Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 7, July 2018, pp. 1391-1400
DOI: 10.23940/ijpe.18.07.p2.13911400

Mixed Weighted KNN for Imbalanced Datasets

Qimin Cao[a], Lei La[b,*], Hongxia Liu[a], and Si Han[a]

[a]*Library, China University of Political Science and Law, Beijing, 100088, China*
[b]*School of Information Technology and Management, University of International Business and Economic, Beijing, 100029, China*

Abstract

It is well known that imbalanced datasets are a common phenomenon and will reduce the accuracy of classification. For solving the class imbalance problem, this paper proposed the mixed weighted KNN algorithm. According to the imbalance between the classes, this algorithm assigns each sample of datasets an inverse proportion weight, and then it combines with the distance weight, making the weight of the training sample close to the test sample greater. In order to improve the operating efficiency and make it easy to handle massive datasets, we implemented the parallelism of MW-KNN based on the Hadoop framework. Experimental results show that the proposed algorithm is simple and effective.

Keywords: accuracy; imbalanced datasets; inverse proportion weight; MW-KNN; text classification; time consumption

(Submitted on April 13, 2018; Revised on May 25, 2018; Accepted on June 25, 2018)

1. Introduction

With the development of Web technology, social networking sites have gradually become the main data source of the Internet, such as micro-blogs (Twitter, Sina micro-blog), SNS (Facebook, LinkedIn), and review sites (Yelp, dianping.com). These new Web applications extended the research field of text classification. Meanwhile, the problem of imbalanced datasets in the field of machine learning and data mining has been attracting more and more attention, and it has become a critical problem [11].

Imbalanced datasets are datasets in which the number of samples of certain categories is significantly less/more than other categories, where the categories containing fewer samples are called the minority class and the categories containing more samples are called the majority class. For the problem of imbalanced datasets, the traditional classification algorithms, especially instance-based classification algorithms such as KNN, tend to the majority class. The reasons for this situation are mainly as follows: the learning goal of the traditional algorithm is to minimize the overall error rate; however, the contribution of the minority class is relatively small; they assume the contribution of the minority and majority class is balanced, and they argue that the error from the minority and majority class has the same cost [3]. In practical applications, such as error detection in software testing, identification of oil spill events using satellite images, labeling of telephone fraud, diagnosis of serious diseases in the process of medical treatment, and so on [14], the classification accuracy of the minority class is more important than that of the majority class. Therefore, improving the overall performance of classifiers has gained attention in related research fields [19].

For the classification problem of imbalanced datasets, the performance of the K-Nearest Neighbor algorithm (KNN) is better than that of the Bayesian algorithm and Support Vector Machine algorithm (SVM), but its performance is still poor [5]. Therefore, in recent literature, there have been a lot of improved KNN algorithms for solving class imbalance problems. In [15], Mahanalobis distance matrices for KNN classification uses semidefinite programming, which is called the large margin nearest neighbor (LMNN) classifier. In [9], DNet is proposed, which used a non-linear feature mapping method pre-trained with Restricted Boltzmann Machines to achieve the goal of a large margin KNN classifier. In [18], a new method of

* Corresponding author.
E-mail address: lalei1984@aliyun.com

WDKNN is introduced, which discovered optimal weights for each instance in the training phase taken into account during test phases. This method has been demonstrated to be superior to the traditional KNN algorithm. In [17], the Kmeans-KNN algorithm is proposed to improve the KNN algorithm. In [8], the proposed method uses the probability of attribute values given class labels to weight samples in KNN. In [12], Neighbor-Weighted K-Nearest Neighbor (NWKNN) is proposed for imbalanced text classification problems. In [2], the proposed method takes into account the class distribution in a wider region around the query instance. In [10], a hybrid weighted nearest neighbor approach (HY-KNN) is proposed to mine imbalanced data. In [6], the proposed algorithm uses prototype-weighting to improve the nearest neighbor classifier. Although these methods have improved the classification accuracy in some ways, they all involve combining other algorithms, so their computational complexity is very high. Thus, they are unsuitable to handle massive data.

