World Wide Web (2015) 18:235–252
DOI 10.1007/s11280-013-0225-5

Collaborative work with linear classifier and extreme learning machine for fast text categorization

Wenbin Zheng · Hong Tang · Yuntao Qian

Received: 30 August 2012 / Revised: 13 April 2013 /
Accepted: 20 May 2013 / Published online: 30 May 2013

Abstract The bloom of Internet has made fast text categorization very essential. Generally, the popular methods have good classification accuracy but slow speed, and vice versa. This paper proposes a novel approach for fast text categorization, in which a collaborative work framework based on a linear classifier and an extreme learning machine (ELM) is constructed. The linear classifier, obtained by a modified non-negative matrix factorization algorithm, maps all documents from the original term space into the class space directly such that it performs classification very fast. The ELM, with good classification accuracy via some nonlinear and linear transformations, classifies a few of documents according to some given criteria to improve the classification quality of the total system. Experimental results show that the proposed method not only achieves good accuracy but also performs classification very fast, which improves the averaged speed by 180 % compared with its corresponding method.

Keywords Fast text categorization · Extreme learning machine · Linear transformation · Non-negative matrix factorization

This work was supported by the 973 Program (No. 2012CB316400), and the National Natural Science Foundation of China (No. 11202202, No. 61272315 and No. 61171151), and the Natural Science Foundation of Zhejiang Province (No. Y6110147).

W. Zheng (✉)
College of Information Engineering,
China Jiliang University, Hangzhou, China
e-mail: zwbben@gmail.com

H. Tang
College of Metrological Technology,
China Jiliang University, Hangzhou, China

Y. Qian
College of Computer Science and Technology,
Zhejiang University, Hangzhou, China

1 Introduction

Text categorization (TC) is a task of automatically assigning predefined categories to a given text document based on its content [22]. In some applications based on TC, both the accuracy and speed are important, e.g., web retrieval [29], email classification [17], and news search [4].

A growing number of machine learning techniques have been used for TC such as linear map method [27, 31], probabilistic model [16], k-nearest neighbor [24], neural networks [2], support vector machines (SVM) [3, 12], and extreme learning machine [18, 32].

Among these methods, the linear map methods perform the classification faster than others because it assign categories directly by a simple linear transformation. Generally, such linear transformation methods could not achieve satisfactory accuracy because of the existence of linearly inseparable data [22]. In contrast, SVM has been regarded as one of the most successful methods in term of the accuracy [5, 22, 32]. However, its parameter tuning usually needs to spend a lot of time [12]. Moreover, the extension from binary classification to multiclass classification will increase additional computational cost, so classification with SVM will become quite slow if the number of categories is relatively large.

Extreme learning machine (ELM) [10] is a rapidly developed learning technology recently. It is shown that ELM could learn much fast with high generalization performance [8, 11]. In addition, ELM could be used to implement the multiclass classification [9] and the multi-label classification [32] easily because of its network output structure.

Nevertheless, text categorization based on ELM needs multiple stages [32], including: dimensionality reduction, nonlinear transformation (from input nodes into hidden nodes), linear transformation (from hidden nodes into output nodes) and category determining. Therefore, it runs slower than linear classifiers.

In order to take the advantages of the linear classifier and ELM, this paper proposes a novel approach for fast categorization. We firstly used a modified non-negative matrix factorization algorithm to obtain two transformation matrices, one is to map documents from the high dimensional term space into a low dimensional semantic subspace, the other is to map documents from the semantic subspace into the class space. With these transformations, a linear classifier was constructed, which has a very low time complexity. Furthermore, a collaborative work framework was given, in which most documents were classified via the linear classifier, only a few of documents were classified via ELM according to some given criteria such that some linearly inseparable data might be correctly discriminated by the ELM with a nonlinear activation function. Therefore, the total system could achieve a good tradeoff between the accuracy and speed.

This article is an extension of [33], the main extensions are (a) we embedded the ELM into the collaborative work framework rather than SVM to obtain more fast classification speed; (b) the new method extended the capacity to handle the multi-label situation; (c) numerous extended experiments were evaluated which verified the effectiveness and efficiency of the proposed method.

The rest of this paper is organized as follows: Section 2 provides a brief review of ELM and non-negative matrix factorization. Section 3 explains the proposed method in detail. Experimental results and analysis are given in Section 4. Finally, we summarize the conclusions in Section 5.

