Soft Comput (2008) 12:647–655
DOI 10.1007/s00500-007-0242-3

FOCUS

An empirical comparison of min–max-modular *k*-NN with different voting methods to large-scale text categorization

Ke Wu · Bao-Liang Lu · Masao Utiyama · Hitoshi Isahara

Published online: 16 October 2007

Abstract Text categorization refers to the task of assigning the pre-defined classes to text documents based on their content. *k*-NN algorithm is one of top performing classifiers on text data. However, there is little research work on the use of different voting methods over text data. Also, when a huge number of training data is available online, the response speed slows down, since a test document has to obtain the distance with each training data. On the other hand, min–max-modular *k*-NN (M^3-*k*-NN) has been applied to large-scale text categorization. M^3-*k*-NN achieves a good performance and has faster response speed in a parallel computing environment. In this paper, we investigate five different voting methods for *k*-NN and M^3-*k*-NN. The experimental results and analysis show that the Gaussian voting method can achieve the best performance among all voting methods for both *k*-NN and M^3-*k*-NN. In addition, M^3-*k*-NN uses less *k*-value to achieve the better performance than *k*-NN, and thus is faster than *k*-NN in a parallel computing environment.

Keywords Text categorization · *k*-NN algorithm · Min–max-modular *k*-NN · Parallel computing

The work of K. Wu and B. L. Lu was supported in part by the National Natural Science Foundation of China under the grants NSFC 60375022 and NSFC 60473040, and the Microsoft Laboratory for Intelligent Computing and Intelligent Systems of Shanghai Jiao Tong University.

K. Wu · B.-L. Lu (✉)
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, 800 Dong Chuan Road,
Shanghai, 200240, China
e-mail: wuke@sjtu.edu.cn

B.-L. Lu
e-mail: bllu@sjtu.edu.cn

M. Utiyama · H. Isahara
Knowledge Creating Communication Research Center,
National Institute of Information and Communications Technology,
3-5 Hilaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
e-mail: mutiyama@nict.go.jp

H. Isahara
e-mail: isahara@nict.go.jp

1 Introduction

The ever-increasing Web documents are available over Internet, which makes it difficult to manage these documents and to retrieve useful information from these documents. Text categorization has become one of the most important techniques to handle the problem. Text categorization aims to automatically assign documents into some predefined categories. Many machine learning methods (Yang and Chute 1994; Joachims 1997; Joachims 1998; Nigam et al. 1999; Yang and Liu 1999; Sebastiani 2002) have been proposed to this end. Among these methods, *k*-NN algorithm has been widely used as a basic algorithm for text categorization, since it is simple yet efficient. In Yang (1999), Bergo (2007), Sebastiani(2002), it shows that although not perfect, *k*-NN is the best overall performing system on diverse sets. Despite its good performance, *k*-NN, however, has not been employed as widely in applications where very large high-dimensional data are involved. One main reason is the computational complexity of distance computation in high-dimensional space, often regarded as prohibitive.

On the other hand, parallel processing is an effective technique for scaling up the learning algorithm. Lu and Ito (1999) proposed a min–max-modular (M^3) network for solving large-scale and complex multi-class classification problems effortlessly and efficiently. And the network model has been applied to learning large-scale, real world multi-class problems such as part-of-speech tagging and classification of high-dimensional, single-trial electroencephalogram signals (Lu et al. 2004b). Recently, Lu et al. (2004a) have proposed a part-versus-part task decomposition method and a new modular support vector machine, called min–max-modular support vector machine (M^3-SVM), which was developed for solving large-scale pattern classification problems (Wang et al. 2005; Fan and Lu 2005; Liu et al. 2005a; Lian et al. 2005; Yang and Lu 2006; Luo and Lu 2006). Meanwhile, Zhao and Lu (2004, 2006) combined k-NN with M^3-network and obtained a comparable yet efficient performance to k-NN based on the Euclidean distance metric.

The k-NN classifier tries to estimate the conditional class probabilities from samples in a local region of the data space. Usually, the majority voting method and Euclidean distance metric is used. However, majority voting method simply assumes the k nearest neighbors of a data point to be contained in a region of relatively small volume, so that sufficiently good resolution in the estimation of different conditional densities can be obtained. Unfortunately, this is not the case. To estimate different conditional densities, some other voting methods need to be explored. To our best knowledge, few literature investigates k-NN with different voting methods on the text data, which are high-dimensional and sparse. In addition, M^3-k-NN is a parallel ensemble of multiple k-NN classifiers, which boost the test speed in parallel computing environments. Therefore, we are interested in application of different voting methods to M^3-k-NN. In the paper, we introduce some voting techniques other than majority voting method into M^3-k-NN to investigate the effect of performance on M^3-k-NN.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to min–max-modular k-NN. Section 3 introduces k-NN classifier and four voting methods in detail. Experiments and discussions are presented in Sect. 4. Section 5 concludes this paper.

2 Min–max-modular *k*-NN

In this section, we give a brief introduction to the min–max-modular k-Nearest Neighbor (M^3-k-NN). M^3-k-NN is a framework that divides a complex classification problem into many relatively smaller independent two-class subproblems, construct sub-classifers by k-NN algorithm, and then integrates these smaller k-NN classifiers according to two module combination rules, namely the minimization principle and the maximization principle (Lu and Ito 1997; Lu and Ito 1999). The model has three basic components: task decomposition, concurrent learning, and module combination. Since k-NN is an algorithm with little learning, we only focus on task decomposition and module combination.

