Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Accelerated *k*-nearest neighbors algorithm based on principal component analysis for text categorization*

Min DU, Xing-shu CHEN‡

(*School of Computer Science, Sichuan University, Chengdu 610065, China*)

E-mail: doingscu@gmail.com; chenxsh@scu.edu.cn

Received Oct. 24, 2012; Revision accepted Apr. 1, 2013; Crosschecked May 13, 2013

Abstract: Text categorization is a significant technique to manage the surging text data on the Internet. The *k*-nearest neighbors (kNN) algorithm is an effective, but not efficient, classification model for text categorization. In this paper, we propose an effective strategy to accelerate the standard kNN, based on a simple principle: usually, near points in space are also near when they are projected into a direction, which means that distant points in the projection direction are also distant in the original space. Using the proposed strategy, most of the irrelevant points can be removed when searching for the *k*-nearest neighbors of a query point, which greatly decreases the computation cost. Experimental results show that the proposed strategy greatly improves the time performance of the standard kNN, with little degradation in accuracy. Specifically, it is superior in applications that have large and high-dimensional datasets.

Key words: *k*-nearest neighbors (kNN), Text categorization, Accelerating strategy, Principal component analysis (PCA)
doi:10.1631/jzus.C1200303 **Document code:** A **CLC number:** TP391

1 Introduction

The number of documents available online has been increasing greatly in recent years. Finding relevant information in time is important for many applications, and text categorization is the key technique for this task (Shang *et al*., 2007). Text categorization has been used in many applications such as spam email filtering (Zhou *et al*., 2010), Web page classification (Qi and Davison, 2009), customer relationship management (Coussement and van den Poel, 2008), and text sentinel classification (Wang *et al*., 2011).

In recent years, many text categorization methods have been developed to allocate text documents to their annotated categories, such as *k*-nearest neighbors (kNN) (He *et al*., 2003), Bayesian (Lee *et al*., 2012), support vector machine, artificial neural networks (de Souza *et al*., 2009), Rocchio (Miao and Kamel, 2011), latent Dirichlet allocation (Wang *et al*., 2012), and many other machine learning and statistical approaches (Chen *et al*., 2011). The kNN algorithm, which has a simple structure, has shown great potential in text categorization.

When using the standard kNN, similarities have to be computed between the unknown sample and all training samples in order to find the exact *k*-nearest neighbors of the query sample. Hence, standard kNN has a fatal defect in that the process of similarity computing is very time-consuming, especially when the dimension of the feature space is high and the training set is very large. Text categorization, as is well known, typically has a high dimension and very large training set, which makes kNN less applicable.

To improve the time performance of standard kNN, an accelerating strategy is proposed in this work. The strategy is based on a simple principle: usually, near points in space are also near when they are projected into a direction, which means that distant points in the projection direction are also distant in the

‡ Corresponding author

* Project (No. 2012BAH18B05) supported by the National Key Technology R&D Program of China

original space. Hence, distant points to the query point in the projection direction are removed when searching for the *k*-nearest neighbors of the query point, which may greatly reduce sample similarity computing. However, the choice of direction is important to the effect of projection. In addition, due to the nature of the *N*-dimensional to one-dimensional mapping, some nearest neighbors may be positioned far apart along the projection direction. To decrease the possibility that these points are overlooked during a search, several appropriate directions are found and considered comprehensively in this strategy. Experimental results show that the proposed strategy greatly improves the time performance of the standard kNN with little degradation in accuracy, and that it is superior in the applications that have large and high-dimensional datasets.

2 Related works

The kNN classification approach has been widely used in various types of classification tasks. This classification approach has gained popularity based on low implementation cost and high degree of classification effectiveness. However, its sample similarity computing is very large, which limits its applications in some cases that have high dimensional spaces or very large training sets. Many researchers have sought ways to reduce the complexity of kNN, which can be classified into three approaches generally:

1. Dimensionality reduction approaches. Feature extraction and feature selection are widely used to reduce the dimension of the feature space, such as principal component analysis (PCA), latent semantic indexing, information gain, and χ^2 statistic. However, when the dimension of the text vector is reduced, the remaining dimension is still high to avoid loss of important information.

2. Training samples reduction approaches. A smaller training set means less similarity computation time. These methods usually apply clustering techniques to the training set and then reduce the training set. Jiang SY *et al*. (2012) divided training samples into clusters and then regarded the center of each cluster as the training sample. Jiang JY *et al*. (2012) grouped the training patterns into clusters and then considered only the training documents in those clusters whose fuzzy similarities to the document exceed a pre-designated threshold in searching for the *k*-nearest neighbors for the document. A solution called semantic-center kNN was proposed by Zhang *et al*. (2009) whereby the training set is processed prior to run time to build a so-called semantic space. Essentially, clusters of closely-spaced documents are collapsed into a single point represented by the centroid of the documents.

3. Space/data partitioning approaches. These methods focus on expediting the process of searching for the exact *k*-nearest neighbors. In Wang and Wang (2007), the training set was pre-processed by creating a tree, which allows for the search of the *k*-nearest neighbors of any given test document. This method has no degradation in kNN performance, but it provides a fixed (and thus non-configurable) increase in speed. Jagadish *et al*. (2005) proposed an 'iDistance' method, based on B^+-tree. iDistance partitions the data based on a space- or data-partitioning strategy, and selects a reference point for each partition. The data points in each partition are transformed to a single dimensional value based on their similarity with respect to the reference point. This allows the points to be indexed using a B^+-tree structure and kNN search to be performed using 1D range search. Different from that, Aghbari (2005) linearized the data partitions produced by a data partitioning method, rather than the points themselves, into a 1D array-index, based on which the exact *k*-nearest neighbors can be found quickly.