2 Preliminaries

2.1 Extreme learning machine

ELM is a single hidden layer feed forward networks (SLFNs) where the input weights are chosen randomly and the output weights are calculated analytically. For n arbitrary distinct samples $(\boldsymbol{x}_i, \boldsymbol{t}_i) \in \mathbb{R}^m \times \mathbb{R}^k$, the SLFNs with $\tilde{n}$ hidden nodes and activation function $g(x)$ are mathematically modeled as:

$$\boldsymbol{o}_j = \sum_{i=1}^{\tilde{n}} \boldsymbol{\beta}_i g\left(\boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{x}_j + b_i\right), \quad j = 1, \cdots, n, \tag{1}$$

where $\boldsymbol{w}_i = [w_{i1,} w_{i2,} \cdots, w_{im}]^{\mathrm{T}}$ is the weight vector connecting the ith hidden node and the input nodes, b_i is the threshold of the ith hidden node, and $\boldsymbol{\beta}_i = [\beta_{i1,} \beta_{i2,} \cdots, \beta_{ik}]^{\mathrm{T}}$ is the weight vector connecting the ith hidden node and the output nodes. If a SLFNs with $\tilde{n}$ hidden nodes can approximate these n samples with zero error (i.e. $\sum_{j=1}^{n} ||\boldsymbol{o}_j - \boldsymbol{t}_j|| = 0$), there exist $\boldsymbol{\beta}_i$, $\boldsymbol{w}_i$ and b_i such that

$$\sum_{i=1}^{\tilde{n}} \boldsymbol{\beta}_i g\left(\boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{x}_j + b_i\right) = \boldsymbol{t}_j, \quad j = 1, \cdots, n. \tag{2}$$

The above n equations can be written compactly as:

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}, \tag{3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} g\left(\boldsymbol{w}_1^{\mathrm{T}} \boldsymbol{x}_1 + b_1\right) & \cdots & g\left(\boldsymbol{w}_{\tilde{n}}^{\mathrm{T}} \boldsymbol{x}_1 + b_{\tilde{n}}\right) \\ \vdots & \cdots & \vdots \\ g\left(\boldsymbol{w}_1^{\mathrm{T}} \boldsymbol{x}_n + b_1\right) & \cdots & g\left(\boldsymbol{w}_{\tilde{n}}^{\mathrm{T}} \boldsymbol{x}_n + b_{\tilde{n}}\right) \end{bmatrix}_{n \times \tilde{n}}, \tag{4}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\beta}_{\tilde{n}}^{\mathrm{T}} \end{bmatrix}_{\tilde{n} \times k} \text{ and } \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{t}_n^{\mathrm{T}} \end{bmatrix}_{n \times k}, \tag{5}$$

$\boldsymbol{H}$ is called the hidden layer output matrix of the neural networks; the ith column of $\boldsymbol{H}$ is the ith hidden node output with respect to inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$.

Huang et al. [10, 11] proved that one may randomly choose and fix the hidden node parameters with almost any nonzero activation function and then analytically determine the output weights when approximating any continuous target function on any compact input sets. Therefore, (3) becomes a linear system and the output weights $\boldsymbol{\beta}$ are estimated as:

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger} \boldsymbol{T}, \tag{6}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore–Penrose generalized inverse of the hidden layer output matrix $\boldsymbol{H}$. Thus the output weights $\boldsymbol{\beta}$ are calculated in a single step, and this avoids any long training procedure where the network parameters are adjusted iteratively with appropriately chosen control parameters.

Huang et al. [7] also show that from the standard optimization method point of view, ELM for classification is equivalent to SVM, but ELM has less optimization constraints due to its special separability feature.

As a rapidly developed technology, numerous variations [1, 6, 19, 21, 26] and applications [25, 30, 32, 35] of ELM have emerged in recent years.

2.2 Basis orthogonal non-negative matrix factorization

In text categorization, a major difficulty is the high dimensionality of the input feature space. Non-negative matrix factorization (NMF) [14] can easily map the documents from the high dimensional term space into a low dimensional semantic subspace [23]. Mathematically, given a terms-by-documents matrix $\boldsymbol{A}$, let

$$\boldsymbol{A} \approx \boldsymbol{W}_{m\times r} \times \boldsymbol{H}_{r\times n}, \tag{7}$$

where m and n are the number of the terms and documents respectively, and r is a positive integer, $\boldsymbol{W}$ is called basis matrix and $\boldsymbol{H}$ is called coefficient matrix. Because each column vector of $\boldsymbol{W}$ is constituted with some non-negative values of terms, it can be regarded as the latent semantic basis vector, and then these basis vectors could span a semantic subspace with dimensionality r. When $r \ll \min\{m, n\}$, the dimensionality of the semantic subspace is far less than the dimensionality of the original term space.

