BALKAN JOURNAL OF ELECTRICAL & COMPUTER ENGINEERING, Vol. 6, No. 2, April 2018

A Distributed K Nearest Neighbor Classifier for Big Data

T. Tulgar, A. Haydar and İ. Erşan

Abstract—**The K-Nearest Neighbor classifier is a well-known and widely applied method in data mining applications. Nevertheless, its high computation and memory usage cost makes the classical K-NN not feasible for today's Big Data analysis applications. To overcome the cost drawbacks of the known data mining methods, several distributed environment alternatives have emerged. Among these alternatives, Hadoop MapReduce distributed ecosystem attracted significant attention. Recently, several K-NN based classification algorithms have been proposed which are distributed methods tested in Hadoop environment and suitable for emerging data analysis needs. In this work, a new distributed Z-KNN algorithm is proposed, which improves the classification accuracy performance of the well-known K-Nearest Neighbor (K-NN) algorithm by benefiting from the representativeness relationship of the instances belonging to different data classes. The proposed algorithm relies on the data class representations derived from the Z data instances from each class, which are the closest to the test instance. The Z-KNN algorithm was tested in a physical Hadoop Cluster using several real-datasets belonging to different application areas. The performance results acquired after extensive experiments are presented in this paper and they prove that the proposed Z-KNN algorithm is a competitive alternative to other studies recently proposed in the literature**

Index Terms—**Big Data Classification, Hadoop, K-Nearest Neighbor, MapReduce.**

I. INTRODUCTION

In the age of the fourth industrial revolution, business's decision-making is highly based on the data retrieved by the internet-connected devices that are capable of collecting and processing ever-growing amounts of information [1].

A. T. TULGAR, is with Department of Computer Engineering, Girne American University, Girne, TRNC via Mersin 10 Turkey, (e-mail: tamertulgar@gau.edu.tr).

B. A. HAYDAR, is with Department of Computer Engineering, Girne American University, Girne, TRNC via Mersin 10 Turkey, (e-mail: ahaydar@gau.edu.tr).

C. İ. ERŞAN, is with Department of Computer Engineering, Girne American University, Girne, TRNC via Mersin 10 Turkey, (e-mail: ibrahimersan@gau.edu.tr).

Manuscript received August 29, 2017; accepted Nov 16, 2017.
DOI: 10.17694/bajece.419551

To achieve precise forecasts, data coming from different environments (e.g. different social media tools, data warehouses, cloud storages etc.) need to be intelligently analyzed by businesses.

Analyzing data retrieved from numerous data sources results in tackling with vast amounts of unstructured raw data, which are big in terms of volume, variety and velocity of acquisition. The process of analyzing such data is recently a popular research area, known as the Big Data Analysis [2].

One of the most important data mining tasks is classification. Classification, which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications [3].

To classify data in the big data age, centralized techniques lack the low classification delay performance, which is vital to cope with the high velocity data streams of big data.

To deal with this timing requirement of the modern data classification, several distributed ecosystems have been tested and used by different researches. One of these distributed ecosystems is popularly known as the Apache Hadoop and the Google's MapReduce Framework [4].

K Nearest Neighbor Classification (K-NN) has been one of the most popular classification algorithms [5]. The classical K-NN algorithm is based on calculating the distances between the test data instance to be classified and all of the instances in the training data set and finding the closest K number of training instances. After detecting the K number of closest training instances, the K-NN algorithm applies majority voting which is the process of detecting the data class with the maximum number of instances among the K selected instances.

Since the classical K-NN algorithm is completely based on individual instance proximities, it heavily suffers from high computation costs. In addition, since the algorithm's decision-making strategy is relying on the individual instance proximities rather than stronger class representations, the algorithm's classification accuracy is also not adequate for modern big data analysis that requires rapid and accurate classification results.

On the other hand, K-NN's individual instance distances strategy makes K-NN a strong candidate for distributed data classification, which is the basis of achieving acceptably low classification delays while classifying big data.