The proposed strategy aims to speed up the process of searching for *k*-nearest neighbors while keeping the advantages of kNN, which is a simple and facile method. By this strategy the approximate, instead of the exact, *k*-nearest neighbors to a query point are retrieved, which may greatly reduce the similarity computation time. This strategy trades accuracy for speed, since the retrieved neighbors are only approximate. However, it can achieve a very significant reduction in runtime with no more than a one percent degradation in precision. Such a loss in precision is negligible in many applications. Furthermore, in this strategy the degree of acceleration is a configurable parameter; it can be adjusted according to different applications.

3 Accelerating strategy to kNN

3.1 Traditional kNN for text categorization

The process of the kNN algorithm is as follows: given a test sample $\boldsymbol{x}$, the algorithm finds the k-nearest neighbors among all the training samples, and scores the category candidates based the category of the k neighbors. Formally, the decision rule of kNN can be written as

$$g(\boldsymbol{x}) = \arg\max_{j=1,2,\ldots,|c|} \text{Score}(\boldsymbol{x}, c_j) = \sum_{\boldsymbol{d}_i \in \text{kNN}} \text{sim}(\boldsymbol{x}, \boldsymbol{d}_i)\delta(\boldsymbol{d}_i, c_j), \tag{1}$$

$$\text{sim}(\boldsymbol{x}, \boldsymbol{d}_i) = \frac{\boldsymbol{x} \cdot \boldsymbol{d}_i}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{d}_i\|}, \tag{2}$$

$$\delta(\boldsymbol{d}_i, c_j) = \begin{cases} 1, & \boldsymbol{d}_i \in c_j, \\ 0, & \boldsymbol{d}_i \notin c_j, \end{cases} \tag{3}$$

where sim($\boldsymbol{x}$, $\boldsymbol{d}_i$) is the similarity between $\boldsymbol{x}$ and the training sample $\boldsymbol{d}_i$, and cosine similarity is used as the similarity measure in our work. $\delta(\boldsymbol{d}_i, c_j)$ is the classification for sample $\boldsymbol{d}_i$ with respect to class c_j. Score($\boldsymbol{x}$, c_j) is the score of candidate category c_j with respect to $\boldsymbol{x}$. $g(\boldsymbol{x})$ is the predicted label of test sample $\boldsymbol{x}$.

3.2 Methodology

As described before, the traditional kNN approach is effective and easy to implement. However, kNN is a sample-based learning method. It uses all the training samples to predict the labels of the test samples and requires huge text similarity computation. As a result, it cannot be widely used in real-world applications. To tackle this problem, we propose an effective strategy to accelerate the traditional kNN algorithm, which can greatly reduce the time consumed in searching for the k-nearest neighbors.

3.2.1 Vector space model

kNN is a classification approach based on the vector space model (Liu, 2011). In this model, each document $\boldsymbol{d}$ is considered to be a vector in the term-space, $\boldsymbol{d}$=(w_1, w_2, …, w_N), and N is the dimension of the term-space. The weight of each word is computed using term frequency (Liu, 2011).

3.2.2 Key concepts

Definition 1 (Original space) The original space is a term-space built with the vector space model.

Definition 2 (Projection direction) The projection direction is a unit vector of the original space to which the sample points will project. Let $\boldsymbol{p}$ be a projection direction, then $||\boldsymbol{p}||$=1.

Definition 3 (Projection space) The projection space is a space constructed by several projection directions.

Definition 4 (Original samples) Original samples are samples in the original space.

Definition 5 (Projection samples) Projection samples are samples in the projection space.

Consider one of the simplest cases where the dimension of the original space is two (Fig. 1). There are three kinds of samples, A, B, and C. $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are the projection directions. To test sample $\boldsymbol{x}$, first we search for its neighbors in each projection direction. If we collect the two nearest neighbors in each projection direction, then the nearest neighbor set of $\boldsymbol{x}$ in direction $\boldsymbol{p}_1$ is S_1={$\boldsymbol{c}_2$, $\boldsymbol{a}_2$}, and it is S_2={$\boldsymbol{a}_2$, $\boldsymbol{a}_1$} in direction $\boldsymbol{p}_2$. Second, combine the neighbor sets into S={$\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{c}_2$}. Finally, perform the traditional kNN algorithm with set S, instead of all the samples in the original space.