This paper presents the mixed weighted KNN algorithm (MW-KNN) based on the combination of weight factor, which relates to the proportion of training samples that belong to different classes and distance weight factor. Firstly, because of the imbalance between the classes, this algorithm assigns each sample of datasets an inverse proportion weight, so it is no longer affected by the class imbalance problem. Secondly, it combines with distance weight, so that the training sample that is closer to the test sample is greater in weight. Lastly, the test sample is classified by the improved decision rule. In the experimental section, in order to improve the operating efficiency and easily handle massive datasets, we implemented the parallelism form of the MW-KNN algorithm based on the Hadoop framework. Experimental results showed that this proposed algorithm is simple, effective, and can solve the class imbalance problem. In addition, the Hadoop cluster approach can effectively improve the efficiency of the algorithm.

The rest of the paper is organized as follows. We briefly review the traditional KNN algorithm in section 2. Then, we present an overview of the MW-KNN algorithm in section 3. The experimental results and analysis are presented in section 4. Section 5 concludes this paper.

2. KNN Algorithm

The KNN algorithm is an instance-based learning method [16]. The basic idea of KNN classification method is: given a training set D and a test object x, firstly, calculate the distance between each of the training samples and the test object x, then find the k nearest neighbors of the test sample, and lastly assign the dominant category of the k nearest neighbor samples to the test object x and obtain the category of the test object x. The pseudocode of the KNN algorithm is shown in algorithm in Figure 1:

Algorithm: The Traditional KNN Algorithm

Input: the training set D, test object x, category label set C
Output: the category c_x of test object x, c_x belongs to the C

begin
  for each y belongs to D do
    calculate the distance $D(y, x)$ between y and x
  end for
  select the subset N from the data set D,
  the N contains k training samples which are the k nearest neighbors of the test sample x
  calculate the category of x:
  $c_x = \arg\max_{c \in C} \sum_{y \in N} I\left(c = class\left(y\right)\right)$
end

Figure 1. Pseudocode of KNN algorithm

In the above algorithm, $I(.)$is an indicator function. When its condition is true, it returns 1, otherwise it returns 0.

It is easy to understand and implement the KNN classification algorithm, and it has good performance in many cases. In [1], it shows that the error rate of the KNN algorithm will asymptotically converge to the error rate of the Bayesian algorithm, so it can be used as an approximation of the Bayesian. Because the KNN is very simple, it is easy to be improved to deal with more complex classification problems. For example, KNN is especially suitable for multi-label categorization problems. However, KNN also has limitations, mainly the classification problem of imbalanced datasets. This means that when the number of training samples that belong to different classes are greatly different from each other, the classification performance of the KNN algorithm will become less accurate. In order to solve this problem, this paper improves the KNN algorithm to avoid the influence of the imbalanced datasets.

3. Mixed Weighted KNN Algorithm

Because it is simple and easy to be implemented, the KNN algorithm has been widely used in the field of text categorization. However, when the traditional KNN algorithm encounters imbalanced datasets, the classification results tend to the majority class, because it did not consider the imbalance between classes [7]. Therefore, this paper proposes the mixed weighted KNN algorithm (MW-KNN). The basic idea is: firstly, according to the distribution of the datasets, it assigns every sample an inverse proportion weight which is normalized, then it assigns the *k* nearest neighbors of the test sample the distance weight, and lastly, it classifies the test sample by the improved decision rule.

3.1. Sample Weight Allocation

Because of imbalanced datasets, the neighbours of the test sample are always the ones in the majority class. If every sample in the datasets is assigned an inverse proportion weight, they are no longer affected by class imbalance, so the semantic distance between samples decides the number of samples in different classes to be classified into the category of the test sample. An example of the proportion reciprocal weight is as follows.