2.1 Task decomposition

Let $\mathcal{T}$ be the training set for an M-class classification problem,

$$\mathcal{T} = (X_l, Y_l)_{l=1}^{L}, \tag{1}$$

where $X_l \in \mathcal{X} \subset \mathbf{R}^n$ is the input vector, $Y_l \in \mathcal{Y} \subset \mathbf{R}^M$ is the desired output, and L is the total number of training data.

It has been suggested that an M-class problem can be divided into $M(M-1)$ relatively smaller two-class subproblems based on the class relations among training data (Lu and Ito 1997). However, in these $M(M-1)$ two-class subproblems, we can observe that half of subproblems need to be learned and another half of subproblems are the same as the previous ones from the point of view of pattern classification. The training set for each of the two-class subproblems is given by

$$\mathcal{T}_{ij} = \{(X_l^{(i)}, +1)\}_{l=1}^{L_i} \cup \{(X_l^{(j)}, -1)\}_{l=1}^{L_j}$$
$$\text{for } i = 1, 2, \ldots, M \quad \text{and} \quad j = i+1, \ldots, M \tag{2}$$

where $X_l^{(i)} \in \mathcal{X}_i$ and $X_l^{(j)} \in \mathcal{X}_j$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$ respectively, $\mathcal{X}_i$ is the set of training inputs belonging to class $\mathcal{C}_i$, $\mathcal{X}_j$ is the set of training inputs belonging to class $\mathcal{C}_j$, L_i and L_j denotes the numbers of data in $\mathcal{X}_i$ and $\mathcal{X}_j$, respectively, $\bigcup_{i=1}^{M} \mathcal{X}_i = \mathcal{X}$, and $\sum_{i=1}^{M} L_i = L$.

If some of the two-class problems defined by Eq. 2 are still large and hard to be learned, such problems can be further broke into many two-class problems as small as the user expects (Lu and Ito 1999). Assume that $\mathcal{X}_i$ is partitioned into $N_i (1 \le N_i \le L_i)$ subsets in the form

$$\mathcal{X}_{ij} = \left\{X_l^{(ij)}\right\}_{l=1}^{L_i^{(j)}} \text{ for } j = 1, \ldots, N_i$$
$$\text{and} \quad i = 1, \ldots, M, \tag{3}$$

where $\bigcup_{j=1}^{N_i} \mathcal{X}_{ij} = \mathcal{X}_i$.

According to the above partition of $\mathcal{X}_i$, the two-class problem $\mathcal{T}_{ij}$ defined by Eq. 2 can be divided into $N_i \times N_j$ relatively smaller and simpler two-class subproblems. The training set for each of the two-class subproblems is given by

$$\mathcal{T}_{ij}^{(u,v)} = \left\{\left(X_l^{(iu)}, +1\right)\right\}_{l=1}^{L_i^{(u)}} \cup \left\{\left(X_l^{(jv)}, -1\right)\right\}_{l=1}^{L_j^{(v)}}$$
$$\text{for } u = 1, \ldots, N_i, v = 1, \ldots, N_j,$$
$$i = 1, \ldots, M \quad \text{and} \quad j = i+1, \ldots, M. \tag{4}$$

where $X_l^{(iu)} \in \mathcal{X}_{iu}$ and $X_l^{jv} \in \mathcal{X}_{jv}$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$

From Eqs. 2 and 3, we see that an M-class problem can be decomposed into $\sum_{i=1}^{M-1}\sum_{j=i+1}^{M} N_i \times N_j$ two-class subproblems. This decomposition process is simple and domain-specific knowledge concerning the problem decomposition is not required. Therefore, any large-scale problem can be easily decomposed into a number of two-class subproblems as small as a user needs.

For ease of description, we assume that each class has the same number of training data J. Let L be the total number of training data for an M-class classification problem, and $L = M \times J$. If an M-class problem is decomposed into $\binom{M}{2}$ two-class subproblems, the number of training data for each of the two-class subproblems is $2 \times J$.

If an M-class problem is decomposed into $\sum_{i=1}^{M-1}\sum_{j=i+1}^{M} N_i \times N_j$ two-class subproblems, the number of training data for each of the two-class subproblems is about

$$\lceil J/N_i \rceil + \lceil J/N_j \rceil \tag{5}$$

where $\lceil z \rceil$ denotes the smallest integer greater than or equal to z.

Since each of the two-class subproblems can be treated as completely independent problems in the training phase, all of the problems can be learned in parallel.

2.2 Module combination

These completely independent modules need to be combined in a reasonable and elegant way. Consequently, M^3 model has proposed a simple yet effective way to perform it. After training each module which is assigned to learn associated subproblems, all of the individual trained modules can be easily integrated into an M^3 network according to the minimization principle and the maximization principle.

Specifically, these submodules are integrated into a M^3-network with $L_i^{(u)}$ MIN units and one MAX unit according to two combination principles (Lu and Ito 1997; Lu and Ito 1999; Lu and Ichikawa 2000) as follows:

$$\mathcal{B}_{ij}^{u}(x) = \min_{v=1}^{L_i^{(v)}} \mathcal{B}_{ij}^{(u,v)}(x)$$

and

$$\mathcal{B}_{ij}(x) = \max_{u=1}^{L_i^{(u)}} \mathcal{B}_{ij}^{u}(x) \tag{6}$$

where $\mathcal{B}_{ij}^{(u,v)}(x)$ denotes the output function of the classifier corresponding to the two-class subproblem $\mathcal{T}_{ij}^{(u,v)}$, $\mathcal{B}_{ij}^{u}(x)$ denotes the transfer function of a combination of multiple classifiers integrated by the MIN unit and $\mathcal{B}_{ij}(x)$ denotes the resulting output of a combination of multiple MIN units by MAX principle.