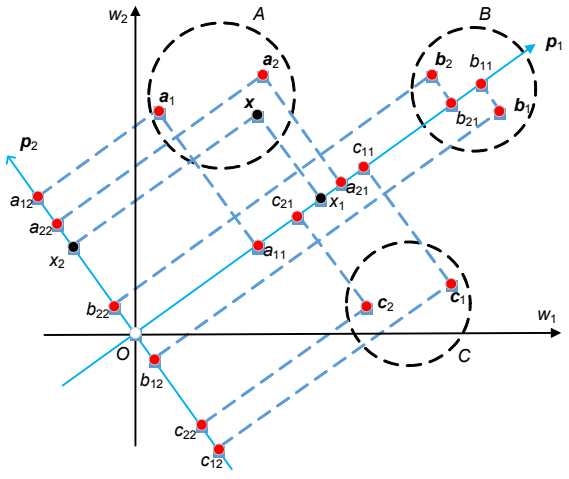

Fig. 1 A two-dimensional projection space

Definition 6 (Projection value) Let $\boldsymbol{p}$=(q_1, q_2, …, q_N)$^{\mathrm{T}}$ be one of the projection directions. Then the projection value of $\boldsymbol{d}$=(w_1, w_2, …, w_N) to $\boldsymbol{p}$ is

$$z=\frac{\boldsymbol{d}\cdot\boldsymbol{p}}{\|\boldsymbol{p}\|}=\frac{\sum w_i q_i}{\sum q_i^2},\quad i=1,2,...,N. \tag{4}$$

Definition 7 (Projection vector) The projection vector is the vector of the projection sample composed by projection values of each projection direction. Suppose there are m projection directions, $\boldsymbol{P}=(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m)$. According to Eq. (4), the projection vector of $\boldsymbol{d}$ can be defined as

$$\boldsymbol{Z}=(z_1, z_2, ..., z_m). \tag{5}$$

3.2.3 Workflow overview

As shown in Fig. 2, from a high-level perspective, the text classification workflow consists of two phases, training and predicting.

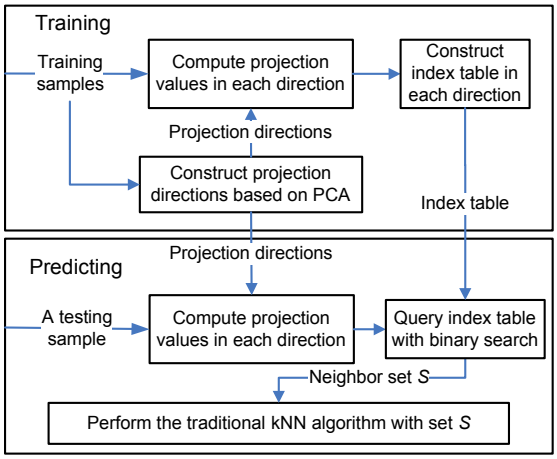

Fig. 2 Workflow of text classification

The main steps are described in Algorithm 1.

Algorithm 1 Accelerated kNN

1. Training phase

Step 1: Construct projection directions $\boldsymbol{P}$ based on PCA, $\boldsymbol{P}=(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m)$, where m is the number of projection directions. This step will be described in detail later.

Step 2: Compute the projection values of the training set in each direction, according to Eqs. (4) and (5).

Step 3: Construct an index table in each direction, which is used to search for neighbors in the predicting phase. In the index table, projection values in each direction are sorted in ascending order. Suppose there are n training text samples. The index table of the ith direction can then be constructed (Fig. 3).

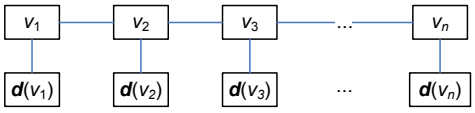

Fig. 3 Index table of each direction

v_i (i=1, 2, …, n) is the projection value of the original sample, $v_1 \le v_2 \le \ldots \le v_n$, and $\boldsymbol{d}(v_i)$ is the vector of the original sample corresponding to v_i

2. Predicting phase

Step 1: For test sample $\boldsymbol{x}$, compute its projection values in each direction by Eqs. (4) and (5).

Step 2: Search the index tables of each direction. We can quickly obtain the L nearest neighbor sets with binary search, owing to ordered index tables. Let $s_i=\{t_{i1}, t_{i2}, \ldots, t_{iL}\}$ be the L nearest neighbor set of the ith direction, and m the number of projection directions. All the possible neighbors in the original space of $\boldsymbol{x}$ can then be defined as

$$S=\bigcup_{i=1}^{m} s_i,\quad |S|\le mL. \tag{6}$$

With the effective projection directions and a proper parameter L, most of the neighbors will be contained in S. In general, S is a very small subset compared with the whole training set. This is the core of the accelerating strategy.

Step 3: Perform the traditional kNN algorithm with set S.

3.2.4 Constructing projection directions based on PCA

There are countless unit vectors in the original space, but not all of them can be used as the projection directions. In fact, most of them are useless in finding a probable neighbor in its direction. Hence, choosing the projection direction is very important to the performance of this strategy, and we should choose those directions that effectively reflect the information of the original space, i.e., the projection directions, to reserve as much original information as possible.

PCA is a widely used method for transforming points in the original (high-dimensional) space into another (usually lower dimensional) space. Using

PCA, most of the information in the original space is condensed into a few dimensions along which the variances in the data distribution are the largest.