Taking into account the K-NN's suitability to distributed environments, many K-NN based studies which try to improve the K-NN algorithms performance, and working on Hadoop and MapReduce environment have been recently proposed in the literature [6-16].

In this paper, a new K-NN and MapReduce based algorithm is proposed named as the Z-KNN algorithm. The Z-KNN algorithm tries to remedy the classification accuracy performance of the classical K-NN classifier.

The main idea of the Z-KNN algorithm is to base the classification decision of the classical K-NN algorithm on the class representations by calculating the centroids of the closest Z training instances belonging to the classes of the K closest instances detected by the K-NN algorithm. In other words, the classical majority voting approach is replaced by a stronger classification decision, which is also computationally not expensive.

The rest of this paper is organized as follows: Section II presents the proposed Z-KNN algorithm in detail. The experimental setup and the achieved performance results are presented in section III. Finally, the section IV concludes the paper and states the future works.

II. THE PROPOSED Z-KNN ALGORITHM

In this section, the proposed Z-KNN algorithm and its MapReduce application will be explained.

A. *The Classical K-NN Algorithm*

In a classification task, if a data instance is considered as a vector of feature values, then a data instance i can be denoted as v_i which corresponds to a vector containing p features <$feat_1$, $feat_2$,...,$feat_p$>. Hence, a classification task can be defined as detecting the correct data classes of n test data instances $tsv_1,..tsv_n$ by using m training data instances $trv_1, \ldots trv_m$.

The classical K-NN algorithm is based on the simple idea of calculating the distances between a test data to be classified and all of the m number of data instances in the training set. After calculating all of the distances, the classical K-NN sorts the measured distances and uses the first K number of training Neighbors of the tested data.

The classification decision is then given by detecting which data class has the most number of instances among the selected K nearest neighbors, which is known as the majority voting.

As it can be deduced from the summary of the classical K-NN algorithm given above, the whole decision is based on the individual instance distances between all tsv_i and trv_j's.

Since the calculation of the instance distances is an independent task, being able to distribute the distance calculations to several processes makes the K-NN strategy suitable for distributed environments.

On the other hand, especially when a data set with high number of instances and high number of features per instance needs to be classified, the classical K-NN algorithm's classification accuracy performance becomes lower than its other well-known competitors, like K-Means classification [17].

Hence, it can be deduced that to become a classification algorithm suitable for modern data analysis needs, the K-NN's classification accuracy performance should be improved on a distributed environment.

Taking into account these needs, the Z-KNN algorithm is proposed and explained in sub-section B

B. *The Z-KNN Algorithm*

The proposed Z-KNN algorithm is a distributed K-NN based classification algorithm, which is designed to work on MapReduce environment.

Hadoop MapReduce is a software framework for easily writing applications that process vast amounts of data in-parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner [18].

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks [18].

In the MapReduce framework, any distributed task is designed as a combination of at least three functions: The Driver, Mapper and Reducer functions, which are inherited from the corresponding MapReduce classes [18].