If the data set has 3 categories, they are class ***A***, class ***B***, and class ***C***. There are 10 samples in class ***A***, 15 samples in class ***B***, and 25 samples in class ***C***. Without considering the semantics, the probability of all kinds of samples being classified into the category of the test sample are shown in Equation (1), Equation (2) and Equation (3):

$$P(A) = |A| / (|A| + |B| + |C|) = 10/(10 + 15 + 25) = 10/50 = 0.2 \tag{1}$$

$$P(B) = |B| / (|A| + |B| + |C|) = 15/50 = 0.3 \tag{2}$$

$$P(C) = |C| / (|A| + |B| + |C|) = 25/50 = 0.5 \tag{3}$$

Where $|A|, |B|, |C|$ respectively represent the numbers of samples in classes *A*, *B*, *C*.

In order to let the initial probability of each category be equal, they can be handled as follows:

$$W(A)P(A) = W(B)P(B) = W(C)P(C) \tag{4}$$

Then, we can obtain $W(A):W(B):W(C)$=15:10:6 of Equation (4), if it is normalized such that:

$$W(A) + W(B) + W(C) = 1 \tag{5}$$

Then, in Equation (5), $W(A)$=15/31, $W(B)$=10/31, $W(C)$=6/31, $W(A)$, and $W(B)$ and $W(C)$ are the proportion reciprocal weight.

If we do not use the proportion reciprocal weight, it is equivalent to no extra data (if the training data set has two categories *A* and *B*, there are 10000 samples in class *A* and 100 samples in class *B*, and then we select 100 samples from class *A* as the training corpus), so it does not make sense. If we really select 100 samples from class *A*, it will lead to a new problem: what is the selection criterion? Whether we choose them randomly or choose the ones which have semantic representation, whether it is manual work or machine selection, they will all increase the labour cost or computational overhead. In order to avoid this new problem, in this paper, we proposed the proportion reciprocal weight: it has low cost and also uses all of the samples which are equivalent to the semantic average.

The traditional KNN algorithm assigns the *k* nearest neighbors the same weight when it judges the category of the test sample. In fact, the contributions of different samples are not the same. The training sample that is closer to the test sample should have greater weight. It is the same for the imbalanced datasets. Although most of the samples classified in the category of the test sample belong to the majority class, sometimes the samples of the minority class could be very close to the test sample and their weights also should be large. As shown in Figure 2, in the circle, although the triangle to square ratio is 3:5, the triangles are closer to the test sample circle. Its weight should be large, so it is also necessary to combine the distance weight.

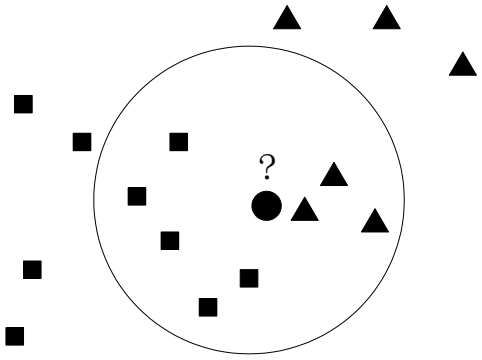

Figure 2. Example of classification for imbalance data

3.2. Mixed Weighted KNN Classification Algorithm

In this section, a mixed weighted KNN algorithm (MW-KNN) is proposed. D denotes the training corpus, m is the number of the classes, C_i ($i=1,\cdots,m$) are class labels, and there are n_i samples in class C_i ($i=1,\cdots,m$). Let x denote the test object and z_j denote the training sample, $j=1,\cdots,N$, $N=n_1+n_2+\cdots+n_m$. The algorithmic process of the proposed algorithm is as follows:

- Without considering the semantics, calculate the initial probability of each category:

$$P(C_i)=\frac{|C_i|}{\sum_{i=1}^{m}|C_i|}, i=1,\cdots,m \tag{6}$$

Where $|C_i|$ is the number of samples in the class $|C_i|$ in Equation (6).

