ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2012. No. 7(123)
ELEKTRONIKA IR ELEKTROTECHNIKA

SYSTEM ENGINEERING, COMPUTER TECHNOLOGY

SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

An Improved kNN Algorithm based on Essential Vector

Weidong Zhao, Shuanglin Tang
School of Software, Fudan University
Weihui Dai
School of Management, Fudan University,
220, Handan Road, Shanghai, 200433, P.R. China, Phone: 86-21-25011241, e-mail:whdai@fudan.edu.cn

crossref http://dx.doi.org/10.5755/j01.eee.123.7.2389

Introduction

The intelligent classification is an important task in data mining, and it is widely used in many fields. It looks for the common characteristics of data in the classified data set, and determines the classification of the data set. Common classification methods include decision tree, Bayesian classification, support vector machine, neural network, rough set, fuzzy set theory, genetic algorithm, k-nearest neighbor algorithm, case-based reasoning and so on. And the k-nearest neighbor (kNN) algorithm has been used widely at present as it is simple, effective and non-parametric [1].

To improve the efficiency problem, many scholars have done further researches on the k-nearest neighbor algorithm and proposed many improved methods from different perspectives: the first method is to simplify the computation by decreasing the quantity of training samples. For example, Li and Hu proposed a density-based method [2]. And Han and Karypis put forward a centroid-based method [3]. Also, Shin K et al proposed an improved method by removing outliers from the training set [4]. The second method is to find the nearest neighbor fast by introducing the fast search algorithm or establishing efficient indexing. For example, Pan et al. proposed the KWENNS algorithm and Yu Wang et al. gave the FKNN algorithm [5,6]. The third method is to lower the computation complexity by decreasing the dimension of the vector space. For example, Adrian et al discussed an improved method through margin maximization [7]. However, only a small quantity of samples can be cut off by these methods, and the efficiency, if increased, is at the cost of prediction accuracy. For example, Wang et al. proposed a method by narrowing the search range of kNN with the accuracy decreased by one percentage [8].

To reduce the computation complexity, an improved kNN algorithm based on the essential vector is proposed in this paper, which can cut out most of the samples but keep the same accuracy. The improved algorithm calculates the distance between the training sample vector and the essential vector. So it can cut out most training samples and reduce the calculation of similarity effectively. It improves the traditional kNN by essential vector calculation, so it is named EV-kNN (essential vector - k nearest neighbor).

The rest of this paper is organized as follows. Section 2 briefly introduces the traditional kNN algorithm. The improved algorithm EV-kNN is proposed in section 3. Experiment results are discussed in section 4. The application of EV-kNN is analyzed in section 5. Finally, section 6 concludes this paper.

Traditional kNN algorithm

As the k-nearest neighbor algorithm (kNN) is a typical instance-based learning method and it is widely used in text classification, customer analysis and so on. Its basic idea is that an object is classified according to the majority vote of its neighbors, with the object being assigned to the class most of its k nearest neighbors belongs to. In this paper, the Euclidean distance is employed for similarity computation.

The kNN algorithm can be described below. Given a training set, the algorithm firstly calculates the distance between the sample x and the training set, and then finds the closest k training samples. Thus we can assign x to the class which most of the k training samples are classified as. The k training samples are determined by using the (1)

$$C_k = \min_1^k (D(x, c_{ij})), \quad (1)$$

where $D(x,c_{ij})$ is the distance between x and $c_{ij,}$ i stands for the number of class, j stands for the number of samples in the training set. If most of the k training samples satisfy the condition $max(C_k) \in C_n$, then the test sample x belongs to the class C_n.

Improved kNN Algorithm

The essential vector proposed in this paper is the vector which can reflect the essence of classification. It is different from the principal component in principal component analysis (PCA). The principal component is a smaller set of variables by converting the original correlative variable set by means of orthogonal transformation. It may be unable to reflect the essence of classification，so it is not the essential vector. Similarly, the essential vector is different from prototype vector, the centroid, the kernel function of SVM and the concept of kernel in knowledge reduction.

The improved algorithm EV-kNN first constructs a kNN classifier based on the essential vector to get the k candidate classes fast, and then constructs another kNN classifier with these k classes. So, it reduces the number of training samples through calculation of the first kNN classifier, and then uses the second kNN classifier to classify left training samples. The efficiency is therefore increased significantly. The extraction method of the essential vector is different in terms of the application area. As to software classification, the detailed extraction procedure is described in this paper.