The MapReduce framework of the Z-KNN algorithm for m

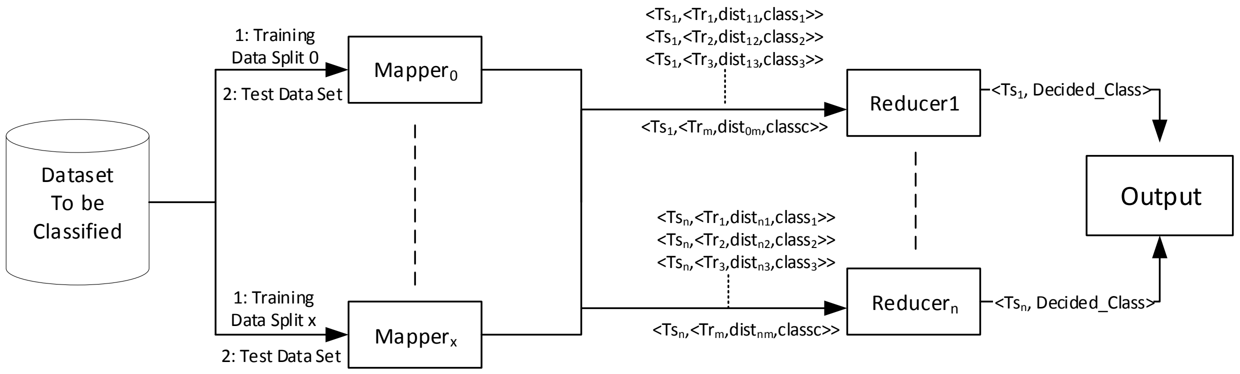
1: Training Data Split 0
2: Test Data Set
Mapper0
Dataset To be Classified
1: Training Data Split x
2: Test Data Set
Mapperx
<Ts1,<Tr1,dist11,class1>>
<Ts1,<Tr2,dist12,class2>>
<Ts1,<Tr3,dist13,class3>>
<Ts1,<Trm,dist0m,classc>>
<Tsn,<Tr1,distn1,class1>>
<Tsn,<Tr2,distn2,class2>>
<Tsn,<Tr3,distn3,class3>>
<Tsn,<Trm,distnm,classc>>
Reducer1
Reducern
<Ts1, Decided_Class>
<Tsn, Decided_Class>
Output

Figure 1. MapReduce Flow of the Z-KNN algorithm

number of instances in the training data set and n number of instances in the testing data set is presented in Figure 1.

As it can be seen in Figure 1, the Z-KNN algorithm's sub tasks are composed of the Mapper and the Reducer functions of the well-known MapReduce Framework.

The Mapper function is responsible of receiving the training data set splits from the MapReduce Driver, which is the main function for configuring the environment and managing the distributed processing running on top of the Hadoop framework, and calculating the distances between the testing instance to be processed and the training instances found in the received training data split.

According to the MapReduce Framework, the output of the Mapper function should be a <Key, Value> vector [18]. In the Z-KNN algorithm, the output Key of the Mapper is the testing instance id and the value is an object, which contains the used training instance, the distance between the testing and the training instances and the class id that the training instance belongs to.

Afterwards, the MapReduce framework shuffles the output <Key, Value> pairs emitted by the mapper functions so that the pairs with common keys are submitted to the same reducer function. In other words, a single reducer will process all of the calculated distances belonging to the same testing instance.

Algorithm 1: Z-KNN Mapper Function()

```
Input: <key, value>
key : the record id of the training instance
value: the set of feature values of the training instance

Output to MapReduce Env. : <key1, value1>
key1: the record id of the test instance
value1: a vector containing the training instance id, distance
and the class id of the training instance

1: class_tr = readClassId(value)
2: for i=1 to n
3:     //the loop to iterate each test instance
4:     $dist_{ij}$ = DistanceFunction($tr_j$, $ts_i$)
5:     Context.write(i, object <$tr_j$ , $dist_{ij}$ , class_tr>)
6: end for
7: return
```

1) The Mapper Function:

As explained above the mapper function is responsible of calculating the distances between the training and testing instances. The complete algorithm of the Z-KNN mapper function can be found in Algorithm 1.

As an example to distance calculation, if a single testing instance ts_i and a single training instance tr_j are considered, than the distance between these two instances is calculated by Equation (1).

$$dist_{ij} = \sqrt{\sum_{k=1}^{p}(tsi_{feat_k} - trj_{feat_k})^2} \qquad (1)$$

If we assume that the class id of the training instance tr_j is class A, then the output of a Z-KNN mapper becomes; <ts_i , <tr_j, $dist_{ij}$, class A>>.