To deal with the uniqueness problems of scaling and permutation of NMF [14], a basis orthogonal constrain could be added into the objective function of (7), i.e.,

$$\begin{cases} l(\boldsymbol{W}, \boldsymbol{H}) = ||\boldsymbol{A} - \boldsymbol{W}\boldsymbol{H}||_{\mathrm{F}}^2 + \lambda||\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} - \boldsymbol{I}||_{\mathrm{F}}^2 \\ s.t. \boldsymbol{W}, \boldsymbol{H} \geq 0 \end{cases}, \tag{8}$$

where $||\cdot||_{\mathrm{F}}$ is the Frobenius norm, $\boldsymbol{I}$ is the identity matrix, and λ is used to control the tradeoff between the approximation error and the orthogonal constraint.

Following the multiplicative update algorithm [15], the solution of (8) can be obtained [33], which is:

$$\begin{cases} \boldsymbol{W}_{i,j} \leftarrow \boldsymbol{W}_{i,j} \dfrac{\left(\boldsymbol{W}\boldsymbol{H}^{\mathrm{T}} + 2\lambda\boldsymbol{W}\right)_{i,j}}{\left(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} + 2\lambda\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}\right)_{i,j}} \\ \boldsymbol{H}_{i,j} \leftarrow \boldsymbol{H}_{i,j} \dfrac{\left(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{A}\right)_{i,j}}{\left(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{H}\right)_{i,j}} \end{cases}. \tag{9}$$

3 Collaborative work for fast text categorization

The goal of the work is to implement a framework in which most documents are classified via a very fast linear classifier, only a few linearly inseparable data are classified via ELM with a nonlinear activation function. Since ELM tends to require more hidden neurons than conventional tuning-based algorithms in many cases [34], its scale will become remarkable large if we use the high dimensional text data as input. Thus, text is firstly represented in a low dimensional semantic subspace using

a modified NMF algorithm before ELM works. With the NMF algorithm, a linear classifier that works in the original term space is also constructed, which has a low time complexity. Finally, based on two reclassified criteria, the collaborative work schema is implemented.

3.1 Text representation

Given a document $\boldsymbol{d} = (t_1, t_2, \cdots, t_m)^{\mathrm{T}}$, where m is the dimensionality in the term space. The $tfidf$ value [20] for each term is defined as:

$$tfidf(t_i, \boldsymbol{d}) = tf(t_i, \boldsymbol{d}) \times idf(t_i), \tag{10}$$

where $tf\left(t_i, \boldsymbol{d}\right)$ denotes the number of times that t_i occurred in $\boldsymbol{d}$, and $idf\,(t_i)$ is the inverse document frequency which is defined as $idf(t_i) = \log(n/df(t_i))$, where n is the number of documents in training set and $df\,(t_i)$ denotes the number of documents in training set in which t_i occurs at least once. A document can be represented as a vector:

$$\boldsymbol{d} = (w_1, w_2, \cdots, w_m)^{\mathrm{T}}, \tag{11}$$

where $w_i = tfidf(t_i, d)/\sqrt{\sum_j^m tfidf(t_j, \boldsymbol{d})^2}$.

Following with [33], we extend the document vector with its category information, and then the class vector associated with $\boldsymbol{d}$ is defined as:

$$\boldsymbol{c} = (c_1, c_2, \cdots, c_k)^{\mathrm{T}}, \tag{12}$$

where k is the number of categories in dataset, and c_i is equal to 1 or 0 depending on whether the related document belongs to the corresponding categories, e.g., assuming there are five categories ($k = 5$), a document $\boldsymbol{d} = (w_1, w_2, \cdots, w_m)^{\mathrm{T}}$ belongs to the first and the forth category, then the corresponding class vector is $\boldsymbol{c} = (1, 0, 0, 1, 0)^{\mathrm{T}}$.

We combine $\boldsymbol{d}$ and $\boldsymbol{c}$ into an extended vector $\boldsymbol{x}$:

$$\boldsymbol{x} = (w_1, w_2, \cdots, w_m, c_1, c_2, \cdots, c_k)^{\mathrm{T}}, \tag{13}$$

and then all training documents can be combined into an extended matrix $\boldsymbol{X}$, represented as

$$\boldsymbol{X} = \left[\boldsymbol{D}^{\mathrm{T}}\ \boldsymbol{C}^{\mathrm{T}}\right]^{\mathrm{T}} \tag{14}$$

where $\boldsymbol{D}$ is the weighting matrix of all training documents, $\boldsymbol{C}$ is the class matrix related to $\boldsymbol{D}$, and each column of $\boldsymbol{X}$ is an extended vector obtained by equation (13).