In the field of IT asset management, application recognition is to determine which software is to be installed on the target machine by matching file characters. The main file(s) is (are) one or more files that can well reflect the essence of application classification. For example, if "firefox.exe" is the main file which is located in the installed directory for a Mozilla Firefox, then the installed file set can be identified as this application through the main file. In the application, the main file is viewed as the essential vector, and its extraction rules can be described as follows:

- Rule 1 Adopt the same file if it is extracted as the main file in another version of the same software;
- Rule 2 Mark the executable file (*.exe, *.com) as the main file directly if there is only one executable file;
- Rule 3 Extract the main file from several executable files by calculating the similarity;
- Rule 4 Make all the executable files as main files if they can not be found through the steps above.

Specifically for Rule 3, it splits the application name and saves the result into the main file list. For example, the probable name of Microsoft word's main file may be Microsoft word, microsoftword, word and Microsoft. It traverses all executable file names and tries to find the matched one in the main file list.

These are the general rules of the extraction algorithm. The extraction process is executed in the data pre-processing stage. So the efficiency of kNN classification is not reduced.

The EV-kNN algorithm executes kNN calculations twice: the kNN algorithm is firstly used to calculate the similarity between the test sample vector and the essential vector, and then the similarity between the test sample vector and all vectors is computed again.

In application classification, the kNN is applied in a simple matching process between the filename of each test sample and the main filename. The kNN classification process is employed to describe details. Given a training sample file set $S(s_1,s_2,...,s_n)$ and a queried file set $Q(q_1,q_2,...,q_m)$, the algorithm is described as followings:

1. Sort the two file sets based on one of the file characters e.g., filename;
2. Adjust the number of files in the two file sets compared. For example, given a sample file set {a, b, c, f, g, y, z} which belongs to a known application and a query file set which is to be classified is {a, c, d, h, w, y, z}. Merge the two sets into a new one {a, b, c, d, f, g, h, w, y, z}. And then adjust the original two file sets. The sample set becomes {a, b, c, null, f, g, null, null, y, z}, and the query set is changed into {a, null, c, d, null, null, h, w, y, z};
3. Calculate the deviation with the Euclidean distance between each sample file x and query file y using the equation (2)

$$d_g(x,y) = \sum_g^h k_g \left| x_g - y_g \right|, \quad (2)$$

where g is one feature, k_g is a weight value for this feature;

4. Get the average distances between two file sets using the equation (3)

$$D(S,Q) = \frac{1}{h}\sum_g^h d_g(x,y); \quad (3)$$

5. The distance between the test set Q and the training set S is determined, and then it is saved into a temporary array. Querying the other file set with the steps above is repeated until all the file sets in the software repository are traversed.

Finally, sort the array by average distance and choose the closest sample to make the final decision whether it is the same class as this sample.

According to the extended Relief algorithms [9] and cross-testing, the weight of the main file character is acquired as shown in table 1. If the similarity is greater than 0.81, the two file sets should belong to the same version of the same application.

Table 1. Weight of file characters

Characters	Weight（%）
File name	50
File size	30
File signature	2
File version	1

Experiments

In this section, the comparative experiments for the traditional kNN algorithm, the improved kNN based on essential vector and several other improved algorithms are shown.

The experimental data in this paper comes from the application repository of the Xseries Company including 32522 categories. From this repository, 1000 data sets are chosen. And all kinds of training data are simulated through these data sets with a script program. The simulation process is described as follows. First, copy the data set of Firefox. Second, modify the value of certain features. Finally, specify the simulated data set as another version of Firefox. More than 30,000 simulated data sets and the previous 1000 real data sets are adopted as the training set. The weight of table 1 and the default k value

(30) are also adopted in the experiment. The test sample includes 100 file sets and each file set has no more than 30 files. Each test adopts the average value of performance measures after every algorithm runs 10 times.

As classifying the files can be regarded as one kind of machine learning process, the commonly used performance measures in machine leaning are precision and recall.

Another measurement is the classification time which can reflect the efficiency of computation. In this experiment, the classification time and the F1 measure are adopted for the performance comparison in traditional kNN, EV-kNN, Rocchio-kNN and Center-kNN algorithm.

First, pre-process the data. The majority of data is generated by the script program and has been normalized. In this step, the main files are marked by the extraction process. Second, run traditional kNN with test samples and save the result. Third, run EV-kNN and other improved algorithms with test samples, and save the result. Finally, compare the experiment results.

Experiment I first compares the classification time with several different algorithms mentioned above based on the similar conditions. The first scenario focuses on the performance comparison with increasingly growing class number. The number of classes are 100, 500, 1000, 3000, 5000, 6000 and 10000 respectively. The results are illustrated in Fig. 1.

In Fig. 1, the horizontal abscissa is the number of classes of training samples; the vertical ordinate represents the time consumption of classification. It indicates that the EV-kNN algorithm is more efficient than the traditional kNN algorithm because of the sharp decrease of the number of training samples, and it is slightly better than the other improved algorithms.