Algorithm 2: Z-KNN Reducer Function()

```
Input: <key1, <List value1's> distances>
key1 : the record id of the test instance
value1: an object which contains <$tr_j$ , $dist_{ij}$ , class_$tr_j$>
distances: List of all value1s

Output to MapReduce Env. : <key2, value2>
key2: the record id of the test instance
value2: the decided class id for test instance i

1: Sort_ascending(distances)
2: new LinkedList K_distances
3: new LinkedList Classes
4: new LinkedList Z_instances
4: for i=1 to K
5:     K_distances.add(distances.get(i))
6: end for
7: for all dist $\in$ K_distances
8:     if dist.getclass() $\notin$classes
9:         classes.add(dist.getclass())
10:    end if
11: end for
12: for all class_id $\in$classes
13:    Z_instances.clear()
14:    for i=1 to Z
15:        if distances.get(i).getclass() = class_id
16:            Z_distances.add(distances.get(i))
17:        end if
18:    end for
19:    $\mu = \frac{1}{Z}\sum_{w=1}^{Z}(Z\_instances.get(w))$
20:    if DistanceFunction(μ,key1) < min
21:        min = DistanceFunction(μ,key1)
22:        decided_class_id = class_id
23:    end if
24: end for
25: key2 = key1
26: value2 = decided_class_id
27: Context.write(key2, value2)
28: return
```

2) The Reducer Function:

The reducer function contains the classification decision phase for the test instances, where the main contribution of the proposed Z-KNN algorithm can be seen.

The input of the reducer function is a list of all of the <key1, value1> pairs emitted by the mapper function of the MapReduce framework. It is worth to mention again that a single reducer receives the list of pairs belonging to a common key value. In other words, a single reducer receives the distances of a single test instance to all of the training instances calculated by the mappers.

Upon receiving the input, the reducer function finds the K closest neighbors from all of the training instances by examining the minimum K distances among all the values in the list. Next, the Z-KNN reducer detects to which classes these K neighbors belong (e.g. class A, class B and class C).

The main contribution in Z-KNN classifier depends on correctly representing the detected classes A, B and C rather than relying only on the distances to the individual training data instances. The main motivation of this strategy is fueled by the fact that, when the size of the data is big and especially when the data is represented by multiple number of classes and large number of features, some data may have similar proximity to different classes instances. In such cases, the minimum distance from the individual instances may not correctly mean that the test instance will belong to that same class. Please consider the following example as a sample case:

Let assume that K is 5 and the following values are emitted to the reducer for test instance 9. Also, for the sake of example, let us assume that the test instance 9 should be classified to class B:

<ts_9, <tr_3, 0.11, A>
<ts_9, <tr_{11}, 0.14, B>
<ts_9, <tr_{27}, 0.15, B>
<ts_9, <tr_{23}, 0.12, C>
<ts_9, <tr42, 0.12, A>

In this case, because of majority voting, the classical K-NN algorithm will conclude that the test instance 9 belongs to class A. The majority voting strategy of the classical K-NN will end up at this decision since class A and class B has equal number of instances among the K nearest neighbors, and the class A contains an instance, which has the closest proximity to the testing instance.

Nevertheless, the same class A has an instance, which is further away from the class B instances to the test instance 9. To give equal chances to classes in the classification decision, Z-KNN proposes to represent the classes A, B and C among the K nearest neighbors by centroids and base the classification decision on the distance of the test instance to the centroids of the classes rather than relying on the individual data members' proximities.

To decrease the computation overhead of the proposed proximity to the centroid representation strategy, in the Z-KNN algorithm, a parameter Z is introduced.

The parameter Z in the Z-KNN classifier is the number of instances from each of the classes A, B and C that have the closest distances to the test instance.

As it was explained earlier in the Mapper function of the algorithm, for each test instance, the distances to every training instance is calculated and emitted to the Reducer function.

Also it is worth to mention that, benefiting from the MapReduce Framework's shuffling/sorting functionality, the coded Reducer Function and the Value class, which defines the value objects in the <Key, Value> pairs, are coded to have an sorted list of the values according to the ascending order of the distances.

That is to say, each reducer receives a list that is already sorted so that the reducers can directly take the first Z number of elements from each class. Hence, with no extra cost, the Reducer is able to use the already available proximity information.

