Postprint: Distributed Exact Fuzzy KNN Classification Algorithm for Big Data

Authors: Zou Jinsong, Li Fang

Date: 2018-10-11T00:00:00+00:00

Abstract

This study investigates the efficiency issues of the K-Nearest Neighbors (KNN) method when handling large datasets and proposes a distributed exact fuzzy KNN classification algorithm based on the Spark framework. The method innovatively integrates the distributed map and reduce processes of the Spark framework with fuzzy KNN. Initially, the category information of training samples across different partitions is fuzzified to obtain class membership degrees, thereby transforming the training set into a fuzzy training set augmented with class membership. Subsequently, the KNN algorithm is employed to compute k nearest neighbors for the test set based on previously calculated class memberships. Finally, classification is performed using distance weighting. Experimental results on million-scale large datasets, along with comparative experiments against other algorithms, demonstrate that the proposed algorithm is both feasible and effective.

Full Text

Preamble

Title: Accurate Distributed Fuzzy KNN Classification Algorithm for Big Data

Authors: Zou Jinsong[1], Li Fang[2]
[1]Dept. of Electronics & Information Engineering, Chongqing College of Water Resources & Electric Engineering, Chongqing 402160, China
[2]School of Computer Science, Chongqing University, Chongqing 400044, China

Abstract: This paper addresses the efficiency issues of the K-Nearest Neighbor (KNN) method when processing large datasets and proposes a distributed exact fuzzy KNN classification algorithm based on the Spark framework. The method innovatively combines Spark' s distributed map and reduce processes with fuzzy KNN. First, the category information of training samples in different partitions is fuzzified to obtain class membership degrees, transforming the training set into

a fuzzy training set with added class membership. Then, the KNN algorithm is used to compute the k nearest neighbors for the test set based on previously calculated class memberships. Finally, classification is performed using distance weighting. Experiments on million-scale datasets and comparative studies with other algorithms demonstrate that the proposed algorithm is both feasible and effective.

Keywords: big data; distributed Spark framework; class membership degree; exact fuzzy KNN algorithm

0 Introduction

The K-Nearest Neighbor (KNN) method is an effective classification algorithm whose decision rule is based on identifying the k most similar samples from the training set for an unknown sample. Similarity is typically measured using distance metrics such as Euclidean distance or Manhattan distance. Despite its simplicity, KNN has become one of the top ten algorithms in data mining due to its performance advantages. However, KNN assigns equal importance to all neighbors during classification, which may lead to misclassification. Consequently, numerous improvements to KNN have been investigated. Fuzzy-KNN addresses this limitation of traditional KNN through fuzzy set theory, thereby improving classification accuracy and has been applied in medicine, economics, and many other fields.

KNN faces two primary challenges when processing large datasets: runtime efficiency and memory consumption. While Fuzzy-KNN achieves higher classification accuracy than standard KNN, it requires an additional stage to compute class membership degrees, resulting in even greater time and memory overhead. To address this issue, research on KNN and its variants for big data processing has gradually increased. Reference [8] proposes a two-stage approximate KNN algorithm that first partitions the data into different segments and then computes KNN within each partition to obtain preliminary results. Reference [9] introduces a fast KNN classification algorithm for big data that first uses clustering to divide samples into blocks, identifies the block nearest to the test sample, and then applies KNN to that block as a new training set. Reference [10] presents an exact KNN method for large datasets that splits the training set and computes KNN for each test sample in the map phase, then collects all candidates as nearest neighbors in the reduce phase to report the final k nearest neighbors, enabling it to handle large training and test sets with the same accuracy as traditional KNN while managing long running times. Reference [11] applies Fuzzy-KNN to each data partition and collects all labels from each group, computing the final result through majority voting, though this approach requires too many partitions and generates extensive voting calculations when both training and test sets are large.