Given a set of centered N-dimensional training text samples $\boldsymbol{x}_i$, i=1, 2, …, n, such that $\sum_{i=1}^{n}\boldsymbol{x}_i=\boldsymbol{0}$, where n represents the number of training samples, PCA diagonalizes the following covariance matrix:

$$\boldsymbol{C}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}, \tag{7}$$

where $\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}$ is a vector product that yields an $N\times N$ matrix. Then we need to solve the following eigenvector problem:

$$\lambda\boldsymbol{v}=\boldsymbol{C}\boldsymbol{v}, \tag{8}$$

where $\lambda\in\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ is the eigenvalue of $\boldsymbol{C}$, and $\boldsymbol{v}\in\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N\}$ is the corresponding eigenvector. $\boldsymbol{v}_i$ (i=1, 2, …, N) are the principal component vectors of the training set. The corresponding eigenvectors are ranked in descending order of eigenvalues. Hence, we can choose the first few eigenvectors as the projection directions. To use category information of the training set, PCA is processed on each class of the training set separately in our strategy. An algorithm for choosing projection directions is proposed as follows:

Algorithm 2 Choosing projection directions based on PCA

1. Let m be the category number of the training set. Compute the corresponding covariance matrix $\boldsymbol{C}_i$ (i=1, 2, …, m) of each category by Eq. (7).

2. Compute eigenvectors $\boldsymbol{V}_i$ of $\boldsymbol{C}_i$ according to Eq. (8). Let N be the dimension of the training set. Then $\boldsymbol{V}_i=(\boldsymbol{v}_{i1}, \boldsymbol{v}_{i2}, ..., \boldsymbol{v}_{iN})$, where i=1, 2, …, m.

3. Choose the first eigenvector of each $\boldsymbol{V}_i$ as the projection direction $\boldsymbol{p}_i$:

$$\boldsymbol{P}=(\boldsymbol{v}_{11}, \boldsymbol{v}_{21}, ..., \boldsymbol{v}_{m1})=(\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_m). \tag{9}$$

3.2.5 Two grades of accelerating strategies

Reviewing the last step of the predicting phase in Algorithm 1, perform the traditional kNN algorithm with set S. As described in Section 3.1, this step contains two main sub-steps: (1) Find the k-nearest neighbors of a test sample $\boldsymbol{x}$ among S; (2) Predict the category of $\boldsymbol{x}$ according to the k-nearest neighbors by Eqs. (1)–(3).

Meanwhile, there are two spaces for each sample, the original space and projection space. Hence, the first sub-step can be performed in the original space or projection space. To distinguish those two different branches of the accelerating strategy, call them 'kNN_A1' and 'kNN_A2', respectively, and 'kNN' means the traditional kNN. The main steps of kNN_A1 and kNN_A2 are as follows:

1. Perform the same as in Algorithm 1 except step 3 in the predicting phase.

2. Find the k-nearest neighbors in the original space and projection space in kNN_A1 and kNN_A2, respectively.

3. Predict the category of $\boldsymbol{x}$ according to the k-nearest neighbors obtained in step 2 by Eqs. (1)–(3) in the original space.

3.3 Analysis of time complexity

As described above, kNN is time-consuming, since most of the time is spent on sample similarity computing, and the complexity of sample similarity computing is very large. Hence, we simply regard time consumption on similarity computing as the time complexity of kNN, kNN_A1, and kNN_A2. Some parameters are defined here: n is the size of the training set, N is the dimension of the original space, k is the k value of kNN, m is the category number of the training set (which is the same as the dimension of the projection space), and L is the size of the nearest neighbor set along each projection direction.

According to Eq. (2), the time complexity of similarity computing depends mainly on the dimension of samples. Let $t(x)$ be the similarity computation time for a pair of x-dimensional samples. Table 1 shows the time complexities of kNN, kNN_A1, and kNN_A2.

Table 1 Time complexities of kNN, kNN_A1, and kNN_A2

Algorithm	Time consumption		Time complexity
	Projection space	Original space	
kNN	0	$n\cdot t(N)$	$n\cdot t(N)$
kNN_A1	0	$mL\cdot t(N)$	$mL\cdot t(N)$
kNN_A2	$mL\cdot t(m)$	$k\cdot t(N)$	$mL\cdot t(m)+k\cdot t(N)$

In general, k, m, and L are much smaller than n and N. Thus, kNN_A1 and kNN_A2 have lower time complexity than kNN, and kNN_A2 has the lowest one.

4 Experiments

4.1 Datasets and evaluation criteria

4.1.1 Datasets

The datasets in the experiments come from two publicly available text datasets 20-Newsgroups (http://qwone.com/~jason/20Newsgroups/) and SogouCorpus (http://www.sogou.com/labs/dl/c.html). A series of pre-processing work is done on the original corpus, including word segmentation, dropping stop words, word frequency statistics, and dropping low term-frequency words (term-frequency lower than 20). Table 2 shows the details of the datasets. The training set and testing set are chosen randomly.

Table 2 Details of the two datasets

Dataset	CN	Set size of each category		Total size	Dimension
		Training	Testing		
20-Newsgroups	4	2000	2000	16000	6538
SogouCorpus	6	600	300	5400	6111

CN: category number. The four categories of 20-Newsgroups are: comp, rec, sci, talk; the six categories of SogouCorpus are: finance, health, sports, tourism, education, military

4.1.2 Evaluation criteria

The experiments are performed using Matlab 9.0b on a PC with Pentium Dual-Core E5200 2.5 GHz CPU, 4 GB RAM, and 500 GB hard disk.