Assuming the matrix $\boldsymbol{X}$ is decomposed with (9) as follows:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{D}_{m\times n} \\ \boldsymbol{C}_{k\times n} \end{bmatrix} \approx \boldsymbol{W}_{(m+k)\times r} \times \boldsymbol{H}_{r\times n}, \tag{15}$$

let

$$\boldsymbol{W}_{(m+k)\times r} = \begin{pmatrix} \boldsymbol{S}_{m\times r} \\ \boldsymbol{L}_{k\times r} \end{pmatrix}, \tag{16}$$

and then

$$\begin{pmatrix} \boldsymbol{D}_{m\times n} \\ \boldsymbol{C}_{k\times n} \end{pmatrix} \approx \begin{pmatrix} \boldsymbol{S}_{m\times r} \times \boldsymbol{H}_{r\times n} \\ \boldsymbol{L}_{k\times r} \times \boldsymbol{H}_{r\times n} \end{pmatrix}. \tag{17}$$

Therefore, $\boldsymbol{D}_{m\times n} \approx \boldsymbol{S}_{m\times r} \times \boldsymbol{H}_{r\times n}$, we let the representation of documents in a low dimensional semantic subspace (r dimensionality) as:

$$\hat{\boldsymbol{H}}_{r\times n} \approx \boldsymbol{P} \times \boldsymbol{D}_{m\times n}, \tag{18}$$

here

$$\boldsymbol{P} = \text{normlize}\,(\boldsymbol{S}_{m\times r})^{\mathrm{T}}, \tag{19}$$

where normlize($\boldsymbol{S}_{m\times r}$) means cosine normalization for each column vector of ($\boldsymbol{S}_{m\times r}$). It should be noted that we does not use $\boldsymbol{P} = (\boldsymbol{S}_{m\times r})^{\dagger}$ (used in [33]) to avoid any negative value appear such that it could provide more semantic explanations for the text representation.

Consequently, $\boldsymbol{S}$ could be regarded as a transformation matrix that maps documents from the term space into a low dimensional semantic subspace spanned by $\boldsymbol{S}$, and $\hat{\boldsymbol{H}}$ is the representation of all documents in this semantic subspace.

3.2 Linear classifier construction

With (17), we have

$$\hat{\boldsymbol{C}}_{k\times n} \approx \boldsymbol{L}_{k\times r} \times \hat{\boldsymbol{H}}_{r\times n}, \tag{20}$$

where $\boldsymbol{L}_{k\times r}$ could be regarded as a transformation matrix that maps documents from the semantic subspace into the class space, and $\hat{\boldsymbol{C}}_{k\times n}$ means the projection of $\hat{\boldsymbol{H}}_{r\times n}$ in the class space.

When the transformation matrices $\boldsymbol{P}$ and $\boldsymbol{L}$ are obtained, we can map documents directly from the term space into the class space, i.e.,

$$\hat{\boldsymbol{C}} \approx \boldsymbol{T} \times \boldsymbol{D}, \tag{21}$$

where $\boldsymbol{T} = \boldsymbol{L} \times \boldsymbol{P}$.

Given a test document $\boldsymbol{d}$, its category coding $\hat{\boldsymbol{c}} = (\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_k)^{\mathrm{T}}$ can be obtained directly with (21). Ideally, if $\boldsymbol{d}$ belongs to some categories, its corresponding bits of $\hat{\boldsymbol{c}}$ should be equal to 1, and 0 otherwise. However, this situation is unpractical because of the computation error and the existence of linearly inseparable data. In spite of this, the values of $\hat{\boldsymbol{c}}$ still imply some meaningful information of category. Normalizing $\hat{\boldsymbol{c}}$ as follows:

$$\bar{\boldsymbol{c}} = \left(\bar{c}_1, \bar{c}_2, \cdots, \bar{c}_k\right)^{\mathrm{T}}, \tag{22}$$

where $\bar{c}_i = \hat{c}_i / \sqrt{\sum_j^k \hat{c}_j^2}$. According to the uni-label corpus (i.e., a document can only be assigned to a unique category in this corpus), a simple intuition is that if a

document belongs to a category, its corresponding bit of $\bar{\boldsymbol{c}}$ should be larger than others. According to the multi-label corpus, a simple intuition is that if a document belongs to some categories, its corresponding bits of $\bar{\boldsymbol{c}}$ should be relatively large. Therefore, we define the category discrimination function as:

$$Category(\boldsymbol{d}) = \begin{cases} \arg\max_i(\bar{c}_i) \text{ for uni-label corpus} \\ \arg_i(\bar{c}_i > \tau) \text{ for multi-label corpus} \end{cases}, \tag{23}$$

where τ is a threshold parameter.

Figure 1 gives an illustration about the linear classifier, its classification procedure has a very low time complexity $\mathcal{O}(k \times m)$.