With the development of cloud computing, research on big data classification

has become a hotspot. Reference [12] proposes a Storm-based streaming data KNN classification algorithm that meets the requirements of high throughput, real-time performance, and accuracy for streaming data classification in big data contexts. Reference [13] presents a big data classification system based on linguistic fuzzy rules using the MapReduce framework to learn and fuse rule bases, effectively achieving big data classification. Reference [14] introduces a new big data fuzzy rule classification system that addresses the significant performance degradation problem as parallelism increases. Reference [15] proposes a multi-layer differential KNN algorithm for big data that avoids discrimination errors caused by sample editing in traditional improved algorithms while substantially reducing invalid computations. Reference [16] presents a MapReduce and distributed cache-based KNN parallel method that improves algorithmic efficiency.

Building upon these big data classification methods and addressing their limitations, this paper proposes an Exact Fuzzy KNN (EF-KNN) algorithm implemented on Spark. The algorithm utilizes Spark' s in-memory primitives to manage large training sets through data partitioning and processes massive test sets by traversing blocks of this collection.

---

1 Fuzzy KNN Algorithm

The fuzzy KNN algorithm is an improvement over standard KNN that demonstrates excellent performance in terms of accuracy. Unlike traditional KNN, this method fuzzifies the distances between unknown samples and their k nearest neighbors, establishing membership degrees for each class. It uses the training set to pre-compute class memberships before calculating the KNN for each sample in the test set.

Let TR be the training dataset and TS be the test set, each consisting of n samples, where each sample x_i is a vector $\langle x_{i1}, x_{i2}, ..., x_{ij}, ..., x_{it} \rangle$ and x_{ij} represents the j-th feature value of the i-th sample. Each sample in TR belongs to a known class w_i, while the class membership for TS is unknown. Fuzzy KNN operates in two distinct stages. The first stage computes the k nearest neighbors of TR itself by searching for the k closest samples through distance calculations between all samples in TR and TS. When the nearest neighbors are identified, class membership is created as shown in equation (1), transforming the training dataset TR to have a class membership vector instead of original class labels.

The second stage computes the k nearest neighbors similarly to the first stage, but differs in that it calculates the k nearest neighbors for each sample in TS within TR, ultimately classifying according to equation (2).

The class membership degree $u_i(x)$ for a training sample x belonging to class i is calculated as:

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij} \cdot \frac{1}{\|x-x_j\|^{2/(m-1)}}}{\sum_{j=1}^{K} \frac{1}{\|x-x_j\|^{2/(m-1)}}}$$

where m is the fuzzy weight adjustment factor. If x_j belongs to class i, then $u_{ij} = 1$; otherwise, $u_{ij} = 0$. When $m > 1$, we can determine that sample x belongs to class i.

---

2 Exact Fuzzy KNN (EF-KNN) Algorithm for Big Data

Spark is a new implementation of MapReduce that addresses several limitations of Hadoop. Its most important feature is that data structures are processed in parallel in a transparent manner through Resilient Distributed Datasets (RDDs). Additionally, RDDs allow for data persistence and reuse. Furthermore, Spark works in conjunction with Hadoop, particularly with its distributed file system (HDFS).

This paper proposes a distributed exact fuzzy KNN classification algorithm for big data environments based on the Spark framework. The algorithm consists of two stages: the first stage computes class membership degrees, and the second stage performs classification.

[Figure 1: see original paper]

As shown in the figure, the computation process is divided into map and reduce phases. The map phase partitions the training dataset TR and calculates distances for each partition, extracting the class labels of the k nearest neighbors for each training sample. The reduce phase collects all candidates into the k nearest neighbors and obtains the k closest defined samples, forming a new fuzzy training set with added class membership degree vectors, referred to as the Fuzzy Training Set (FTS).

2.1 Computing Class Membership

This process is a MapReduce operation for computing class membership degrees of the training set. The training set TR read from HDFS begins as an RDD object and is divided into m partitions as a user-defined parameter. Consequently, for each partition, there is a map task $\langle map_1, map_2, ..., map_m \rangle$, with each map containing approximately the same number of training samples.

Each map process compares all training samples in its partition TR_j with all training samples in TR to identify the k nearest neighbors for each training sample. Assuming TR_j and TR can be used together in memory; otherwise, TR will be partitioned into v blocks and iterated sequentially to enable proper execution with in-memory storage. Algorithm 1 contains the pseudo-code for the map function.