3 Voting methods for k-NN

3.1 k-NN algorithm

The k-NN algorithm has been introduced by Fix and Hodges (Fix and Hodges 1951) and has since become well-known in the pattern recognition literature. Also, it is one of top performing methods of automatic text categorization (Yang and Liu 1999; Sebastiani 2002; Lewis et al. 2004; Liu et al. 2005b). In text categorization, k-NN algorithm typically is used as follows. Given a test document d, the system finds the k nearest neighbors among training documents, and exploit their classes to weight class candidates. The similarity score of each nearest neighbor document to the test document is used as the weight of the classes of the neighbor document. If more than one of k nearest neighbors share a class, these weights of that class are added together, and the resulting weighted sum is used as the likelihood score of that class regarding the test document. Consequently, a ranked class list is obtained for the test document by sorting the scores of each possible classes. The decision rule can be formulated as follows:

$$\text{score}(d, c_i) = \sum_{d_j \in k\text{NN}(d)} \text{Sim}(d, d_j)\delta(d_j, c_i) \tag{7}$$

where $k\text{NN}(d)$ indicates the set of k nearest neighbors of document d; $\delta(d_j, c_i)$ is 1 if document d_j belongs to class c_i and 0 otherwise and $Sim(d, d_j)$ denotes the similarity metric between d and d_j. For test document d, it is assigned the class with the highest score.

3.2 Different voting methods

A k-NN classifier tries to estimate the conditional class probabilities from samples in a local region of the data space. Typically, we apply a majority voting method to k-NN, which regards each neighbor in its neighbors equally. However, we might want to weight nearer neighbors more heavily. Cover and Hart (1967) have proved that as the number N of samples and k tend to infinity in such a manner that $k/N \to 0$, the error rate of the k-NN rule approaches the optimal Bayes error rate. However, in the finite sample case, the majority-voting k-NN rule is not guaranteed to be optimal way of exploited the information provided by the neighborhood of unclassified patterns, since it simply assume that the sufficiently good resolution in the estimated different conditional densities in a region of relatively small volume. In practice, since the number of samples is finite, the closest neighbors do not provide the sufficient information about the different conditional densities. Thus, the distance between x and one of its closest neighbors may provide more clues for the estimation of the conditional densities.

Dudani (1976) proposed three different methods. The first is referred to linear voting method, in which the nearest neighbor gets a weight of 1, the furthest neighbor a weight of 0, and the other weights are scaled linearly to the interval in between. The method can be formalized as follows:

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1}, & \text{if } d_k \neq d_1 \\ 1, & \text{if } d_k = d_1 \end{cases} \tag{8}$$

where d_j is the distance to the query of the jth nearest neighbor, d_1 is the nearest neighbor, and d_k is the furthest (or kth) neighbor.

The second is the inverse distance, that is, the neighbors of a query are assigned the weight reciprocally to the distance to vote for the predicted class. In text categorization, this voting policy is the variant of the method, since a similarity measure is the converse of a distance function. In pattern recognition community, the distances between a query and samples in training set are calculated, while in text categorization, similarities between a query and samples in training set are used. Usually, similarity metric is the reciprocal of distance metric. For clarity, we use distance metric in this paper. The inverse distance method can be formulated as follows:

$$w_j = \frac{1}{d_j} \quad \text{if } d_j \neq 0 \tag{9}$$

The third is the rank voting method. Its basic idea is to consider the ranking position according to the ranked list of the distances between a query and samples in training set and then to weight more the sample with high rank. We formulate the method as follows.

$$w_j = k - j + 1 \tag{10}$$

In this paper, Gaussian voting method was also investigated, since a lot of events conform to Gaussian distribution. It is also based on the same idea as three aforementioned voting methods that a training sample closer to the test sample will have a higher influence on the final classification compared to a sample that is further away from the test sample. However, various methods employ difference descending function. Gaussian voting method uses the following equation:

$$w_j = \frac{1}{\sqrt{2\pi}\delta} \exp^{-d_j^2/2\delta^2} \tag{11}$$

Now, to generalize k-NN algorithm with different voting methods, we reformulate Eq. 7 as follows:

$$\text{score}(d, c_i) = \sum_{d_j \in k\text{NN}(d)} \mathcal{W}(\text{dis}(d, d_j)\delta(d_j, c_i)) \tag{12}$$

where $\text{dis}(d, d_j)$ denotes the distance between document d and document d_j, and $\mathcal{W}$ denotes the weight function.

Table 1 Distributions of training and test data of the extracted subset from Yomiuri news corpus

#	Category	# data		
		Training	# of subsets	Test
c_1	Events	31354	98	2714
c_2	Core industry	39608	124	3363
c_3	Environment	21247	67	1371
c_4	Culture	3344	11	357
c_5	Food	19645	62	1992
c_6	Labor	5757	18	317
c_7	Fishery	320	1	17
c_8	Diplomacy	767	3	43
c_9	Music	2274	8	164
c_{10}	Social security	3654	12	226
	total	127970	404	10564

Here '# of subsets' indicates the number of sub-modules randomly divided by M^3-k-NN when the size of a module is 320

4 Experiments

4.1 Experimental setup

In the section, we present our results on Yomiuri News Corpus, which is the largest corpus currently consisting of 2,190,512 documents in the full collections from the years 1987 to 2001. Since the simulations are conducted on a PC machine with Pentium D 2.8 GHz CPU and 2.0 GB memory, time spent on a very large amount of corpus is large and thus we extract one subset of the corpus, which contains 127,970 training samples and 10,564 test samples. The details are described in Table 1.