3.3 Collaborative classification based on linear classifier and ELM

With (21) and (23), the linear classifier could be implemented. However, such directly map method usually could not achieve satisfactory accuracy because of the numeral error and the linearly inseparability of some data. Since ELM has high classification accuracy in a low dimensional semantic subspace with the nonlinear activation functions [32], we then use it to classify the documents that are easily misclassified by our linear classifier. The motivation is that we can take the advantages of our linear classifier and ELM to achieve a good tradeoff between accuracy and classification speed.

A key issue of this collaborative work framework is how to judge which documents are easily to be misclassified by the linear classifier. Our work is based on following intuitions:

- In a space where data cannot be linearly separated completely, the points closing to the hyperplane are usually to be misclassified because of the nonlinear distribution.
- In a space where data can be linearly separated completely, the points closing to the hyperplane are usually to be misclassified because of the numeral computation error.

Figure 2 shows an illustration about which points are easy to be misclassified. In this figure, points could be well separated via a nonlinear boundary (denoted by the

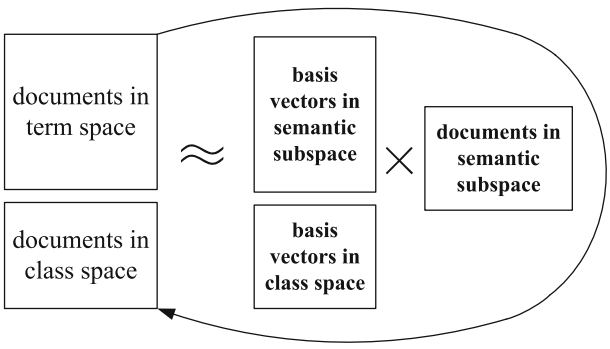


Figure 1 An illustration of the linear classifier.

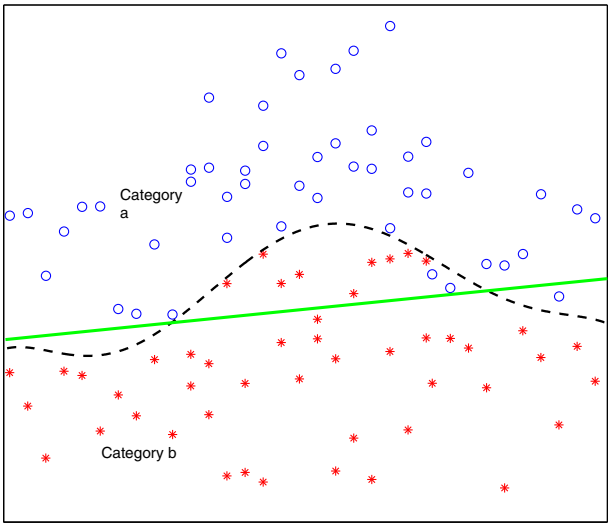


Figure 2 Illustration of the points that are easy to be misclassified (the *solid line* denotes a hyperplane that separates category *a* and *b*, and the *dash line* denotes an ideal separating boundary)

dash line), but some points that are close to the category boundary are misclassified by the hyperplane (denoted by the solid line). Moreover, some points, even be correctly separated by the hyperplane, are also easy to be misclassified due to the computational error when they are very close to the hyperplane. Mathematically, if a point is closes to the hyperplane, its corresponding class vector might contain two or more elements with relative large values. Let secondmax($\bar{\boldsymbol{c}}$) denotes the second large value in class vector $\bar{\boldsymbol{c}}$ of a document $\boldsymbol{d}$, we define a threshold function as follows:

$$\text{threshold}(\boldsymbol{d}) = \frac{\text{secondmax}(\bar{\boldsymbol{c}})}{\max(\bar{\boldsymbol{c}})}. \tag{24}$$

For the uni-label corpus, a document $\boldsymbol{d}$ needs to be reclassified when it follows the criterion as:

$$\text{threshold}(\boldsymbol{d}) > \theta, \tag{25}$$

where θ is used to control the number of reclassified documents. It is worth to note that when $\theta = 1$, the system only performs our linear classifier, and when $\theta = 0$, all results are given by ELM.

For the multi-label corpus, we can not use (25) to judge whether a document is easy to be misclassified, since a document might belong to several categories (i.e., two or more relative large values in the class vector are permitted). Alternatively, we define a reclassified criterion as follows:

$$\max(\bar{c}_i) < \sigma \tag{26}$$

where $\sigma \leq 1$ is a threshold parameter that has a similar function with θ. The criterion is based on an intuitional assumption: when a document satisfies (26), we think that a relatively large computational error might occurs (otherwise there is at least one relatively large value in its class vector), so the classification result with (23) becomes unreliable, i.e., it needs to be reclassified. Note that the system only performs our linear classifier if $\sigma = 0$, and all results are given by ELM if $\sigma = 1$.

