
Webspam	0.980	0.976	0.975	0.982
Mean	0.832	0.813	0.815	0.833
Median	0.854	0.824	0.820	0.854
Wilcoxon p		0.002	0.004	0.125

Table 11: The test accuracy of different algorithms under $s = 1000$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN	kNN
Acoustic	0.757	0.742	0.744	0.756
Aloi	0.958	0.852	0.854	0.916
Cifa	0.724	0.709	0.718	0.728
Combined	0.829	0.818	0.821	0.828
MiniBooNE	0.828	0.818	0.811	0.880
Mnist	0.983	0.970	0.974	0.981
Poker	0.545	0.535	0.534	0.546
Seismic	0.725	0.714	0.715	0.718
Skin-noskin	0.998	0.995	0.996	0.999
Webspam	0.982	0.975	0.972	0.982
Mean	8.329	0.813	0.814	0.833
Median	0.829	0.818	0.816	0.854
Wilcoxon p		0.002	0.002	0.805

ments. Meanwhile, the mean and median of R_{ts} of the NPR-kNN algorithm on different datasets are close to or greater than 0.8 and have the largest value among three algorithms under $s = 500, 1000, 2000, 5000$. Finally, the p value of the Wilcoxon signed-rank test between the NPR-kNN algorithm and one of the other algorithms is smaller than the given significant level $\alpha = 0.05$. Therefore, the NPR-kNN algorithm obtains the best result of the measurement R_{ts} compared with the MKD-kNN algorithm and PCA-kNN algorithm. The NPR-kNN algorithm searches k-nearest neighbors from the union of several divided subsets rather than only one divided subset. This increases the probability that the test instance and its k-nearest neighbors are still in the candidate subset.

On the other hand, the value of the parameter s has a different effect on the values of R_{ts} of these algorithms. The NPR-KNN algorithm has the value of R_{ts} with a small change on each data under different values of parameter s, while other algorithms have a large difference in the value of R_{ts}. This fact is that these algorithms use different numbers of divided subsets to find k-nearest neighbors, and the parameter s controls the number of instances in the divided subset. Each test instance finds its k-nearest neighbors using only one divided subset for both MKD-kNN algorithm and PCA-kNN algorithm, and then, this takes a great effect on the value of R_{ts}. k-nearest neighbors are continually updated by successively merging the divided subsets until they do not change rather than only one divided subset, and this operation greatly reduces the effect of the parameter s to the NPR-KNN algorithm. Therefore, the performance of the NPR-KNN algorithm on R_{ts} is not sensitive to the value of s, and this advantage increases its availability for dealing with practical problems.

4.2.3. Test Performance. The generation ability can be measured by the classification accuracy on the test data, which is the most commonly used performance indicator. Tables 10–13 list the results of three algorithms under different values of the parameter s.

The results on Tables 10–13 also show that the classification accuracy of the NPR-KNN algorithm is not less than

Table 12: The test accuracy of different algorithms under $s = 2000$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN	kNN
Acoustic	0.755	0.742	0.747	0.756
Aloi	0.918	0.865	0.872	0.916
Cifa	0.728	0.715	0.714	0.728
Combined	0.831	0.821	0.828	0.828
MiniBooNE	0.880	0.862	0.867	0.880
Mnist	0.984	0.972	0.975	0.981
Poker	0.545	0.542	0.533	0.546
Seismic	0.726	0.718	0.715	0.718
Skin-noskin	1.000	0.996	1.000	0.999
Webspam	0.980	0.975	0.972	0.982
Mean	0.835	0.821	0.822	0.833
Median	0.856	0.842	0.848	0.854
Wilcoxon p		0.002	0.002	0.242

Table 13: The test accuracy of different algorithms under $s = 5000$.