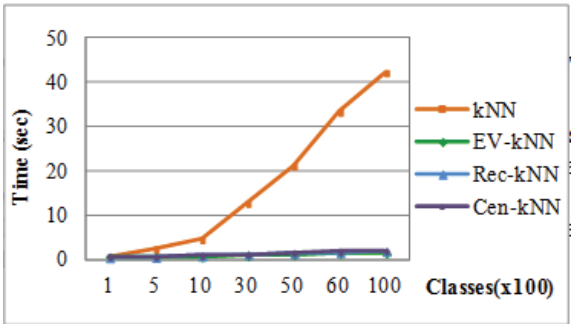

Fig. 1. Classification time comparison with different number of classes

The second scenario focuses on the performance comparison with the same number of training samples (10000 classes) and the increasingly growing number of the essential vector. The number of essential vector are 100, 500, 1000, 3000, 5000, 6000, 8000 and 10000 respectively. The results are shown in Fig. 2.

In Fig. 2, the horizontal abscissa is the number of classes that includes the essential vector; the vertical ordinate represents the consumption of classification time. It shows that the EV-kNN algorithm is more efficient than the traditional kNN algorithm. The classification time seems to be stably less in several other improved algorithms because they do not care about the number of essential vectors.

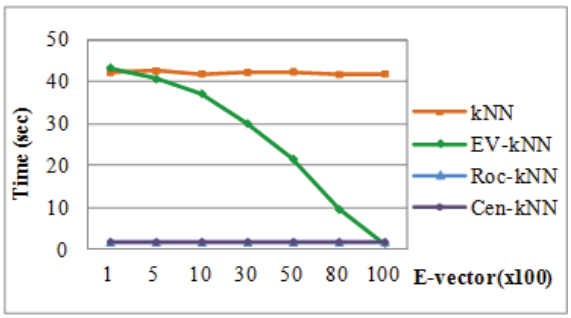

Fig. 2. Classification time comparison with different number of Essential vectors

Experiment II compares the prediction accuracy with different algorithms in the similar conditions. It adopts the same training samples as experiment I with the growing number of test samples (1000, 5000, 10000, 15000, 20000 and 25000). The new 30 training samples are obtained by cutting out the original sample sets in the improved algorithms above (k=30). The results are shown in Fig. 3.

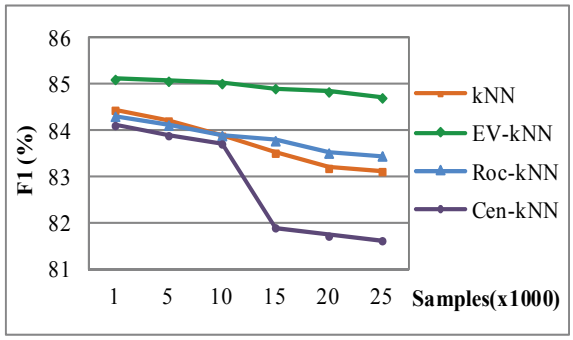

Fig. 3. Comparison on classification accuracy by F1

The EV-kNN algorithm performs better than the other algorithms as shown in Fig. 3 (the average value of F1 is one percentage higher). The reason is that it cuts off n-k classes through the first kNN based on the essential vector and reduces the interference of the boundary samples, thus the EV-kNN algorithm enhances the classification accuracy. The accuracy of the Rocchio-kNN and kNN seems to be slightly worse, but the accuracy of the Center-kNN drops significantly because of the effect of data noise.

Experimental results show that the improved algorithm EV-kNN has obvious improvement of efficiency compared with several traditional kNN algorithms. Meanwhile, it does not bring about decrease in quality. First, the calculation time of the essential vector is much less than that of the training samples which are cut out. Second, the improved algorithm EV-kNN significantly reduces the number of training samples, while ensuring that there is no significant decrease in prediction accuracy. The time complexity of kNN is O(nmk). Generally speaking, only different versions of the same software have the same main file, so that the number of left training samples is around five after the first kNN (about five different versions for one software). It means that the average time is equivalent to the time of scanning five training samples and the related time complexity is reduced to O(nmk). So, when n is infinitely large, the algorithm time complexity is O($5mk$) according to the basic limit theorem, and the time is no longer affected by

the number n of training samples. Finally, the prediction accuracy decreases in a small number of improved algorithms, because the real class the test samples belong to is not included in the new training samples after the sample set is cut out. Therefore, the classifier may generate a completely wrong prediction when using the kNN again. The EV-kNN is improved by employing the essential vector. Consequently, it is more reliable for kNN classification and the left samples are trustworthy so that the final prediction accuracy is higher.