Repeating the centroid calculation for each class, the Z-KNN Reducer calculates the class centroids using the first Z elements, in other words closest Z training instances to the test instance to be classified, the reducer ends up with a number of centroids as many as the number of classes found among the K nearest neighbors.

The Z parameter contribution simply proposes that, instead of using the complete class population to calculate a class center, using only Z number of instances of a class, the reducer calculates the centroid for that class with a much lower computation cost and still maintaining a strong class representation compared to relying on individual instance proximities.

Then, the classification decision will be given by the Z-KNN reducer function, according to which centroid the test instance have the minimum distance.

In this way, the outliers in the class data will have less significance and the decision will be based on a stronger representation of the classes.

The complete Z-KNN reducer function's algorithm can be seen in Algorithm 2.

III. The Experimental Setup and The Results

A. *The Experimental Setup*

The MapReduce functions of the Z-KNN are coded in Sun JAVA JDK 1.8 [19]. Z-KNN classification experiments are conducted on a small cluster of HP Workstations installed with Ubuntu Linux 16.04 and Apache Hadoop 2.7.4.

To validate the classification scheme, for each dataset used in the experiments, 10-fold cross validation is used, where each test is repeated 10 times and the averages of the 10 tests are considered so that the reliable results can be achieved.

In the experiments, real datasets downloaded from UCI Machine Learning Repository [20] are used. The 5 real datasets that are used in the experiments are summarized in Table I.

TABLE I
THE REAL DATASETS USED IN THE EXPERIMENTS

Dataset	Instances	Features	Classes
ionosphere	351	34	2
wdbc	569	32	2
wine	178	13	3
seeds	210	7	3
satimage	6435	36	7
pendigits	10992	16	10

B. *Datasets Used In The Experiments*

1) Ionosphere: Ionosphere data set is the data coming from the classification of radar returns from the ionosphere. The dataset contains 351 instances belonging to 2 classes. Each instance contains values belonging to 34 features. This dataset is also used in [14].

2) WDBC: The Wisconsin Diagnostic Breast Cancer WDBC) was first used in [21]. The dataset contains 569 instances belonging to 2 classes. Each instance contains values belonging to 32 features. WDBC dataset is also used in [14].

	Z			
K	**3**	**5**	**7**	**9**
5	**0.9507**	0.9443	0.9474	0.9443
7	0.9474	0.9443	0.9474	0.9443
9	0.9474	0.9443	0.9474	0.9443
11	0.9474	0.9478	0.9474	0.9443

(a) Wdbc Dataset

	Z			
K	**3**	**5**	**7**	**9**
5	0.9524	**0.9714**	0.9714	0.9714
7	0.9524	0.9714	0.9714	0.9714
9	0.9524	0.9714	0.9714	0.9714
11	0.9524	0.9714	0.9714	0.9714

(b) Seeds Dataset

	Z			
K	**3**	**5**	**7**	**9**
5	0.9199	0.9141	0.9147	0.9147
7	0.9196	0.9144	**0.9203**	0.9144
9	0.9196	0.9144	0.9203	0.9144
11	0.9196	0.9144	0.9203	0.9144

(c) Ionosphere Dataset

	Z			
K	**3**	**5**	**7**	**9**
5	0.9797	0.9803	0.9803	0.9803
7	0.9803	0.9808	0.9808	0.9808
9	0.9803	0.9808	0.9811	0.9811
11	0.9806	0.9811	**0.9814**	0.9814

(d) Pendigits Dataset

	Z			
K	**3**	**5**	**7**	**9**
5	0.9220	0.9245	0.9245	0.9220
7	0.9235	0.9265	0.9270	0.9245
9	0.9250	0.9280	**0.9285**	0.9255
11	0.9240	0.9275	0.9285	0.9255

(e) Satimage Dataset

	Z			
K	**3**	**5**	**7**	**9**
5	0.7993	0.7974	**0.8320**	0.8203
7	0.7917	0.7974	0.8209	0.8092
9	0.7882	0.7973	0.8209	0.8092
11	0.7271	0.7379	0.7611	0.7608

(f) Wine Dataset

Figure 2. The classification Accuracy Results for (a) Wdbc, (b) Seeds, (c) Ionosphere, (d) Pendigits , (e) Satimage and (f) Wine datasets

3) Wine: Wine dataset contains data from chemical analysis to determine the origin of wines. The dataset is composed of 178 instances in 3 classes containing 13 features. Wine dataset is also used in the experiments of [14].