Dataset	NPR-KNN	MKD-kNN	PCA-kNN	kNN
Acoustic	0.765	0.745	0.749	0.756
Aloi	0.911	0.876	0.887	0.916
Cifa	0.729	0.715	0.714	0.728
Combined	0.832	0.824	0.825	0.828
MiniBooNE	0.882	0.878	0.876	0.880
Mnist	0.985	0.981	0.976	0.981
Poker	0.538	0.539	0.536	0.546
Seismic	0.728	0.716	0.714	0.718
Skin-noskin	1.000	1.000	1.000	0.999
Webspam	0.983	0.975	0.972	0.982
Mean	0.835	0.825	0.825	0.833
Median	0.857	0.850	0.851	0.854
Wilcoxon p		0.002	0.004	0.223

that of the MKD-kNN algorithm and PCA-kNN algorithm on all the datasets under different values of s, and it obtains the similar classification accuracy as the kNN classification algorithm. Moreover, the NPR-KNN algorithm has much better test performance than the MKD-kNN algorithm and PCA-kNN algorithm on the multiclassification dataset Aloi with 1000 classes. For the mean and median of classification accuracy on all the datasets, the NPR-KNN algorithm has a better performance than other improved algorithms and obtains similar results. Finally, the p values of the Wilcoxon signed-rank test between the NPR-kNN algorithm and one of the other improved algorithms are smaller than the given significant level 0.05 under different values of s. Then, the NPR-kNN algorithm achieves better classification than them. There exists no significant difference between NPR-kNN algorithm and kNN classification algorithm because the p values between them are larger than the given significant level 0.05. The reason for this fact is that the NPR-kNN algorithm obtains k-nearest neighbors that are most likely to be the same as the original algorithm compared with other algorithms, and this conclusion is also verified in the above subsection experience results.

5. Summary

We have proposed a novel algorithm to explore the effect of data partition on the classification performance of kNN classification algorithm, which could largely keep k same nearest neighbors as the original algorithm. Different from previous improved kNN classification algorithms based on data partition, the proposed algorithm theoretically studies the effect of data partition from the perspective of optimization, and it proves that the similarity of instances within the different partitioned subsets to be smaller is the key factor for the generation ability of the classifier. To this end, the minibatch k-Means clustering algorithm is adopted to execute the data partition for its high efficiency and effectiveness, and an early stopping rule is designed to search k-nearest neighbors from the divided subsets. Moreover, it can effectively deal with large-scale data for its linear time complexity. Experiment results on multiple real datasets show that the proposed algorithm gets the similar k-nearest neighbors and classification performance with the original kNN classification algorithm and better results than two state-of-the-art algorithms. The method in this paper takes a paradigm to handle large-scale data, and it also offers a promising way to scalable algorithms based on data partition. In future work, we will study how to combine the result of multiple data partitions to improve the performance of the kNN classification algorithm.

Data Availability

All the used datasets can be downloaded from the LIBSVM dataset (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/) and UCI machine learning repository (https://archive.ics.uci.edu/ml/index.php).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was also supported by the project ZR2020MF146 supported by Shandong Provincial Natural Science Foundation, China, Major Scientific and Technological Innovation Project of Shandong Province (No. 2019JZZY010716), and Open Project Foundation of Intelligent Information Processing Key Laboratory of Shanxi Province (No. CICIP2021002).

References

[1] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, MIT press, 2018.

[2] 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 Recognition*, vol. 74, pp. 531–543, 2018.

[3] J. Gou, W. Qiu, Z. Yi, X. Shen, Y. Zhan, and W. Ou, "Locality constrained representation-based k-nearest neighbor classification," *Knowledge-Based Systems*, vol. 167, pp. 38–52, 2019.

[4] E. Prasetyo, R. Purbaningtyas, and R. Adityo, "Cosine K-nearest neighbor in milkfish eye classification," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 3, pp. 11–25, 2020.

[5] B. Tang, H. He, and S. Zhang, "Mcenn: a variant of extended nearest neighbor method for pattern recognition," *Pattern Recognition Letters*, vol. 133, pp. 116–122, 2020.

[6] D. Jiangyi and F. Bian, "A privacy-preserving and efficient k-nearest neighbor query and classification scheme based on k-dimensional tree for outsourced data," *IEEE Access*, vol. 8, pp. 69333–69345, 2020.