Algorithm 1: Map Function for Membership Computation

1. Input: TR_j, TR_v, k_{memb}
2. For each training sample t in TR_j:
   - Compute $KNN(TR_j, TR_v, k_{memb}) \rightarrow \langle Class, Dist\rangle_{t,j}$
   - Emit $\langle key : t, value : \langle Class, Dist\rangle_{t,j}\rangle \rightarrow result_j$
3. End for
4. Output $result_j$

Each map process constructs a vector $\langle class, distance\rangle$ for every training sample in TR_j. Step 2 computes the class labels and distances to the k nearest neighbors. To accelerate the reducer' s implementation of nearest neighbor computation, each vector is sorted in ascending order by distance. Each map task constructs a matrix representing candidate samples as nearest neighbors, obtained through the process in step 3. This scheme enables multiple reducers to process large training sets.

Algorithm 2: Reduce Pseudo-code for Membership Computation

1. Input: $result_{key}$, k_{memb}, $cont1 = 0$, $cont2 = 0$
2. While $i < k_{memb}$ do:
   - If $result_{key}(cont1).Dist \leq result_{reducer}(cont2).Dist$ then
     - $result_{key}(cont1).Dist \rightarrow out(i)$, $cont1++$
   - Else if $result_{key}(cont1).Dist = result_{reducer}(cont2).Dist$ then
     - If $i < k_{memb}$ then
       - $result_{key}(cont2) \rightarrow out(i)$, $cont2++$
     - End if
     - $result_{key}(cont2) \rightarrow out(i)$, $cont2++$
   - End if
3. End while
4. Output out

Multiple reducers obtain the output from map tasks, partitioned to compute the k nearest neighbors for each sample in TR_v. The goal is to acquire the nearest neighbors for each training sample contained in TR_v. To achieve this, all elements are grouped by key, and class membership degrees are calculated. The reduce task updates the k_{memb} nearest neighbor candidate selection by merging the map outputs. Since vectors from the map are sorted by distance, the update process becomes more efficient, involving the merging of two sorted lists while preserving neighbors with identical distances whenever possible. This function compares distances one by one: if a distance is smaller than the current distance, it updates the distance and class; if the distance is larger, it is discarded; if distances are identical, both are preserved. The final result yields the k_{memb} nearest neighbors for each sample in the training set. Another map phase then computes class memberships using equation (1), transforming the original label representing each class membership vector. Finally, it returns each original sample with its preserved features and modified labels for computed class memberships, producing a new fuzzy training set that serves as

input for the classification phase in Section 2.2.

2.2 Classification Process

[Figure 2: see original paper]

The classification phase flow is also divided into MapReduce phases. The map phase requires the training dataset TR, test set TS, and parameter k as inputs. It partitions the data and computes the K-nearest neighbors for each partition of TR and every sample in TS, calculating distances. The reduce phase collects all k nearest neighbors from each partition, computes the k closest samples, and finally reports the classification label for each sample through majority voting.

The focus of the classification phase is to obtain exact results starting from the fuzzy training set FTR. The FTR is divided into m parts containing approximately the same number of samples, while the test set TS must remain unpartitioned to enable distributed map operations that compute all candidates as the k nearest neighbors for each partition of FTR. If the number of partitions in FTR matches that of TR, the shuffle phase is avoided, thereby improving runtime efficiency.

Algorithm 3: Map Pseudo-code for Classification Process

1. Input: TR_j, TS, k
2. For each test sample t in TS:
   - Compute $KNN(TR_j, TS, k) \rightarrow neigh\text{-}memb_{t,j}$
   - Emit $\langle key : t, value : neighbors_{t,j} \rangle \rightarrow result_t$
3. End for
4. Output $result_t$

For each sample in TS, the algorithm computes its k nearest neighbors. The variable *neigh-memb* stores distances and class membership vectors, with each test sample' s ID added as the key and sent to the reducer.