- By using Equation (7) and Equation (8)

$$W(C_1)P(C_1)=W(C_2)P(C_2)=\cdots=W(C_m)P(C_m) \tag{7}$$

$$W(C_1)+W(C_2)+\cdots+W(C_m)=1 \tag{8}$$

Calculate the weight $W(C_i)$of the samples of each category, $i=1,\cdots,m$.

- Calculate the distance between every sample z_j and the test object x.
- Select the subset N from the data set D, the N contains k training samples which are the k nearest neighbors of the test sample x, denoted as y_t, t=1,…,k.
- Calculate the distance weight S_t of the samples in subset N, t=1,…,k. This paper takes the cosine distance between y_t and x as the distance weight S_t as Equation (9).

$$S_t=\frac{y_t\cdot x}{|y_t||x|} \tag{9}$$

- Calculate the class label of the test object x by the following decision rule of Equation (10)

$$C_x=\arg\max_{C_i}\sum_{t=1}^{k}W(C_i)S_tI(y_t\in C_i) \tag{10}$$

The workflow of this algorithm is in Figure 3.

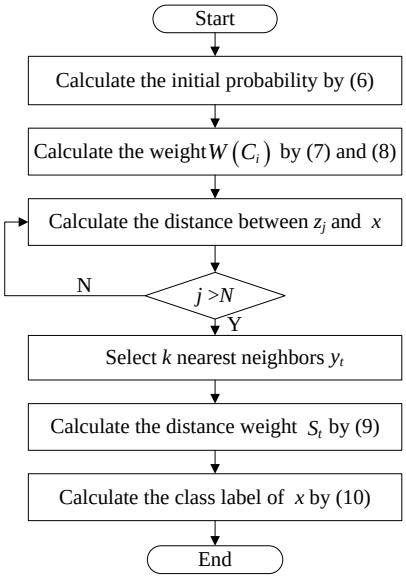

Figure 3. The logical workflow of MW-KNN algorithm

4. Experiments and Analysis

4.1. MapReduce Paralleling KNN Algorithm

In order to improve the KNN algorithm's ability to process massive datasets, in this section, the parallelism of KNN based on the MapReduce programming model [20] is used. Its basic idea is to decompose the massive datasets into many small splits, the Map function performs the computing task in each split and generates intermediate results, and then the intermediate results are used as the input of Reduce function and perform the computing tasks to obtain the final result. In the MapReduce programming model, the input and output of the Map and Reduce function are in the form of key-value pair <key, value>. The framework of MapReduce is shown in Figure 4.

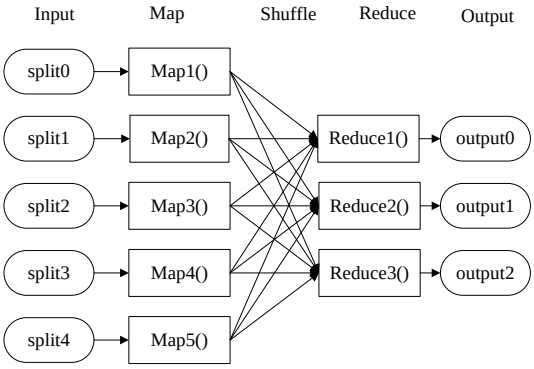

Figure 4. The framework of MapReduce

The parallelism of KNN based on MapReduce is: the calculation of every test sample in the serial algorithm can start over once the computing process of MapReduce programming model completes the calculation of the distance between the test sample and the training data, and the sorting operation [4]. In order to fit the MapReduce programming model, the data to be processed and all the results should be stored in rows. This should be given in the driver file so that the data to be processed can be segmented by row and data parts have no correlation between each other [21]. By default, the segmentation process is completed by the MapReduce runtime environment, codes do not need to be written.