[7] X. Wu, V. Kumar, J. Ross Quinlan et al., "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[8] E. Marchiori, "Class conditional nearest neighbor for large margin instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 364–370, 2010.

[9] Y. Li and L. Maguire, "Selecting critical patterns based on local geometrical and statistical information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 6, pp. 1189–1201, 2011.

[10] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[11] T. Liu, A. W. Moore, A. G. Gray, and K. Yang, "An investigation of practical approximate nearest neighbor algorithms," in *Advances in Neural Information Processing Systems*, pp. 825–832, MIT Press, Vancouver, Canada, 2004.

[12] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, 2012.

[13] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time,"

Association for Computing Machinery transactions on mathematical software, vol. 3, no. 3, pp. 209–226, 1977.

[14] N. Verma, S. Kpotufe, and S. Dasgupta, "Which Spatial Partition Trees Are Adaptive to Intrinsic Dimension?," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 565–574, Montreal, Canada, 2009.

[15] B. McFee and G. R. G. Lanckriet, "Large-scale music similarity search with spatial trees," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pp. 55–60, Miami, USA, 2011.

[16] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 128–131, 2008.

[17] C. García-Osorio, A. de Haro-García, and N. García-Pedrajas, "*Democratic* instance selection: a linear complexity instance selection algorithm based on classifier ensemble concepts," *Artificial Intelligence*, vol. 174, no. 5-6, pp. 410–441, 2010.

[18] A. de Haro-García, N. García-Pedrajas, and J. A. R. del Castillo, "Large scale instance selection by means of _federal_ instance selection," *Data & Knowledge Engineering*, vol. 75, pp. 58–77, 2012.

[19] C.-J. Hsieh, S. Si, and I. Dhillon, "A divide-and-conquer solver for kernel support vector machines," in *Proceedings of the 31st International Conference on Machine Learning*, pp. 566–574, Beijing, China, 2014.

[20] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515-516, 1968.

[21] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1450–1464, 2007.

[22] F. Angiulli and G. Folino, "Distributed nearest neighbor-based condensation of very large data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 12, pp. 1593–1606, 2007.

[23] K. Nikolaidis, J. Y. Goulermas, and Q. H. Wu, "A class boundary preserving algorithm for data condensation," *Pattern Recognition*, vol. 44, no. 3, pp. 704–715, 2011.

[24] H. Zhang and G. Sun, "Optimal reference subset selection for nearest neighbor classification by tabu search," *Pattern Recognition*, vol. 35, no. 7, pp. 1481–1490, 2002.

[25] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on world wide web*, pp. 1177-1178, Raleigh, USA, 2010.

[26] A. Hidayat, D. Jamaluddin, and D. S. Maylawati, "Data analytics for effectiveness evaluation of islamic higher educationusing k-means algorithm," *International Journal of Advanced Science and Technology*, vol. 29, no. 3, pp. 4149–4161, 2020.

[27] T. Li, Y. Ma, and T. Endoh, "Normalization-based validity index of adaptive k-means clustering for multi-solution application," *Ieee Access*, vol. 8, pp. 9403–9419, 2020.

[28] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society*, vol. 63, no. 2, pp. 411–423, 2001.

[29] S. C. Hicks, R. Liu, Y. Ni, E. Purdom, and D. Risso, "Mbkmeans: fast clustering for single cell data using mini-batch k-means," *PLoS Computational Biology*, vol. 17, no. 1, pp. 1–18, 2021.

[30] C.-C. Chang and C.-J. Lin, "LIBSVM," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[31] K. Bache and M. Lichman, "Uci machine learning repository," 2021, http://archive.ics.uci.edu/ml/index.php.

[32] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*, pp. 196–202, Springer, 1992.

[33] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[34] M. Kordos, M. Blachnik, and D. Strzempa, "Do we need whatever more than k-nn?," in *International Conference on Artificial Intelligence and Soft Computing*, pp. 414–421, Heidelberg, Germany, 2010.