We use Macro-F1 and Elapsed-time to measure the algorithm performance:

$$\mathrm{F1}_i=\frac{2r_ip_i}{r_i+p_i},\quad i=1,2,...,m, \tag{10}$$

$$\text{Macro-F1}=\frac{1}{m}\sum_{i=1}^{m}\mathrm{F1}_i, \tag{11}$$

$$\text{Elapsed-time = Time consumption for classifying the testing set,} \tag{12}$$

where r_i is recall, p_i is precision, Macro-F1 is the average of the F1 values of all individual categories, and m is the number of categories. We use Macro-F1 and Elapsed-time to evaluate the classifier performance for a corpus.

In addition, Macro-F1-loss and Acceleration-ratio are used to compare kNN with the improved kNN:

$$\text{Macro-F1-loss=(Macro-F1 of kNN)}-\text{(Macro-F1 of improved kNN)}, \tag{13}$$

$$\text{Acceleration-ratio}=\frac{\text{Elapsed-time of kNN}}{\text{Elapsed-time of improved kNN}}. \tag{14}$$

Abbreviations are used in the following sections: MF1 (%), Macro-F1; ET (s), Elapsed-time; MF1L (%), Macro-F1-loss; AR, Acceleration-ratio.

4.2 Performance comparisons

The proposed approach is compared with the standard kNN and two improved kNN algorithms in the literature: AI-kNN (array-index kNN, which is a typical method based on space/data partitioning) (Aghbari, 2005), and INNTC (improved kNN for text categorization, which is a new method based on reduction of training samples) (Jiang SY *et al*., 2012). Here, we compare kNN_A2 with the baseline methods, and the performances of kNN_A1 and kNN_A2 are analyzed in detail in the following.

4.2.1 Parameter settings

The k value of kNN is set as 10, around which kNN has better performance in our experiments. There are two types of AI-kNN: sphere/rectangle-tree (SR-tree) based and self-organizing map (SOM) based. In this study, we use the SOM based AI-kNN, which has better performance in experiments, for comparison. The output layer of SOM is set as 5×8, the maxIteration in training SOM is set as 50, reference node is center R, and k=10. INNTC has two primary parameters, ε (number of cluster centers) and k, which are set as the empirical values suggested by Jiang SY *et al*. (2012), ε=9 and k=30. Parameters of kNN_A2 are set as L=60, k=50.

4.2.2 Experimental results

The experimental results are presented in Table 3. AI-kNN has no Macro-F1-loss compared with kNN,

and both INNTC and kNN_A2 have a little (about 1%) loss in precision. This is because AI-kNN aims to find the exact k-nearest neighbors while INNTC finds the k-nearest neighbors of the cluster center, and kNN_A2 retrieves the approximate k-nearest neighbors.

Table 3 Performance comparisons

Dataset	Method	Training time (s)	MF1 (%)	ET (s)	MF1L (%)	AR
Sogou-Corpus	kNN		89.89	686.10	0	1
	AI-kNN	36 461	89.89	134.08	0	5.12
	INNTC	70.14	89.08	159.27	0.81	4.31
	kNN_A2	203.64	89.17	29.15	0.72	**23.54**
20-News-groups	kNN		92.35	1578.42	0	1
	AI-kNN	81 024	92.35	304.03	0	5.19
	INNTC	147.29	91.14	366.22	1.21	4.31
	kNN_A2	344.58	91.29	57.35	1.06	**27.52**

MF1: Macro-F1; ET: Elapsed-time; MF1L: Macro-F1-loss; AR: Acceleration-ratio

Both AI-kNN and INNTC are efficient in speeding up kNN, reducing the classification time by a factor of 5, but the training time of AI-kNN is much longer. The proposed method, kNN_A2, speeds up kNN by a factor of more than 20, significantly outperforming AI-kNN and INNTC, and its training time is acceptable.

On the whole, kNN_A2 greatly reduces the classification time of kNN while having very little loss in precision. It will be useful in the fields where one would like to obtain great improvement in time complexity at the expense of a little degradation in precision.

4.3 Performance of kNN_A1

For brevity, only the experiment results of SogouCorpus are shown here, since the performances of kNN_A1 are similar for both datasets. Fig. 4a shows the Macro-F1 obtained by kNN_A1 with different L and k. A larger L leads to a higher Macro-F1. As L is the size of the neighbor set along each projection direction, with a larger L more real neighbors of the original space can be included by S, according to Eq. (6).

Similar to kNN, with the increase of k Macro-F1 increases first, and then decreases, but more slowly, especially at large L. As the enlarged diagram shows, Macro-F1 is close to that of kNN when $L \geq 60$.

Fig. 4b shows the time performance of kNN_A1 with different L and k. A larger L takes more time, and similar to kNN, the Elapsed-time is not affected by k. As shown in Table 1, the time complexity of kNN_A1 is related to L. A larger L leads to a higher time complexity.