4) Seeds: The seeds dataset contains the measurements of geometrical properties of kernels belonging to three different varieties of wheat. The dataset contains 210 instances in 3 classes. Each instance is defined by the values of 7 features. Seeds data set is first used in [22] and also investigated in [14].

5) Satimage: The Satimage dataset was generated from Landsat Multi-Spectral Scanner image data. The dataset contains 6435 instances belonging to 7 classes. Each instance contains the data of 36 features. Satimage dataset is also used by [12]–[14].

6) Pendigits: Pen-Based Recognition of Handwritten Digits Data Set (pendigits) is a digit database of 250 samples from 44 writers [23]. This dataset contains 10992 instances belonging to 10 classes. Each instance contains the data of 16 features. Pendigits is also used by [12]–[14].

C. The Results

In this section, the results acquired after extensive experiments are presented. The performance of the Z-KNN algorithm is measured in terms of classification accuracy, which represents the ratio of the number of correct classifications to the number of all classifications. The classification accuracy results are given in Fig. 2.

As the overall classification accuracy performance, it can be seen in Fig. 2 that the Z-KNN managed to correctly detect the class of more than 92% of the tested data in all of the data sets.

Also, looking at the accuracy performance of the Z-KNN it can be seen that, for the majority of the datasets, the Z-KNN algorithm manages to detect the correct class of the test instances with K values 5 or 7, without needing to analyze more number of nearest neighbors and hence attaining a reasonable computation cost.

As for the Z parameter, it can be observed that the Z-KNN algorithm manages to achieve a high classification accuracy with 5 to 7 nearest neighbors in the class representation, which also shows that the addition of the Z parameter does not increase the computation cost significantly.

Especially on Pendigits and Satimage datasets, which contain higher number of instances, features and classes compared to other datasets, it is worth to mention that by attaining Z values smaller or equal to 7, Z-KNN shows realistic applicability also to real big data applications.

The accuracy performance of the proposed Z-KNN algorithm and its comparison against the classical K-NN's accuracy is given in Table II.

TABLE II
CLASSICAL K-NN VS Z-KNN CLASSIFICATION ACCURACIES

Dataset	**Classical K-NN**	**Z-KNN**
Wine	0.8295	0.8320
Wdbc	0.6548	0.9507
Seeds	0.8424	0.9714
Ionosphere	0.6286	0.9203
Pendigits	0.978	0.9814
Satimage	0.9065	0.9285

As it can be seen in Table II, The Z-KNN significantly improves the accuracy performance of the Classical K-NN algorithm in all data sets. In addition, the performance of the Z-KNN algorithm is compared against two algorithms recently proposed in [13-14]. The comparative results are presented in Table III.

TABLE III
PERFORMANCE COMPARISONS

Dataset	**LC-KNN [13]**	**SR-KNN [14]**	**Z-KNN**
Wine	-	0.9707	0.8320
Wdbc	-	0.965	0.9507
Seeds	-	0.9019	0.9714
Ionosphere	-	0.8971	0.9203
Pendigits	0.9721	0.9452	0.9814
Satimage	0.8883	0.8806	0.9285

As it can be seen in the performance comparisons, Z-KNN performs better almost in all of the datasets compared to other KNN based proposals, which is demonstrating that the proposed Z instance representation significantly improves the accuracy performance of the classical K-NN and some of its variations.

The only dataset where the proposed Z-KNN algorithm is not performing better than the competitors is the Wine dataset. From the results, which were observed in [17], it can be deduced that the low performance of the Z-KNN can be explained by the data distribution features of the Wine dataset that can be remedied by introducing the variance effect contribution to the similarity analysis.