4.1.1. Map Function

The task of the Map function is to read the training sample set and format the form of key-value pair <key, value>. The Split function first reads the sample by row and converts it to a file with a specific format, then it traverses each test sample to calculate its similarity with every training sample, and finally, it stores the results in the context set [13]. The key-value pair

of input data is in the form of <row number, sample>, the key-value pair of output data is in the form of <row number, vector set<class label of training sample, similarity value>>, and the similarity value is measured by the standard Euclidean distance. The corresponding pseudo code is shown in Algorithm 2.

Algorithm: The Map Function

Input: <row number, sample>
Output: < row number, vector set> Context

```
1   begin
2     for (i=0;i<=n; i++) //n: the number of training data
3       C=FindCatalog(i)
4       for (y=0; y<=m; y++) // m: number of test data
5         d= EuclideanDistance(y, i)
6         Context.write (line number, vector set (c, d));
7       end for
8     end for
9   End
```

Figure 5. The MAP function

4.1.2. Reduce Function

The Reduce function is used for the protocol operations, and its main task is to take out the k nearest neighbors and count the class with the highest score to assign the test sample. The background system needs to perform the shuffle operation so that the samples with the same key value in different nodes can be sent to the same Reduce function. Then, it performs the sorting, classification, and output operations. The key-value pair of input data is in the form of <row number, vector set<class label of training sample, similarity value >>, and the key-value pair of output data is in the form of <row number, category ID of test sample>. The corresponding pseudo code is shown in Algorithm 3.

Algorithm: The Reduce Function

Input: <row number, vector set>
Output: <row number, category ID> Context

```
1  begin
2    for (j=0; j<=m; j++)
3      ArrayList (vector (row number, vector set (c, d)))
4      Sort (ArrayList)
5      New ArrayList Result
6      for (t=0; t<=k; t++)
7        result.add (row number, ArrayList. get(t))
8      end for
9      context.write (row number, proposed kNN(result))
10   end for
11 end
```

Figure 6. The reduce function

4.2. Experimental Environment and Data Set

The cluster scale of the experiment is 8 computers, and the main configurations are as follows: the CPU is Intel Core 2 Duo 2.93GHZ, the memory is 2G, the hard disk capacity is 500G, and the version of Hadoop is 1.1.2. The building of the cluster refers to the recommended method in the Hadoop official website.

In the experiment, we use texts to download from standard corpora. For evaluating its performance in the different corpus, classical Reuters 21578 corpus and Sogou text classification corpus are used.

4.3. Experiment for Classification Accuracy Analysis

In order to verify the effectiveness of the MW-KNN algorithm for the problem of imbalanced datasets, experiments are carried out to compare with the traditional KNN algorithm as well as the LMNN, WDKNN, and HY-KNN algorithms. In

this group of experiments, we selected 1,000 Politics texts, 4,000 Society texts, 9,000 Economics texts, 12,000 Science texts, 20,000 Military texts, and 25,000 Culture texts. The comparative results are shown in Table 1 and Table 2.

Table 1. The accuracy of different algorithm for Chinese documents

Algorithm / category	KNN	MW-KNN	LMNN	WDKNN	HY-KNN
Society	0.411	0.871	0.782	0.824	0.853
Economics	0.609	0.869	0.779	0.830	0.849
Science	0.710	0.872	0.783	0.827	0.851
Politics	0.716	0.875	0.779	0.826	0.854
Military	0.709	0.868	0.781	0.831	0.854
Culture	0.715	0.877	0.782	0.820	0.850

Table 2. The accuracy of different algorithm for English documents

Algorithm / category	KNN	MW-KNN	LMNN	WDKNN	HY-KNN
Society	0.412	0.872	0.783	0.823	0.854
Economics	0.610	0.870	0.780	0.829	0.850
Science	0.708	0.869	0.782	0.830	0.848
Politics	0.714	0.876	0.780	0.828	0.855
Military	0.709	0.871	0.779	0.831	0.853
Culture	0.713	0.876	0.782	0.825	0.850

The experimental results show that, in dealing with the classification problem of imbalanced datasets, the performance of the MW-KNN algorithm proposed in this paper is far exceeding that of the traditional KNN algorithm and LMNN algorithm. In addition, the novel algorithm has better performance than the WDKNN and HY-KNN algorithms. That is because we used the proportion reciprocal weight so that the training samples are no longer affected by the class imbalance problem and the training sample that is close to the test sample has greater weight. Therefore, the performance of the proposed method is better than the others.