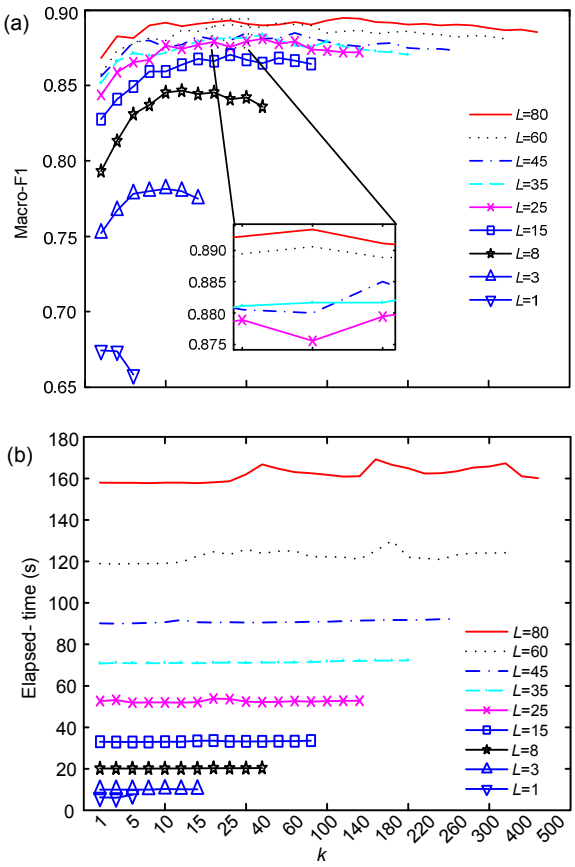

Fig. 4 Macro-F1 (a) and Elapsed-time (b) of kNN_A1

4.4 Performance of kNN_A2

Likewise, only the experiment results of SogouCorpus are given here. Fig. 5a shows the Macro-F1 obtained by kNN_A2. It has the same characteristics as kNN_A1. Fig. 5b shows the time performance of kNN_A2 with different L and k. The Elapsed-time depends mainly on k, and increases rapidly as k increases. As the enlarged diagram shows, Elapsed-time is also slightly affected by L; that is, a larger L leads to a little more time. As shown in Table 1, the time complexity of kNN_A2 is related to both k and L, but m=6 is much smaller than N=6111 in our experiment. Thus, the Elapsed-time depends mainly on k.

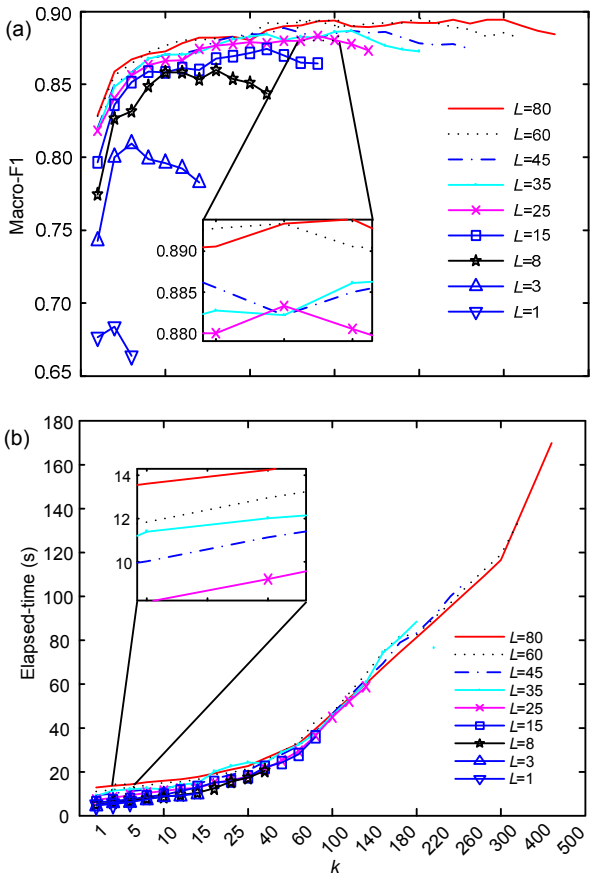

Fig. 5 Macro-F1 (a) and Elapsed-time (b) of kNN_A2

4.5 Comparison of kNN_A1 and kNN_A2

In Table 4, the performances of kNN_A1 and kNN_A2 are compared with those of kNN. The baseline is the highest Macro-F1 of kNN, with k=10, Macro-F1=89.89, Elapsed-time=686 s. Table 4 shows the comparison when k=50 in both kNN_A1 and kNN_A2. Table 5 shows the comparison when L=60 in both kNN_A1 and kNN_A2.

Tables 4 and 5 show that kNN_A1 and kNN_ A2 are both effective in accelerating kNN, and that the Macro-F1-loss is not much.

kNN_A2 usually has a much higher Acceleration-ratio than kNN_A1 at the same Macro-F1-loss. For example, with the Macro-F1-loss of 0.72%, the Acceleration-ratio is 5.49 for kNN_A1, and 23.53 for kNN_A2 (Table 5).

As shown in Table 4, for kNN_A1, Macro-F1 and Elapsed-time both depend mainly on the L value. A larger L leads to a higher accuracy, but also a larger Elapsed-time.