This algorithm has been used in the products of software recognition of the Xseries Company. It can get almost all the vectors so that the EV-kNN algorithm is ran smoothly. The class number of training data could be reduced from 30,000 to around five, thus significantly enhance the efficiency of recognition. Simultaneously, the improved algorithm shows higher classification accuracy.

Conclusions

In this paper, the EV-kNN algorithm is proposed to reduce the number of training samples by the first kNN calculation with the essential vector. It also lowers the computation complexity and improves the prediction accuracy distinctly comparing with the traditional k-nearest neighbor algorithm. However, it is not only applied to application classification but also text classification. It is worth further research on how to get the essential vector from text through the calculation of word frequency, so that the improved algorithm is eventually extended to the field of text classification in the future.

Acknowledgements

This work was supported partly by the Natural Science Foundation of China (71071038). Many thanks to Hongzhi Hu for her assistant work to the corresponding author Weihui Dai of this article.

References

1. **David W. A., Dennis K., Marc K. A.** Instance–based learning algorithms // Machine learning, 1991. – Vol. 6. – No. 1. – P. 37–66.
2. **Li R., Hu Y.** A density–based method for reducing the amount of training data in kNN text classification // Computer Research and Development, 2004. – Vol. 45. – No. 4. – P. 539–544.
3. **Han E., Karypis G.** Centroid–based document classification: analysis and experimental results // Proceedings of 4th European Conference on Principles of Data Mining and Knowledge Discovery. – Springer, 2000. – P. 424–431.
4. **Kwangcheol S., Ajith A., Sang Y**. Improving kNN text categorization by removing outliers from training set // Proceedings of 7th International Conference on Computational Linguistics and Intelligent Text Processing. – Springer, 2000. – P. 563–566.
5. **Jengshyang P.,Yulong Q., Shenghe S**. A fast K nearest neighbors classification algorithm // IEICE Trans. Fundam. Electron. Commun. Comput. Sci., 2004. – No. 4. – P. 961–963.
6. **Yu W., Zhengou W.** A fast KNN algorithm for text categorization // Proceedings of 6th International Conference on Machine Learning and Cybernetics. – IEEE Press, 2007. – P. 19–22.
7. **Adrian P., Francesc J. F., Jesús V. A.** An online metric learning approach through margin maximization // Proceedings of 5th Iberian conference on Pattern recognition and image analysis. – Springer, 2001. – P. 500–507.
8. **Jingzhong W., Xia L**. An improved KNN algorithm for text classification // Proceedings of 2010 International Conference on Information Networking and Automation. – IEEE Press, 2010. – P. 436–439,
9. **Xuning C., Weiping Z**. Network analysis for exploring systems biology // Proceedings of 3rd International Conference on Biomedical Engineering and Informatics. – IEEE Press, 2010. – P. 2578–2581.
10. **Kononenko I**. Estimating attributes: analysis and extension of RELIEF // Proceedings of European Conference on Machine Learning. – Springer, 1994. – P. 171–182.

Received 2012 03 19
Accepted after revision 2012 05 12

Weidong Zhao, Shuanglin Tang, Weihui Dai. An Improved kNN Algorithm based on Essential Vector // Electronics and Electrical Engineering. – Kaunas: Technologija, 2012. – No. 7(123). – P. 119–122.

There are some limitations in traditional k-nearest neighbor (kNN) algorithm, one of which is the low efficiency in classification applications with high dimension and large training data. In this paper, an improved kNN algorithm EV-kNN is proposed to reduce the computation complexity by cutting off the number of training samples. It firstly gets k classes by the kNN calculation with the essential vector, then assigns corresponding category using the kNN again. Experimental results show that the improved algorithm can perform better than several other improved algorithms. Ill. 3, bibl. 10, tabl. 1 (in English; abstracts in English and Lithuanian).

Weidong Zhao, Shuanglin Tang, Weihui Dai. Pagerintas kAK algoritmas pagrįstas pagrindiniu vektoriumi // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2012. – Nr. 7(123). – P. 119–122.

Yra keletas tradicinio k-artimiausiojo kaimyno (kAK) algoritmo apribojimų. Vienas iš jų – mažas klasifikavimo aplikacijų efektyvumas naudojant didelės apimties mokymosi duomenis. Skaičiavimų kompleksiškumui sumažinti apkarpant mokymo imtis, pasiūlytas pagerintas kAK algoritmas PV-kAK. Pirmiausia išskiriamos k klasės skaičiuojant kAK su pagrindiniu vektoriumi, paskui, vėl naudojant kAK, priskiriama atitinkama kategorija. Eksperimento rezultatai rodo, kad pagerintas algoritmas yra našesnis nei keletas kitų algoritmų. Il. 3, bibl. 10, lent. 1 (anglų kalba; santraukos anglų ir lietuvių k.).