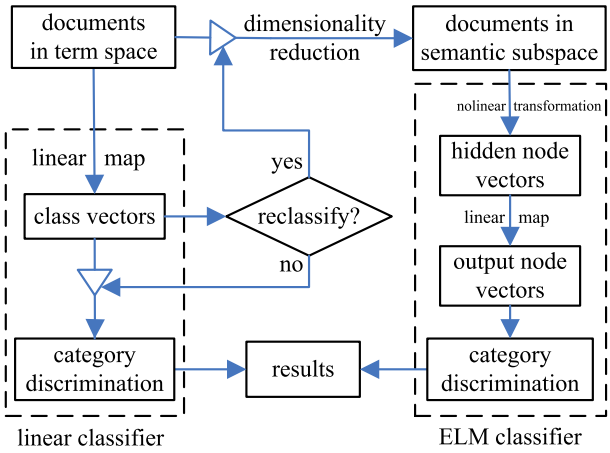


Figure 3 An illustration of the collaborative framework for classification

Thus, for a test document $\boldsymbol{d}$ that needs to be reclassified, its output target vector of ELM can be evaluated as:

$$\boldsymbol{o} = \tilde{\boldsymbol{h}}\hat{\boldsymbol{\beta}}, \tag{27}$$

where $\tilde{\boldsymbol{h}}$ is the hidden layer output vector of $\boldsymbol{d}$, and $\hat{\boldsymbol{\beta}}$ is the output weights learned by training set. Following [32], the discrimination function is:

$$Category(\boldsymbol{d}) = \begin{cases} \arg\max_i(o_i) \text{ for uni-label corpus} \\ \arg_i(o_i > \rho) \text{ for multi-label corpus} \end{cases}, \tag{28}$$

where ρ is a threshold parameter.

Figure 3 illustrates the collaborative framework for classification. If we suppose that the time complexity of the classification procedure with ELM in a r-dimensionality semantic subspace is $\phi(r)$, and the probability of reclassifying is p, then the time complexity of the collaborative classification procedure is $\mathcal{O}(k \times m + p \times (\phi(r) + r \times m))$. Hence, if p is small, the total classification procedure still performs very fast.

4 Experiments

For convenience, we denote our method as LELM (**L**inear classifier & ELM) and compared it with three related methods: ELM [32], SVM [13], and LSVM [33] (**L**inear classifier & SVM), here we extended [33] to multi-label case for comparative need.

We set the discrimination threshold parameters $\tau = 0.5$ (in (23)), $\rho = 0.5$ (in (28)), where 0.5 is the midpoint between 0 and 1 (corresponding to class label). λ (in (9)) was simply set 0.5 for computational convenience. For ELM, the radial basis function was selected as the activation functions, the number of hidden nodes was set to 2,000, and its regularized parameter was set to 5. For SVM, its parameters were obtained by 4-fold cross-validation, and the same values were used in LSVM.

Other parameters in the experimentation will be discussed in the result part, and all programs were run on Matlab 2012a with 3.3 GHz CPU and 6 GB memory.

4.1 Datasets

Three popular TC benchmarks are tested in our experiments: Reuters-21578, WebKB, and 20-Newsgroups. The Reuters-21578 dataset[1] is a standard multi-label TC benchmark which contains 135 categories. In our experiments, we use a subset called Reuters-top10 that includes the ten most frequent categories among the 135 topics. The subset was divided into the training and test set with the standard "ModApte" version. The pre-processed procedure includes: removing the stop words, switching upper case to lower case, stemming,[2] and removing the low frequency words (less than three). After that, 5,920 training documents and 2,315 testing documents with 5,585 term features are obtained.

The WebKB dataset is a standard uni-label TC benchmark which contains web pages gathered from university computer science departments. A subset[3] including four most populous entity-representing categories was used in our experimentation. After pre-processed procedure, 2,777 training documents and 1,376 testing documents with 7,287 term features are obtained.

The 20-Newsgroups dataset[4] is also a uni-label TC benchmark which contains approximately 20,000 articles evenly divided among 20 usenet newsgroups. After pre-processed procedure, 11,269 training documents and 7,505 testing documents with 31,116 term features are obtained.

4.2 Evaluation measures

In text categorization, the most commonly used accuracy measures are recall, precision and their harmonic mean F_1. For multiple categories, we use the micro-averaged F_1 [22] to evaluate the total performance and denoted it as mF_1 for briefness.

We also define a measure to evaluate the classification speed, which is defined as follows:

$$tpd = \frac{elapsedTime}{\#test}, \tag{29}$$

where $elapsedTime$ denotes the time cost during the classification stage, and $\#test$ denotes the number of documents in test set, thus tpd means the averaged classification time for each test document (**t**ime **p**er **d**ocument). We use microsecond per document (μs/d) as the unit of tpd.