Table 4 Comparisons among kNN, kNN_A1, and kNN_A2 with different L (k=50)

L	MF1 (%)		ET (s)		MF1L (%)		AR	
	A1	A2	A1	A2	A1	A2	A1	A2
10	85.17	85.33	23.56	24.11	4.72	4.56	29.11	28.45
15	86.83	87.00	33.26	24.24	3.06	2.89	20.63	28.30
20	87.72	87.72	42.84	24.72	2.17	2.17	16.01	27.75
25	87.78	88.00	52.20	25.12	2.11	1.89	13.14	27.31
30	88.00	88.22	61.69	27.03	1.89	1.67	11.12	25.38
35	88.00	88.06	71.17	28.98	1.89	1.83	9.64	23.67
40	88.06	88.44	80.89	26.72	1.83	1.45	8.48	25.67
45	88.17	88.89	90.57	27.95	1.72	1.00	7.57	24.54
50	88.56	89.17	100.15	29.61	1.33	0.72	6.85	23.17
60	89.17	89.17	124.90	29.16	0.72	0.72	5.49	23.53
70	88.83	89.33	138.94	28.39	1.06	0.56	4.94	24.16
80	89.06	89.00	164.55	29.40	0.83	0.89	4.17	23.33
100	89.44	89.17	195.05	31.73	0.45	0.72	3.52	21.62

A1: kNN_A1; A2: kNN_A2. MF1: Macro-F1; ET: Elapsed-time; MF1L: Macro-F1-loss; AR: Acceleration-ratio

Table 5 Comparisons among kNN, kNN_A1, and kNN_A2 with different k (L=60)

k	MF1 (%)		ET (s)		MF1L (%)		AR	
	A1	A2	A1	A2	A1	A2	A1	A2
8	88.17	87.22	118.83	13.85	1.72	2.67	5.77	49.53
10	88.00	87.67	119.02	15.00	1.89	2.22	5.76	45.74
12	88.67	88.06	119.48	15.40	1.22	1.83	5.74	44.53
15	88.61	88.06	122.42	16.53	1.28	1.83	5.60	41.51
20	88.94	88.17	124.59	18.74	0.95	1.72	5.51	36.60
25	89.06	88.17	123.32	19.72	0.83	1.72	5.56	34.78
30	88.89	88.39	125.87	21.43	1.00	1.50	5.45	32.01
40	88.89	89.22	123.76	25.03	1.00	0.67	5.54	27.41
50	**89.17**	**89.17**	**124.90**	**29.16**	**0.72**	**0.72**	**5.49**	**23.53**
60	89.06	89.28	125.05	33.47	0.83	0.61	5.49	20.50
80	89.00	89.33	122.20	43.11	0.89	0.56	5.61	15.91
100	88.50	89.06	122.18	46.90	1.39	0.83	5.61	14.63
120	88.61	89.00	121.87	55.81	1.28	0.89	5.63	12.29

A1: kNN_A1; A2: kNN_A2. MF1: Macro-F1; ET: Elapsed-time; MF1L: Macro-F1-loss; AR: Acceleration-ratio

As shown in Table 5, for kNN_A2, Macro-F1 depends mainly on the L value, while the Elapsed-time depends mainly on the k value, which is different from kNN_A1. For kNN_A2, a larger L and a smaller k are the better choice, leading to a higher accuracy and lower running time. Hence, kNN_A2 is superior to kNN_A1. In addition, L and k are configurable parameters, and we can always find a balance between running time and precision in different applications.

4.6 Impacts of training set size and original space dimension

As analyzed in Section 3.3, the time complexity is also related to the training set size and original space dimension. However, how these two parameters affect the performance of our accelerating strategy is unknown. Therefore, we perform experiments with different sizes of training sets and different dimensions of original spaces (Fig. 6). Here, we compare kNN with kNN_A2 at certain values of k and L:

kNN: k=10;
kNN_A2: k=50, L=60.

As shown in Fig. 6, the Acceleration-ratio increases continually as the training data size and original space dimension increase, which indicates that kNN_A2 is superior in cases of very large volume data. In other words, when the training dataset is small, the performance of our approach degrades to the same as that of kNN.

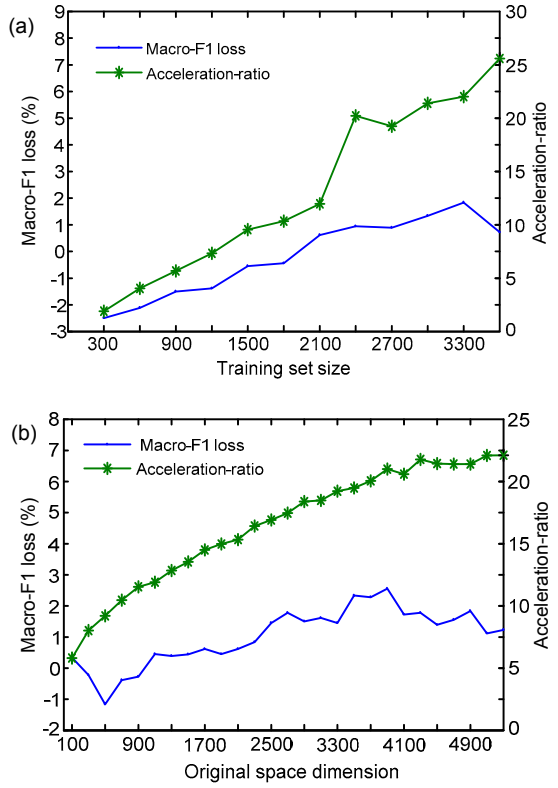

Fig. 6 Impacts of the training set size (a) and original space dimension (b) on the Macro-F1 loss and Acceleration-ratio

Interestingly, for the small-scale data (small dataset or low dimension), Macro-F1-loss is negative, which indicates that kNN_A2 is still better than kNN in terms of both speed and accuracy with respect to small-scale data.