4.4. Efficiency Test and Analysis

In order to measure the time consumption of the algorithm proposed in this paper, under the stand-alone state, an experiment is carried out to compare with other algorithms. In this experiment, from each corpus, we selected 1,000 Politics texts, 4,000 Society texts, 9,000 Economics texts, 16,000 Science texts, 20,000 Military texts, and 50,000 Culture texts. The results are shown in Figure 7 and Figure 8.

As shown in Figure 7 and Figure 8, the traditional KNN algorithm has the lowest cost and the WDKNN algorithm needs the longest time. The time consumption of the MW-KNN algorithm is lower than that of the LMNN and HY-KNN algorithms. According to the experimental results, the proposed method has better efficiency.

In order to facilitate the processing of massive datasets, we implemented the parallelism of the MW-KNN algorithm based on the Hadoop framework. The second group of experiments under the Hadoop clusters state is carried out to compare with other algorithms. In this experiment, we used the same dataset as the first group of experiments. The comparative results are shown in Figure 9 and Figure 10.

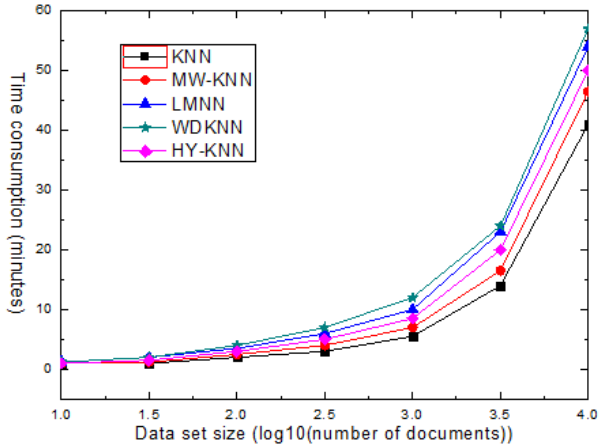

Figure 7. Time consumption comparison under the stand-alone state for Chinese documents

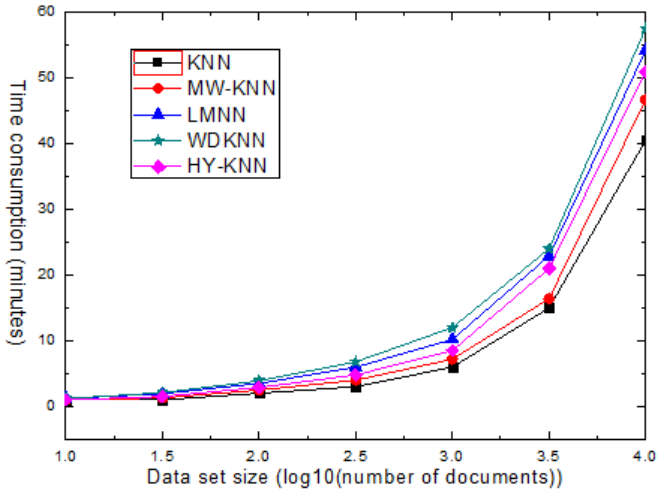

Figure 8. Time consumption comparison under the stand-alone state for English documents