In all the experiments, we use 5000 features for which χ^2 statistical method was applied and adopt *tf-idf* weighting scheme to weight each feature. Additionally, each vector representation is L2-normalized to a unit vector. In the simulations, according to Eq. 2, the text categorization problem is divided into $\binom{10}{2} = 45$ two-class subproblems. From Table 1 and the definition of the two-class subproblems, we see that the number of training data for the smallest two-class subproblem $\mathcal{T}_{78}$ is 1,087, while the largest two-class subproblem $\mathcal{T}_{12}$ is 71,062. Although these two-class subproblems are smaller than the original problem, they are not adequate for massively parallel computation. In order to speed-up response of k-NN algorithm during recognition phase, we should further decompose each of the bigger two-class subproblems into a number of relatively smaller and simpler two-class subproblems. By using the decomposition method, each of the training sets for the bigger two-class subproblems is randomly divided into a number of relatively smaller subsets according to Eq. 4.

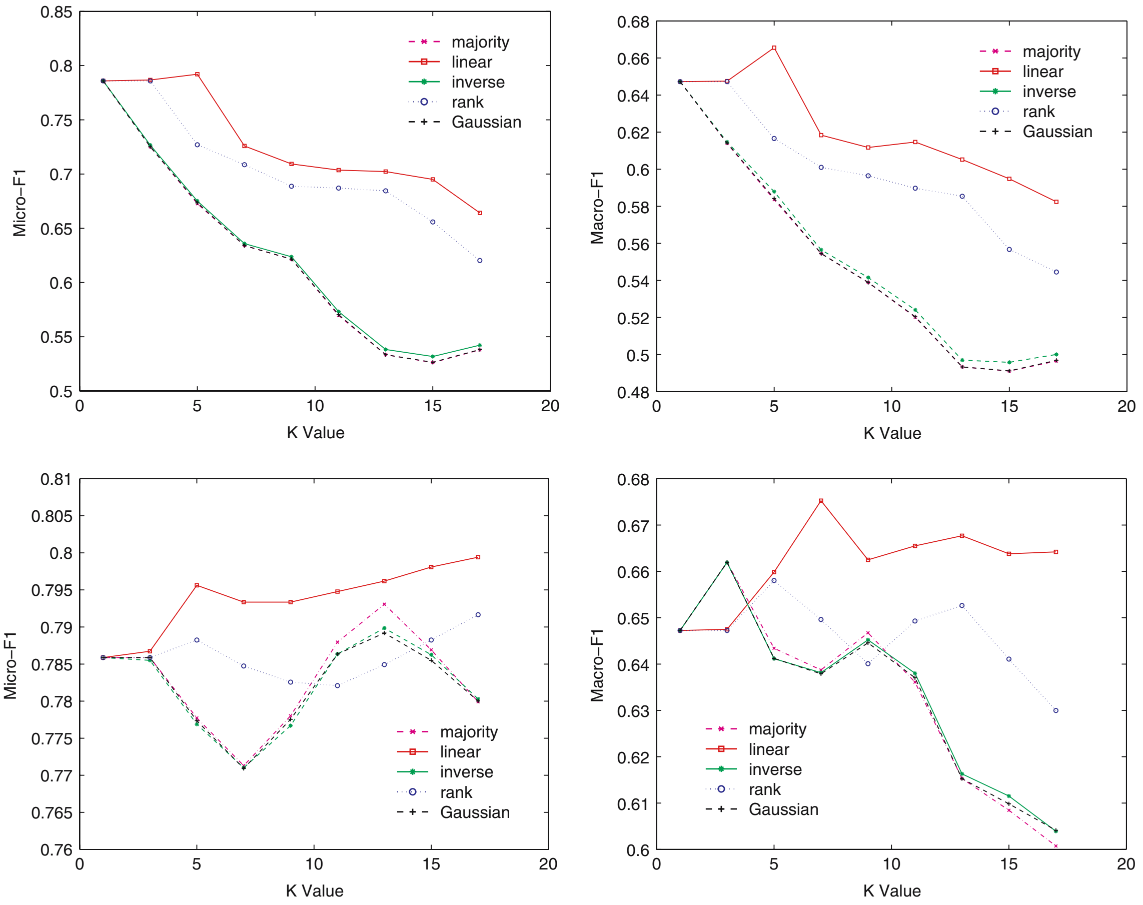

Fig. 1 Micro-$F1$ and Macro-$F1$ results of different weighted-distance methods with Euclidean distance. The *first row* shows the results of M^3-k-NN, and the *second row* shows the results of k-NN

In the simulation, the size of the modular is taken as 320, since the number of the smallest class in training set is 320. The samples in each class are divided approximately equally. In addition, two distance metrics were used. One is Euclidean distance metric, which is a common metric used in pattern recognition community. Another is cosine distance metric, which is typically used to measure the similarity between documents. However, for a voting method, it is typically based on distance metric, which is dissimilarity between documents. Therefore, for a uniform presentation, we used angular distance metric as another metric in our experiments. Specifically, the angular distance corresponds to the arc cosine between two documents.

4.2 Performance measure

To evaluate the effectiveness of a text categorization system, we use the standard recall, precision and $F1$ measure. Recall is defined to be the ration of the number of the correctly assigned documents to the number of positive samples. Precision is the ratio of the number of the correct documents in the positively assigned documents. The $F1$ measure combines recall and precision in the following way:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

For convenience, we summarize the $F1$ scores over the different categories by the macro-average of $F1$ scores and over all the samples by the micro-average $F1$ scores. The two measures are called Macro-$F1$ and Micro-$F1$, respectively. Macro-$F1$ emphasizes the performance of the system on rare categories, while Micro-$F1$ embody the performance of the system on major categories since they contain more samples.

In addition, Macro-recall is defined to be average recall over categories and Macro-precision is defined to be average precision over categories.