Overall, Macro-F1-loss remains at a relatively low level (which fluctuates by 2%) with increase of the training set size and original space dimension.

5 Discussion and conclusions

This work focuses on improving the time performance of the standard kNN and an effective accelerating strategy is proposed. Experimental results show that this strategy greatly reduces the classification time of kNN, at the expense of little reduction in prediction accuracy. It is superior in the applications that have high dimensions and very large datasets. In practical applications, we can always find a balance between the acceleration level and the degradation in accuracy by adjusting parameters in the strategy.

The proposed strategy can be combined well with the dimensionality reduction approaches and training samples reduction approaches. The combination of different kinds of approaches will be our future work, to achieve better performance in terms of time and accuracy.

References

Aghbari, Z., 2005. Array-index: a plug&search K nearest neighbors method for high-dimensional data. *Data Knowl. Eng.*, **52**(3):333-352. [doi:10.1016/j.datak.2004.06.015]

Chen, E.H., Lin, Y.G., Xiong, H., Luo, Q.M., Ma, H.P., 2011. Exploiting probabilistic topic models to improve text categorization under class imbalance. *Inf. Process. Manag.*, **47**(2):202-214. [doi:10.1016/j.ipm.2010.07.003]

Coussement, K., van den Poel, D., 2008. Integrating the voice of customers through call center emails into a decision support system for churn prediction. *Inf. Manag.*, **45**(3): 164-174. [doi:10.1016/j.im.2008.01.005]

de Souza, A.F., Pedroni, F., Oliveira, E., Ciarelli, P.M., Henrique, W.F., Veronese, L., Badue, C., 2009. Automated multi-label text categorization with VG-RAM weightless neural networks. *Neurocomputing*, **72**(10-12):2209-2217. [doi:10.1016/j.neucom.2008.06.028]

He, J., Tan, A.H., Tan, C.L., 2003. On machine learning methods for Chinese document categorization. *Appl. Intell.*, **18**(3):311-322. [doi:10.1023/A:1023202221875]

Jagadish, H.V., Ooi, B.C., Tan, K.L., Yu, C., Zhang, R., 2005. iDistance: an adaptive B^+-tree based indexing method for

nearest neighbor search. *ACM Trans. Database Syst.*, **30**(2):364-397. [doi:10.1145/1071610.1071612]

Jiang, J.Y., Tsai, S.C., Lee, S.J., 2012. FSKNN: multi-label text categorization based on fuzzy similarity and *k* nearest neighbors. *Expert Syst. Appl.*, **39**(3):2813-2821. [doi:10.1016/j.eswa.2011.08.141]

Jiang, S.Y., Pang, G.S., Wu, M.L., Kuang, L.M., 2012. An improved *K*-nearest-neighbor algorithm for text categorization. *Expert Syst. Appl.*, **39**(1):1503-1509. [doi:10.1016/j.eswa.2011.08.040]

Lee, L.H., Isa, D., Choo, W.O., Chue, W.Y., 2012. High relevance keyword extraction facility for Bayesian text classification on different domains of varying characteristic. *Expert Syst. Appl.*, **39**(1):1147-1155. [doi:10.1016/j.eswa.2011.07.116]

Liu, B., 2011. Web Data Mining (2nd Ed.). Springer, Berlin, Heidelberg, p.217. [doi:10.1007/978-3-642-19460-3]

Miao, Y.Q., Kamel, M., 2011. Pairwise optimized Rocchio algorithm for text categorization. *Pattern Recogn. Lett.*, **32**(2):375-382. [doi:10.1016/j.patrec.2010.09.018]

Qi, X.G., Davison, B.D., 2009. Web page classification: features and algorithms. *ACM Comput. Surv.*, **41**(2):12-42. [doi:10.1145/1459352.1459357]

Shang, W., Huang, H., Zhu, H., Lin, Y., Qu, Y., Wang, Z., 2007. A novel feature selection algorithm for text categorization. *Expert Syst. Appl.*, **33**(1):1-5. [doi:10.1016/j.eswa.2006.04.001]

Wang, B.K., Huang, Y.F., Yang, W.X., Li, X., 2012. Short text classification based on strong feature thesaurus. *J. Zhejiang Univ.-Sci C (Comput. & Electron.)*, **13**(9):649-659. [doi:10.1631/jzus.C1100373]

Wang, S.G., Li, D.Y., Song, X.L., Wei, Y.J., Li, H.X., 2011. A feature selection method based on improved Fisher's discriminant ratio for text sentiment classification. *Expert Syst. Appl.*, **38**(7):8696-8702. [doi:10.1016/j.eswa.2011.01.077]

Wang, Y., Wang, Z.O., 2007. A Fast KNN Algorithm for Text Categorization. Proc. 6th Int. Conf. on Machine Learning and Cybernetics, p.3436-3441. [doi:10.1109/ICMLC.2007.4370742]

Zhang, X., Huang, H., Zhang, K., 2009. KNN Text Categorization Algorithm Based on Semantic Centre. Proc. Int. Conf. on Information Technology and Computer Science, p.249-252. [doi:10.1109/ITCS.2009.57]

Zhou, B., Yao, Y.Y., Luo, J., 2010. A Three-Way Decision Approach to Email Spam Filtering. Proc. 23rd Canadian Conf. on Artificial Intelligence, p.28-39. [doi:10.1007/978-3-642-13059-5_6]