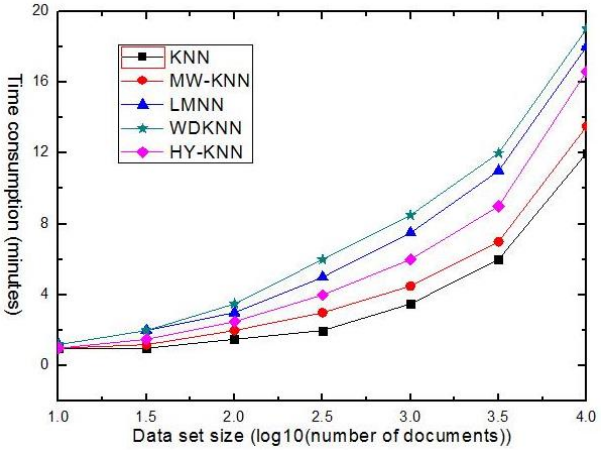

Figure 9. Time consumption comparison under the Hadoop clusters state for Chinese documents

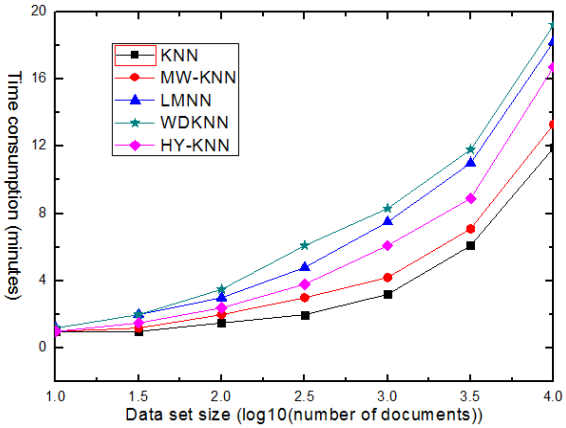

Figure 10. Time consumption comparison under the Hadoop clusters state for English documents

By comparing Figure 7 and Figure 9 or Figure 8 and Figure 10, we found that in Figure 9 or Figure 10, the difference between the various algorithms is more obvious. After the Hadoop framework based parallelism, the efficiency of every algorithm has improved. The traditional KNN algorithm improved the most, the WDKNN algorithm improved the least, and the MW-KNN algorithm improved much more than the LMNN algorithm and HY-KNN algorithm. This shows that the parallelism of the MW-KNN algorithm based on the Hadoop framework can effectively improve its operational efficiency.

In a word, the MW-KNN algorithm is an effective algorithm for addressing the imbalanced datasets classification problem.

5. Conclusions

Because imbalanced datasets is a common phenomenon in some new social network sites, this paper proposed the MW-KNN algorithm to deal with this issue. In dealing with imbalanced datasets, we proposed the proportion reciprocal weight, and then the selection of k nearest neighbor samples is no longer affected by the class imbalance problem. We also combined with the distance weight, ensuring that the weight of the sample close to the test object is large so that the performance of the classification is more ideal. The experimental results demonstrated its effectiveness.

In future work, we can further study the distribution of the samples in the MW-KNN algorithm, making the weight assigned to the sample more accurate to achieve more ideal classification performance.

Acknowledgements

This work is supported by “the Fundamental Research Funds for the Central Universities” no. 1000-10817453 and “Beijing Municipal Social Science Foundation” No. 16GLC067.