4.3 Results and discussion

The experimental results of k-NN and M^3-k-NN using Euclidean distance metric are shown in Figs. 1 and 2. The results of k-NN and M^3-k-NN using angular distance metric are shown in Figs. 3 and 4. When Gaussian voting method is used for k-NN and M^3-k-NN based on Euclidean distance

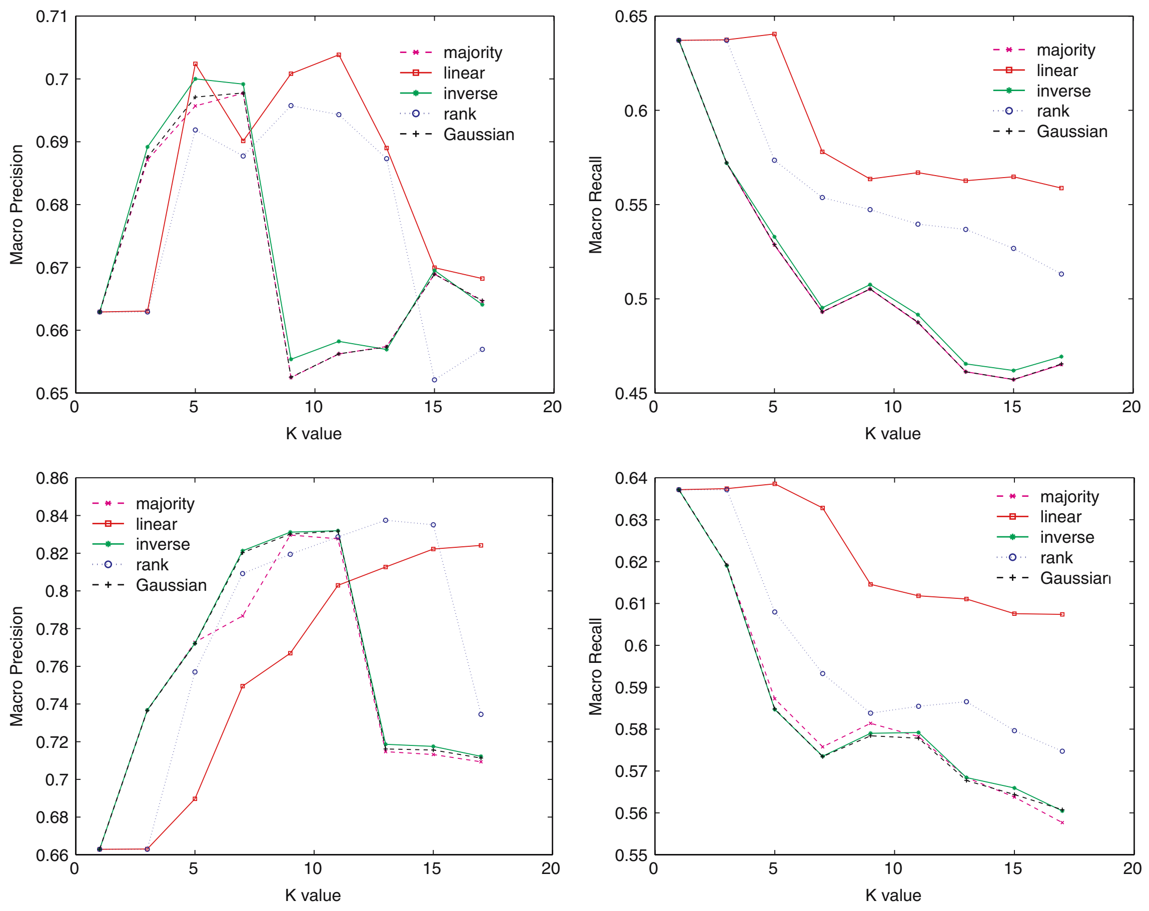

Fig. 2 Micro-precision and Macro-recall results of different weighted-distance methods with Euclidean distance. The *first row* shows the results of M^3-k-NN, and the *second row* shows the results of k-NN

metric, δ value was taken 1.0 empirically. In addition, when angular distance metric was used, δ value was set to 0.4 empirically.

- We can observe that M^3-k-NN and k-NN can achieve almost the same best performance on both Micro-$F1$ and Macro-$F1$. It is worthwhile to notice that the Macro-$F1$ of M^3-k-NN is higher than k-NN, when angular distance metric is applied. Although data decomposition makes data more sparse, M^3-network structure results in a comparatively good performance.
- M^3-k-NN uses less k value to achieve its peak performance on both Micro-$F1$ and Macro-$F1$ than traditional k-NN. Multiple parallel modules lead to more samples involved in the final decision and thus a small k value is needed to achieve the optimal performance.
- When we use Euclidean distance metric, Macro-precision is raised with k value increasing. However, an opposite case occurs when angular distance metric is used. This may be because angular distance metric is more appropriate for text data than Euclidean distance.
- Linear voting method is superior to the other methods at larger values of k and its best performance is better than other methods.

In order to have a close look to the five different voting methods, these methods are depicted in Fig. 5. The linear voting method is an adaptive voting method which adapt the effective size of the neighborhood to the local properties of the data and thus often can achieve a good performance in all experiments. In contrast, inverse distance voting method makes a global assumption about the relevance gradient, which leads to a comparatively poor performance. In addtion, since $\frac{1}{\sqrt{2\pi\delta}}$ is a constant after δ is assigned, Guassian voting function just uses $\exp^{-d_j^2/2\delta^2}$ with δ=1 for illustration. For Guassian voting method, an assumption is made that the relation between a distance and its voting weight conforms to Guassian distribution. In Euclidean space, sparse and high-dimensional characteristics of text data results in large distance between a query and training samples and thus its practical performance is similar to the majority voting method.