IV. The Conclusion and the Future Works

In this paper a new K-NN based algorithm, named Z-KNN is presented and the performance results are presented after extensive experiments conducted on Hadoop MapReduce environment.

The performance results show that, the main contribution, which proposes to use centroid representation of the data classes instead of relying on individual instance distances, proves to improve the classification accuracy over classical K-NN algorithm.

In the experiments, it was observed that the proposed Z-KNN algorithm proves to be a strong competitor with its high classification accuracy achieved for several different real datasets.

As the future works, it is planned to introduce the effect of the variance to the distance calculation, from the study proposed in [17]. It is expected that, especially the weakness that can be seen in the wine data set can be significantly improved when variance effect is introduced to the distance calculations.

In addition, instead of the classical distance measure, a new similarity measure will be introduced to the Z-KNN algorithm so that the algorithm becomes applicable to any kind of quantitative/categorical features containing datasets.

As an immediate improvement, it is planned to improve the Z instances usage during centroid calculations by introducing a weighted contribution of the Z instances to the centroids. With this improvement, it is expected that, especially if the weights of the Z instances can be set or calculated effectively, the overall classification accuracy of the Z-KNN algorithm can be improved significantly.

Lastly, after the planned future works, the Z-KNN algorithm will be applied to other datasets containing number of instances in the measure of 10^6 and above to further prove the algorithms applicability to Big Data applications.

References

[1] Klaus Schwab, "The Fourth Industrial Revolution", Crown Business, 2017.

[2] D. Singh and .K. Reddy, "A survey on platforms for big data analytics", Journal of Big Data vol. 1, no. 8, 2014.

[3] P. Tan, M. Steinbach and V. Kumar, "Introduction to Data Mining", 1st ed., Reading, MA: Addison-Wesley, 2005.

[4] J. Dean, S. Ghemawat , "MapReduce: A Flexible Data Processing Tool", Communications of the ACM, vol. 53 no. 1, pp.72-77, 2010.

[5] X. Wu et. Al., "Top 10 algorithms in data mining", Knowledge and Information Systems,vol. 14, no. 1, pp 137, 2008.

[6] Fahad et. AL., "A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis", IEEE Trans.on Emerging Topics in Computing, vol. 2, no.3, pp. 267-279, 2014.

[7] S. Zhang, M. Zong and D. Cheng, "Learning k for KNN Classification", ACM Transactions on Intelligent Systems and Technology, vol. 8, no. 3, pp. 43:1-19, 2017.

[8] K. Niu, F. Zhao and S. Zhang, "A Fast Classification Algorithm for Big Data Based on KNN", Journal of Applied Sciences, vol. 13,no. 12, pp. 2208-2212, 2013.

[9] Bifet, J. Read, B. Pfahringer and G. Holmes, "Efficient Data Stream Classification via Probabilistic Adaptive Windows", in Proc. 28th Annual ACM Symposium on Applied Computing, 2013, pp. 801-806.

[10] S. S. Labib, "A Comparative Study to Classify Big Data Using fuzzy Techniques", in Proc. 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA), 2016.

[11] M. El Bakry, S. Safwat and O. Hegazy, "A Mapreduce Fuzzy technique of Big Data Classification, in Proc. SAI Computing Conference 2016, pp. 118-128.

[12] B. Quost and T. Denoeux, "Clustering and Classification of fuzzy data using the fuzzy EM algorithm", Fuzzy Sets and Systems, vol. 286, pp. 134-156, 2016.

[13] Z. Deng, X. Zhu, D. Cheng, M. Zong and S. Zhang, "Efficient kNN classification algorithm for big data", Neurocomputing, vol.195, pp. 143-148, 2016.