Algorithm 4: Reduce Pseudo-code for Classification Process

1. Parameters: $cand_{key,1}$, $cand_{key,2}$, k, $C1 = 0$, $C2 = 0$
2. While $i < k$ do:
   - If $cand_{key,1}(C1).Dist < cand_{key,2}(C2).Dist$ then
     - $cand_{key,1}(C1) \rightarrow result(i)$
   - If $cand_{key,1}(C1).Dist = cand_{key,2}(C2).Dist$ then
     - If $i < k$ then
       - $cand_{key,2}(C2) \rightarrow result(i)$
     - End if
   - End if
   - $cand_{key,2}(C2) \rightarrow result(i)$
3. End if
4. End while

The reduce process aims to aggregate all candidate nearest neighbors to finally obtain the k nearest neighbors of the entire fuzzy training set FTR while preserving distances and class membership degrees. Considering that some samples may have identical distances, both neighbors are retained when the k parameter allows. Finally, using the obtained neighbors, a last map function computes the predicted class labels by applying equation (2) to each sample in the test set.

---

3 Experimental Results and Analysis

Three large datasets from the UCI Machine Learning Repository (PokerHand, Susy, Higgs) were used to evaluate the proposed model. provides the dataset descriptions.

The parameters used in the model include k_{memb} (selected as 3, 5, 7 for neighbor count in class membership computation) and k (neighbor count for unknown samples), with the number of concurrent map tasks corresponding to the number of partitions in training dataset TR.

All experiments were conducted on a cluster of 16 compute nodes managed by a master node. Each node features 2 Intel Xeon CPU E5-2620 processors with 6 cores per processor (12 threads), 2 GHz clock speed, and 64 GB RAM, with 2 GB available main memory per task. Each node utilizes the Spark framework.

Dataset Descriptions

Runtime and accuracy analyses of the proposed method are presented in , which shows the class membership computation time (MembT), classification time (ClasT), total runtime (TotalT), and classification accuracy (Acc) using the Poker dataset. To simplify the presented results, k_{memb} and k values are identical and set to 3, 5, and 7, with the number of map tasks set to 256.

Experimental Results on Poker Dataset

The data in demonstrates that for parameters k_{memb} and k, the total runtime does not increase significantly across the three models despite sample size growth and increased neighbor computation in the reduce phase. Additionally, classification accuracy varies minimally when k_{memb} and k are set to 3, 5, or 7.

Experimental results for the Susy dataset with different map counts are shown in .

Experimental Results on Susy Dataset with Varying Map Counts

The data in indicates that big data classification accuracy decreases as the number of map tasks increases. This occurs because for test samples, several training data points have identical distances, and during distributed execution, the final neighbors depend on the arrival order from each map output.

[Figure 3: see original paper] illustrates the relationship between total runtime (in seconds) and the number of map tasks, with k_{memb} and k both equal to 3. [Figure 4: see original paper] shows the runtime (in seconds) for classification across the three datasets, with k_{memb} and k equal to 3 and the number of map tasks set to 384.

[Figure 3: see original paper] Impact of Map Task Count on Total Runtime

[Figure 4: see original paper] Runtime Across Different Datasets

Analysis of [Figure 3: see original paper] and [Figure 4: see original paper] reveals that the proposed model exhibits linear scalability. However, the Higgs dataset shows considerably higher runtime, exposing the algorithm' s weakness and indicating that more hardware resources are required during execution.

To validate the advancement of the proposed method, comparisons with other algorithms were conducted. In these experiments, both k_{memb} and k were set to 3 in the proposed algorithm. The classification accuracy comparison results are presented in .

Classification Accuracy Performance Comparison Across Different Algorithms (%)

The data in demonstrates that compared with other big data classification algorithms, the proposed algorithm achieves superior classification accuracy performance, confirming the advancement of the method.