From Fig. 5, we can see that the weight is a number close 0 when the distance is greater than 2. That is, the nearest

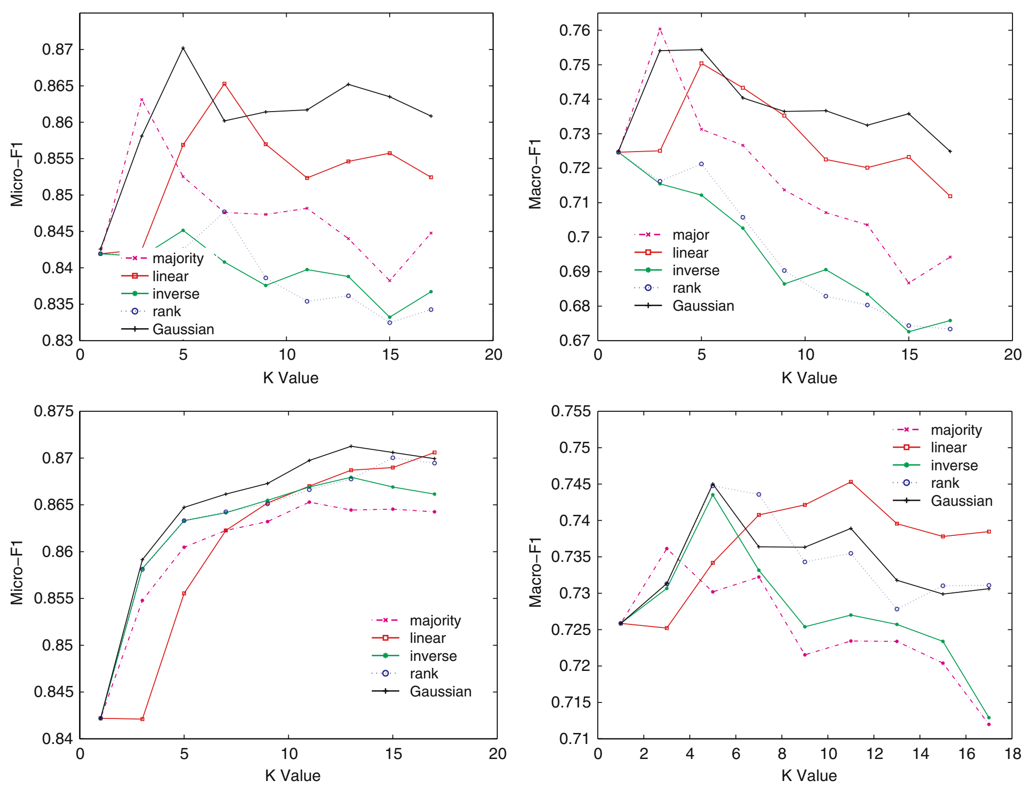

Fig. 3 Micro-$F1$ and Macro-$F1$ results of different weighted-distance methods with the angular distance metric. The *first row* shows the results of M^3-*k*-NN, and the *second row* shows the results of *k*-NN

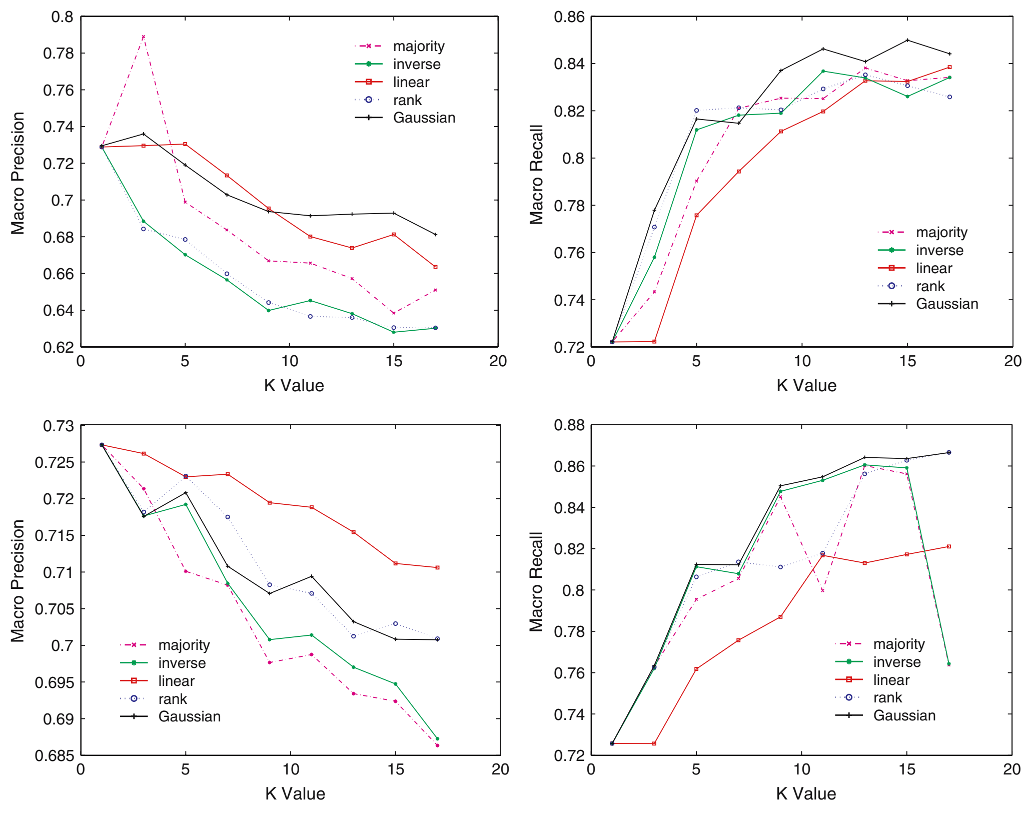

Fig. 4 Micro-precision and Macro-recall results of different weighted-distance methods with the angular distance metric. The *first row* shows the results of M^3-*k*-NN, and the *second row* shows the results of *k*-NN