[14] S. Zhang, D. Cheng, M. Zong and L. Gao, "Self representation nearest neighbour search for classification", Neurocomputing, vol.195, pp. 137-142, 2016

[15] G. Song, J. Rochas, L. El Beze, F. Huet and F. Magoules, "K Nearest Neighbour Joins for Big Data on MapReduce:A Theoretical and Experimental Analysis", IEEE Trans. on Knowledge and Data Engineering, vol. 28, no. 9, pp. 2376-2392, 2016.

[16] J. Maillo, S. Ramirez, I. Triguero and F. Herrera, "kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbours classifier for big data", Knowledge-Based Systems, vol. 117, pp. 3-15, 2017.

[17] T.Tulgar, A.haydar and İ.Erşan, "Data Distribution Aware Classification Algorithm based on K-Means", International Journal of Advanced Computer Science and Applications, Article in Press, 2017.

[18] T. White, "Hadoop: A Definitive Guide", 4th ed., O'Reilly, 2015.

[19] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, (2017,AUG 01). The Java Language Specification-Java SE 8 Edition Online. Available: https://docs.oracle.com/javase/specs/jls/se8/html/index.html

[20] UCI Center for Machine Learning and Intelligent Systems, (2017, AUG 01). UC Irvine Machine Learning RepositoryOnline.Available: https://archive.ics.uci.edu/ml/

[21] O.L. Mangasarian, W.N. Street and W.H. Wolberg, “Breast cancer diagnosis and prognosis via linear programming”, Operations Research, vol. 43, no. 4, pp. 570-577, July-August 1995.

[22] M. Charytanowicz, J. Niewczas, P. Kulczycki, P.A. Kowalski, S. Lukasik, S. Zak, “A Complete Gradient Clustering Algorithm for Features Analysis of X-ray Images”, Information Technologies in Biomedicine, Springer-Verlag, Berlin-Heidelberg, pp. 15-24, 2010.

[23] F. Alimoglu, E. Alpaydin, “Methods of Combining Multiple Classifiers Based on Different Representations for Pen-based Handwriting Recognition”, in Proc. Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96), June 1996.

BIOGRAPHIES

Tamer Tulgar received his B.Sc. degree in Computer Engineering and the M.Sc. degree in Computer Engineering from the Eastern Mediterranean University in 1999 and 2001, respectively. He has a Ph.D. degree in Computer Engineering from the Eastern Mediterranean University (2008) in the area of Cellular Wireless Communication. Tamer Tulgar worked as a Research Assistant in Eastern Mediterranean University during his Graduate Studies. Upon receiving his Ph.D. degree, he joined the the Girne Ameircan University, Department of Computer Engineering staff, where he currently works as Associate Professor. His research interests include Wireless Communication, Computer Networks, Data Mining, Machine Learning and currently Big Data Analysis.

Ali Haydar received the B.Sc. degree in Electrical and Electronics Engineering and the M.Sc. degree in Electrical and Electronics Engineering from the Middle East Technical University in 1991 and 1994, respectively. He has a Ph.D. degree in Electrical and Electronics Engineering from the Eastern Mediterranen University (1999). He has worked in the research labs of TÜBİTAK during his graduate studies in a project in the field of speech recognition. His research interests include Artificial Neural Networks, Speech Recognition, Optimization, Fuzzy Logic and Data Analysis.

İbrahim Erşan, was born in Nicosia, North Cyprus, in 1974. He received the B.Sc. and M.Sc. degrees in electrical and electronic engineering from Eastern Mediterranean University (EMU), Famagusta, in 2000 and the Ph.D. degree in computer engineering from Girne American University (GAU), Kyrenia, in 2012. In 1997, he was an assistant lecturer in EMU, Electrical and Electronics Engineering Department. From 1998 to 2001, he was a research assistant in EMU, Information Technologies Research and Development laboratory. From 2001 to 2005, he was working in construction industry as electrical engineer and project manager. Since 2006, he is working in GAU and since 2017, he is an Associated Professor in Computer Engineering Department. He is currently the head of Computer Engineering Department. He is the author of more than 15 articles. His research interests include decision support systems, machine learning, neural networks and big data.