References

1. T. Cover, P. Hart, “Nearest Neighbor Pattern Classification,” *IEEE Trans. Inf. theory*, vol.13, no.1, pp.21-27, 1967.
2. H. Dubey, V. Pudi, “ Class based Weighted K-nearest Neighbor over Imbalance Dataset,” *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.305-316,2013.
3. A. Fernández, S. García, F. Herrera, “Addressing the Classification with Imbalanced Data: Open Problems and New Challenges on Class Distribution,” *Hybrid Artificial Intelligent Systems*, pp.1-10, 2011.
4. B. He, W. Fang, Q. Luo, et al. “Mars: a MapReduce Framework on Graphics Processors,” *International Conference on Parallel Architectures and Compilation Techniques IEEE*, PP. 260-269, 2008.
5. J. V. Hulse, T. Khoshgoftaar, “Knowledge Discovery from Imbalanced and Noisy Data,” *Data & Knowledge Engineering*, vol.68, no.12, pp.1513-1542, 2009.
6. Z. Hajizadeh, M. Taheri, M. Z. Jahromi. “Nearest Neighbor Classification with Locally Weighted Distance for Imbalanced Data,” *International Journal of Computer and Communication Engineering*, vol.3, no.2, pp. 81-86, 2014.
7. N. Japkowicz, S. Stephen, “The Class Imbalance Problem: A Systematic Study,” *Intelligent Data Analysis*, vol.6, no.5, pp. 429-449, 2002.
8. W. Liu, S. Chawla, “Class Confidence Weighted Knn Algorithms for Imbalanced Data Sets,” *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.345-356, 2011.
9. R. Min, D. A. Stanley, Z. Yuan, et al, “A Deep Non-linear Feature Mapping for Large-margin KNN Classification,” *Ninth IEEE International Conference on Data Mining (ICDM'09)* IEEE Computer Society, pp.357-366, 2009.

10. H. Patel, G. S. Thakur, "A Hybrid Weighted Nearest Neighbor Approach to Mine Imbalanced Data," *Proceedings of the International Conference on Data Mining (DMIN)*, pp.106-110, 2016.
11. D. Ramyachitra, P. Manikandan, "Imbalanced Dataset Classification and Solutions: A Review," *International Journal of Computing and Business Research*, vol.5,no.4, pp.1-9, April 2014.
12. S. Tan, "Neighbor-weighted K-nearest Neighbor for Unbalanced Text Corpus," *Expert Systems with Applications*, vol.28, no.4, pp. 667-671, 2005.
13. R. Vernica, M. J. Carey, C. Li, "Efficient Parallel Set-similarity Joins Using MapReduce," Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, PP. 495-506, 2010.
14. S. Vajda, G. A. Fink, "Strategies for Training Robust Neural Network based Digit Recognizers on Unbalanced Data Sets," *International Conference on Frontiers in Handwriting Recognition* IEEE, pp.148-153, 2010.
15. K. Q. Weinberger, L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *The Journal of Machine Learning Research*, vol.10, no.1, pp. 207-244, 2009.
16. X. D. Wu, and V. Kumar, W. B. Li, S. Y. Wu, "The Top Ten Algorithms in Data Mining," Tsinghua University press, Beijing, 2013.
17. Y. Wang, L. Xu, "Research on Text Categorization of KNN based on K-Means for Class Imbalanced Problem," *Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control IEEE*, pp.579-583,2016
18. Q. Yang, X. Wu, "10 Challenging Problems in Data Mining Research," *International Journal of Information Technology & Decision Making*, vol.5, no.4, pp.597–604, 2006.
19. T. Yang, L. Cao, C. Zhang, "A Novel Prototype Reduction Method for the K-nearest Neighbor Algorithm with K≥1," Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, 2010.
20. Y. Yan, T. Ma, J. Wang. "Parallel Implementing KNN Classification Algorithm Using MapReduce Programming Mode," *Journal of Nanjing University of Aeronautics & Astronautics*, vol.45, no.4, pp.550-555, 2013.
21. W. Zhao, H. Ma, Q. He, "Parallel K-means Clustering based on MapReduce," Cloud Computing, Springer, Berlin, Heidelberg, 2009.

Qimin Cao is an associate research fellow of the Library of China University of Political Science and Law. Her research interests include system engineering and information analysis.

Lei La is an associate professor of the School of Information Technology & Management at the University of International Business and Economics. His research interests include semi-supervised machine learning and network data analysis.

Hongxia Liu is an associate research fellow of the Library of China University of Political Science and Law. Her research interests include information resource management and information analysis.

Si Han is a lecturer of the School of Information Management for Law at the China University of Political Science and Law. Her research interest is information security.