---

4 Conclusion

To address the efficiency issues of big data classification, this paper proposes a scalable distributed exact fuzzy K-NN algorithm based on the Spark framework. The algorithm first partitions data on the Spark framework, computes class membership degrees to obtain a fuzzy dataset, and finally uses KNN to achieve large-scale data classification. Experiments demonstrate that the proposed method can classify data at the million-to-ten-million scale without significant increases in total runtime while maintaining classification precision. Comparative experiments with other algorithms further verify the feasibility and effectiveness of the proposed approach. Future work will focus on accelerating the fuzzy KNN model through an approximate approach, specifically targeting the reduction of class membership computation time.

References

[1] Ma Chuang, Wu Tao, Duan Mengya. Clustering algorithm based on membership degree of K-nearest neighbor [J]. Computer Engineering and Applications, 2016, 52(10): 55-58.

[2] Guo Xi, Wang Pan, Wang Jianyong, et al. Program multiple execution paths verification based on K-proximity weakest precondition [J]. Chinese Journal of Computers, 2015, 38(11): 2203-2214.

[3] Samanthula B K, Elmehdwi Y, Jiang W. K-nearest neighbor classification over semantically secure encrypted relational data [J]. IEEE Trans on Knowledge and Data Engineering, 2015, 27(5): 1261-1273.

[4] Kermani E F, Barani G A, Hessaroeyeh M G. Cavitation damage prediction on dam spillways using fuzzy-KNN modeling [J]. Journal of Applied Fluid Mechanics, 2018, 11(2): 323-329.

[5] Derrac J, Chiclana F, García S, et al. Evolutionary fuzzy K-nearest neighbors algorithm using interval-valued fuzzy sets [J]. Information Sciences, 2016, 329: 144-163.

[6] Chen H L, Huang C C, Yu X G, et al. An efficient diagnosis system for detection of Parkinson' s disease using fuzzy K-nearest neighbor approach [J]. Expert Systems with Applications, 2013, 40(1): 263-271.

[7] Jivani A G, Shah K, Koul S, et al. The adept K-nearest neighbour algorithm: an optimization to the conventional K-nearest neighbour algorithm [J]. Trans on Machine Learning and Artificial Intelligence, 2016, 4(1): 52-57.

[8] Deng Z, Zhu X, Cheng D, et al. Efficient KNN classification algorithm for big data [J]. Neurocomputing, 2016, 195(C): 143-148.

[9] Su Yijuan, Deng Zhenyun, Cheng Debo, et al. Fast KNN classification algorithm under big data [J]. Application Research of Computers, 2016, 33(4): 1003-1006.

[10] Maillo J, Ramírez S, Triguero I, et al. KNN-IS: an iterative spark-based design of the K-nearest neighbors classifier for big data [J]. Knowledge-Based Systems, 2017, 117: 1003-1006.

[11] Hegazy O, Safwat S, El Bakry M. A mapreduce fuzzy techniques of big data classification [C]// Proc of SAI Computing Conference. London, UK: IEEE Press, 2016: 118-128.

[12] Zhou Zhiyang, Feng Baiming, Yang Penglin, et al. Research and Implementation of KNN classification algorithm for streaming data based on Storm [J]. Computer Engineering and Applications, 2017, 53(19): 71-75.

[13] Río S D, López V, Benítez J M, et al. A MapReduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules [J]. International Journal of Computational Intelligence Systems, 2015, 8(3): 1003-1006.

[14] Elkano M, Galar M, Sanz J, et al. A global distributed approach to the Chi et al. fuzzy rule-based classification system for big data classification problems [C]// Proc of IEEE International Conference on Fuzzy Systems. Naples, Italy: IEEE Press, 2017: 1-6.

[15] Geng Lijuan, Li Xingyi. Improvements of KNN algorithm for big data classification [J]. Application Research of Computers, 2014, 31(5): 1342-1344.

[16] Tu Jingwei, Pi Jianyong. Parallelized K-nearest neighbor algorithm based on MapReduce and distributed cache [J]. Microcomputer & its Applications, 2015, 34(2): 18-21.

[17] Li Zhengjie, Huang Gang. Research on SVM KNN classification algorithm based on hadoop platform [J]. Computer Technology and Development, 2016, 26(3): 75-79.

Note: Figure translations are in progress. See original paper for figures.