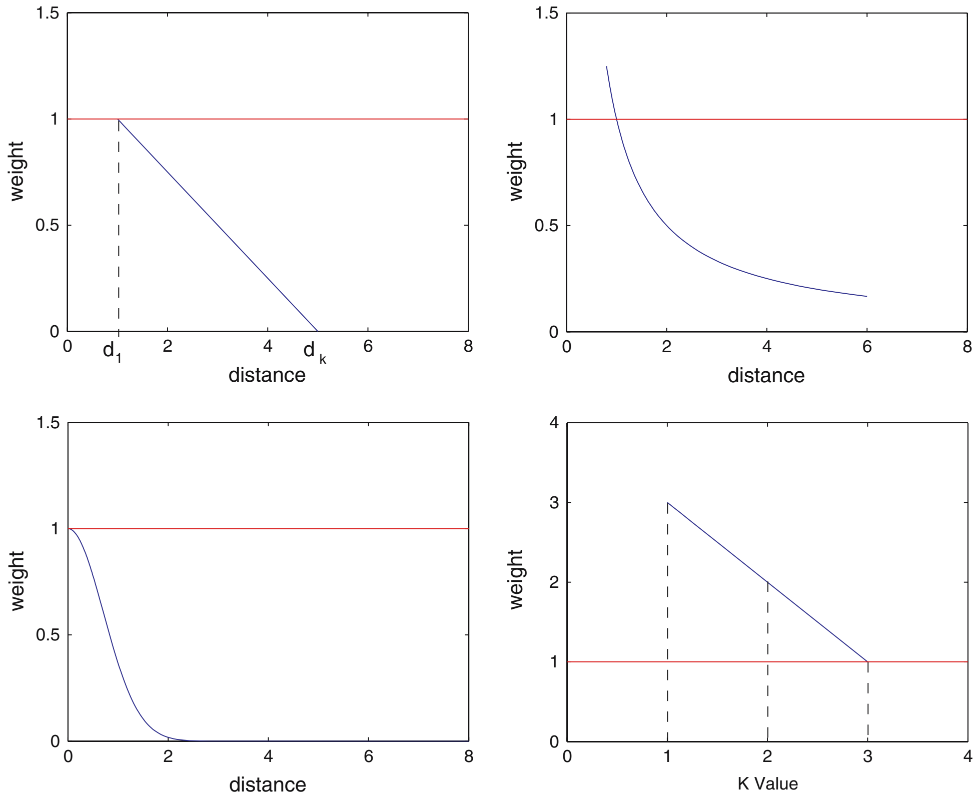

Fig. 5 Illustration of four different voting methods. *Left top* is linear voting method; *right top* is inverse distance voting method; *left bottom* is Gaussian voting method; and *right bottom* is rank voting method. Here d_1 denotes the distance of the nearest neighbor and d_k denotes the distance of the furthest (kth) nearest neighbor, and *red line* indicates majority voting method

neighbors with the distance greater than 2 are treated equally. However, its overall performance is the best over micro-$F1$ measure among the five methods in a comparatively compact distance space. It benefits from the narrow interval of angular distance ($[0, \pi/2]$). In addition, rank voting method is also an adaptive method, but it ignores the length of the distance and focuses on the rank of the distance. Therefore, it does not produce an expected performance. As for majority voting method, it is a common method for voting. Although for k-NN, a comparatively poor performance is produced, for M^3-k-NN based on the angular method, the best macro-$F1$ value is generated.

To compare M^3-k-NN with k-NN, the best performance for each voting method is summarized in Table 2. From this table, we can see that Guassian voting method based on angular distance achieves the best micro-$F1$ performance over both k-NN and M^3-k-NN. In addition, for k-NN, linear voting method obtains the best macro-$F1$ performance, while for M^3-k-NN, simple majority voting method gets the best macro-$F1$ performance.

4.4 Time complexity analysis

For an M-class problem, we suppose that the number of all training samples is N. Predication for one new query will be done in $O(N)$, since k-NN needs to retrieval all training samples to acquire k nearest neighbors. When M^3-k-NN is used in the simulations for an M-class problem, the M-class problem is first decomposed into $M(M-1)/2$ two-class problem and then each two-class problem is handled via M^3-k-NN. Although it seems to increase computational load, the computation can be done in a parallel way. When M^3-k-NN is not used in a parallel environment, the computation can be done in about $2N/M$. If we apply M^3-k-NN to each two-class problem in a parallel way, the computational time will be sharply down with the decrease of each modular size. In the extreme case, if the number of positive samples is M_1

Table 2 The best performance and the corresponding k value of different voting methods based on two distance metrics

Voting method	k-NN (%)				M^3-k-NN (%)			
	Micro-$F1$	k	Macro-$F1$	k	Micro-$F1$	k	Macro-$F1$	k
Majority	79.31	13	66.20	3	78.59	1	64.72	1
	86.53	11	73.61	3	86.31	3	**76.04**	**3**
Linear	79.94	17	67.53	7	79.20	5	66.56	5
	87.07	17	**74.53**	**11**	86.53	7	75.04	5
Inverse	78.99	13	66.20	3	78.59	1	64.72	1
	86.79	13	74.35	5	84.51	5	72.50	1
Rank	79.17	17	65.58	5	79.09	3	65.01	3
	87.00	15	74.50	5	84.77	7	72.12	5
Gaussian	78.92	13	66.20	3	78.59	1	64.72	1
	87.13	**13**	74.50	5	**87.02**	**5**	75.44	5

For each voting method, the data in the first row is based on Euclidean distance metric, and the data in the second row is based on angular distance metric and the number of negative samples is M_2 in each smaller modular classifier, the computation will be done in $2N/(M \times M_1 \times M_2)$.