4.3 Results

Figure 4 gives a result of LELM on Reuters-top10 where the dimensionality in the semantic subspace was set to 50. Here, the parameter σ was changed and its

[1] http://www.daviddlewis.com/resources/

[2] http://tartarus.org/~martin/PorterStemmer/

[3] http://web.ist.utl.pt/~acardoso/datasets/

[4] http://people.csail.mit.edu/jrennie/

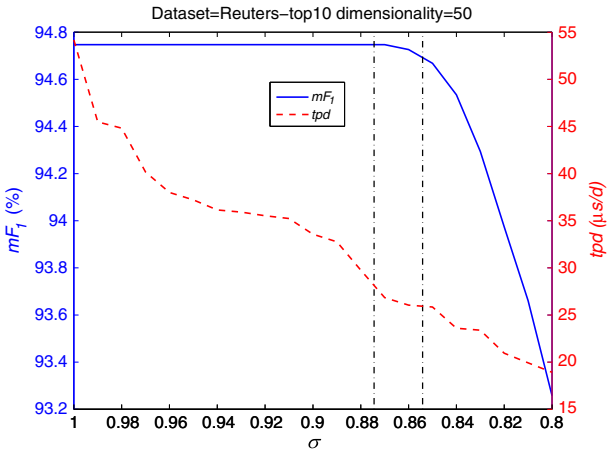


Figure 4 Accuracy vs. speed on Reuters-top10 (the interval bounded by *two dash-dot line* indicates the ideal range for parameter σ)

corresponding mF_1 and *tpd* were recorded. It can be observed that the time cost (plotted by the dashed line) descends as σ descends, while the accuracy (plotted by the solid line) almost has no variation until σ achieves the left dash-dot line. The interval bounded by two dash-dot line indicates the ideal range for parameter σ, where *tpd* reduces to about 50-30% but the corresponding mF_1 almost has no change.

Figures 5 and 6 give the results of LELM on WebKB (dimensionality = 70) and 20-Newsgroups (dimensionality = 210), respectively. Similarly, the time cost descends as the parameter θ increases, while the accuracy almost has no variation until θ achieves the left dash-dot line. The best tradeoff occurs when the parameter falls into the interval bounded by two dash-dot line. In this situation, the accuracy has little loss but the elapsing time declines dramatically.

Table 1 presents more results on Reuters-top10, where the Dim means the dimensionality in the semantic subspace. For each dimensionality, σ was selected by 4-fold cross-validation on training set such that the value of $(mF_1 - \omega * \overline{tpd})$ achieves maximum, where $\overline{tpd}$ is the normalization of *tpd* for all σ, and ω is a

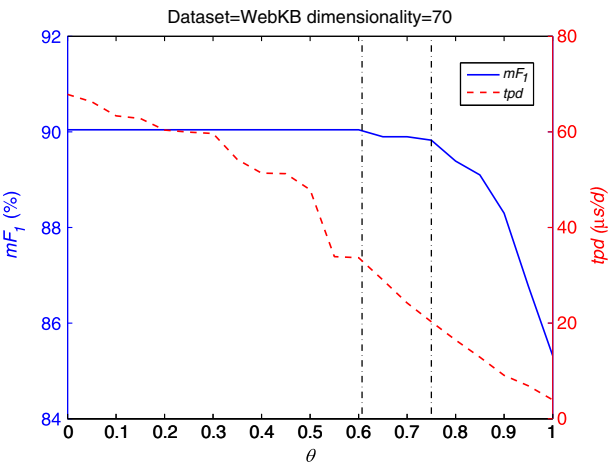


Figure 5 Accuracy vs. speed on WebKB (the interval bounded by *two dash-dot line* indicates the ideal range for parameter θ)

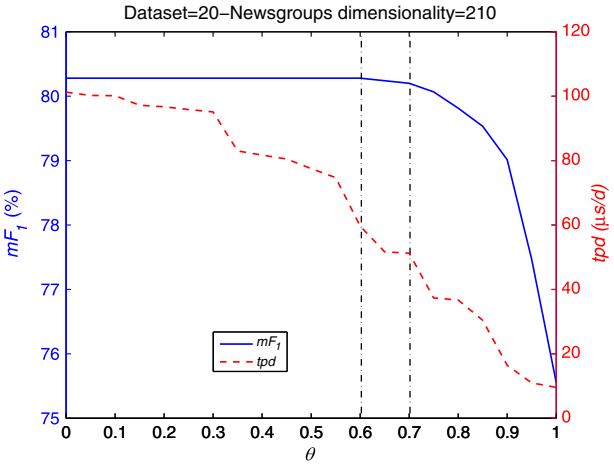


Figure 6 Accuracy vs. speed on 20-Newsgroups (the interval bounded by *two dash-dot line* indicates the ideal range for parameter θ)

weighting that is used to control the tradeoff between accuracy and speed, it was set a small number (0.02) experimentally, since the main aim of parameter selection is to increase speed while accuracy could be ensured. From this table, we can observed that the SVM, LSVM, ELM, and our LELM almost have completely same accuracy (their accuracy difference according to each dimensionality was equal to zero at the 5 % significance level when T-test was performed). Moreover, LELM runs very fast than its competitors. In most cases, LELM performs two times faster than ELM, and hundreds of time faster than SVM.