5 Conclusion

In this paper, we have compared several voting methods for k-NN and M^3-k-NN. The experimental results show that the majority voting method can achieve the best macro-$F1$ performance when M^3-k-NN is applied to text categorization, while linear voting method can obtain the best macro-$F1$ performance when k-NN is used. In addition, the Gaussian voting method can achieve the best micro-$F1$ performance for both k-NN and M^3-k-NN. From the experiment results, we can see that M^3-k-NN has two important advantages over traditional k-NN. 1) M^3-k-NN can use less k value to achieve peak performance than k-NN, and 2) M^3-k-NN can spend less time to complete prediction than k-NN in a parallel computing environment. As to future work, we will use a combination of several distance-weighted policies for M^3-k-NN, since different methods have complementary performance when k value is small, and improve the Gaussian voting method by automatically adjusting its δ parameter.

References

Bergo A (2007) Text categorization and prototypes. (In: http://www.illc.uva.nl/Publications/ResearchReports/MoL-2001-08.text.pdf)

Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inform Theory IT-13(1):21–27

Dudani S (1976) The distance-weighted k-nearest-neighbor rule. IEEE Trans Syst Man Cybern SMC-6:325–327

Fan ZG, Lu BL (2005) Multi-view face recognition with min–max modular svms. In: ICNC (2), pp 396–399

Fix E, Hodges J (1951) Discriminatory analysis, nonparametric discrimination: consistency properties. Technical report, USAF Scholl of aviation and medicine, Randolph Field 4

Joachims T (1997) A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: Fisher DH (ed) Proceedings of ICML-97, 14th international conference on machine learning, Morgan Kaufmann Publishers, San Francisco, USA, pp 143–151

Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: Nédellec C, Rouveirol C (eds) Proceedings of ECML-98, 10th European conference on machine learning, Springer, Heidelberg, DE, pp 137–142 (Published in the “Lecture Notes in Computer Science” series, number 1398)

Lian HC, Lu BL, Takikawa E, Hosoi S (2005) Gender recognition using a min–max modular support vector machine. In: ICNC (2), pp 438–441

Lewis DD, Yang Y, Rose TG, Li F (2004) Rcv1: A new benchmark collection for text categorization research. J Mach Learn Res 5:361–397

Liu FY, Wu K, Zhao H, Lu BL (2005a) Fast text categorization with min–max modular support vector machines. In: IEEE international joint conference on neural networks, vol 1, pp 570–575

Liu TY, Yang Y, Wan H, Zhou Q, Gao B, Zeng HJ, Chen Z, Ma WY (2005b) An experimental study on large-scale web categorization. In: WWW ’05: special interest tracks and posters of the 14th international conference on World Wide Web, ACM Press, New York, NY, USA, pp 1106–1107

Lu BL, Ichikawa M (2000) A Gaussian zero-crossing discriminat function for min–max modular neural networks. In: Proceedings of 5th international conference on knowledge-based intelligent information engineering systems and allied technologies (KES’01), pp 298–302

Lu BL, Ito M (1997) Task decomposition based on class relations: a modular neural network architecture for pattern classification. In: Mira J, Moreno-Diaz R, Cabestany J (eds) Biological and artificial computation: from neuroscience to technology, Lecture Notes in Computer Science, vol 1240. Springer, Heidelberg, pp 330–339

Lu BL, Ito M (1999) Task decomposition and module combination based on class relations: A modular neural network for pattern classification. IEEE Trans Neural Netw 10(5):1244–1256

Lu BL, Wang KA, Utiyama M, Isahara H (2004a) A part-versus-part method for massively parallel training of support vector machines. In: Proceedings of 2004 IEEE international joint conference on neural networks, pp 735–740

Lu BL, Shin J, Ichikawa M (2004b) Massively parallel classification of single-trial EEG signals using a min–max-modular neural network. IEEE Trans Biomed Eng 3(51):551–558

Luo J, Lu BL (2006) Gender recognition using a min–max modular support vector machine with equal clustering. In: ISNN (2), pp 210–215

Nigam K, Lafferty J, McCallum A (1999) Using maximum entropy for text classification. In: IJCAI-99 workshop on machine learning for information filtering, pp 61–67

Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv 34(1):1–47

Wang K, Zhao H, Lu BL (2005) Task decomposition using geometric relation for min–max-modular svms. In: ISNN (1), pp 887–892

Yang Y (1999) An evaluation of statistical approaches to text categorization. Inf Retrieval 1(1/2):69–90

Yang Y, Chute CG (1994) An example-based mapping method for text categorization and retrieval. ACM Trans Inf Syst 12(3):252–277

Yang Y, Liu X (1999) A re-examination of text categorization methods. In: Hearst MA, Gey F, Tong R (eds) Proceedings of SIGIR-99, 22nd ACM international conference on research and development in information retrieval, ACM Press, New York, USA, pp 42–49

Yang Y, Lu BL (2006) Prediction of protein subcellular multi-locations with a min–max modular support vector machine. In: ISNN (2), pp 667–673

Zhao H, Lu BL (2004) A modular k-nearest neighbor classification method for massively parallel text categorization. In: International symposium on computational and information sciences (CIS’04), LNCS, vol 3314, pp 867–872

Zhao H, Lu BL (2006) A modular reduction method for k-nn algorithm with self-recombination learning. In: ISNN (1), pp 537–544