More results on WebKB and 20-Newsgroups are given in Tables 2 and 3, respectively. For each dimensionality, the parameter θ is obtained using a similar method as selecting σ. Based on the two tables, T-test was performed. The accuracy differences between LELM and ELM (or between LSVM and SVM) were less than 0.25 % at the 5 % significance level. It is clearly shown that LELM not only has a comparable accuracy with other methods but also has a very fast classification speed. In sum, the averaged speed of LELM increased by 180 % (compared with its corresponding method) without loss of accuracy.

Table 1 Accuracy and speed on Reuters-top10

Dim	mF_1 (%)				*tpd* (µs/d)			
	SVM	LSVM	ELM	LELM	SVM	LSVM	ELM	LELM
30	94.41	94.34	**94.91**	94.87	1,729	797	57	**25**
50	94.65	94.59	**94.75**	94.67	2,352	1,137	60	**23**
70	94.74	94.63	**94.99**	94.90	2,954	1,320	61	**25**
90	94.83	94.64	**94.88**	94.70	3,587	1,505	65	**23**
110	94.62	94.41	**94.81**	94.68	4,225	1,695	73	**27**
130	94.93	94.90	94.92	**95.05**	4,793	1,960	77	**25**
150	94.56	94.33	**94.65**	94.54	5,540	2,094	88	**27**
170	94.82	94.75	94.82	**94.85**	6,217	2,402	90	**30**
190	94.91	94.92	94.90	**94.99**	6,929	2,753	93	**29**
210	**94.71**	94.66	94.57	94.59	7,581	3,298	95	**37**

The bold items denote the best performance (accuracy or speed) for each dimensionality

Table 2 Accuracy and speed on WebKB

Dim	mF_1 (%)				*tpd* (μs/d)			
	SVM	LSVM	ELM	LELM	SVM	LSVM	ELM	LELM
30	89.24	89.24	89.68	**89.75**	1,093	687	51	**20**
50	89.03	88.88	**90.04**	89.75	1,484	531	53	**20**
70	90.12	**90.12**	90.04	89.83	1,944	932	53	**20**
90	90.26	**90.26**	90.12	89.90	2,454	1,081	66	**19**
110	89.97	89.90	89.90	**89.97**	2,911	1,141	71	**28**
130	**89.97**	89.90	89.32	89.89	3,407	1,671	74	**27**
150	**89.75**	89.46	89.68	89.68	3,959	1,649	74	**30**
170	89.83	**89.83**	88.74	89.30	4,450	2,113	80	**22**
190	89.83	**89.90**	88.81	88.32	4,970	2,908	87	**28**
210	**89.17**	88.81	88.30	88.59	5,550	1,891	91	**22**

The bold items denote the best performance (accuracy or speed) for each dimensionality

From the results above, we could find that LELM achieved better accuracy than ELM in some cases. It means that the combination with linear and nonlinear transformations might obtain some extra advantages for classification, which is very interesting.

Therefore, an important problem is which nonlinear transformation should be used in our correlative work framework. In other words, which activation functions should be selected?

Figure 7 plots five commonly used activation functions: "sine" "sigmoid" "radial basis function" (abbreviated as "radial"), "hard limit" (abbreviated as "hardlim"), and "triangular basis function" (abbreviated as "tribas"). The accuracy comparisons about these activation functions on Reuters-top10, WebKB, and 20-Newsgroup are given in Figures 8, 9, and 10 respectively. For each dimensionality, parameter σ (or θ) was selected as mentioned above.

From these figures, we can observe that the accuracy with activation function "hardlim" is very poor. The results could be understood easily, since the "hardlim" function actually does not take the advantage of nonlinear transformation, i.e., it can not map the linearly inseparable data into a linearly separable space. By contrast, the "radial" function, with a axial symmetry and smooth curve, performs well in most cases. However, as is well-known, the "radial" function has two parameters

Table 3 Accuracy and speed on 20-Newsgroups

Dim	mF_1 (%)				*tpd* (μs/d)			
	SVM	LSVM	ELM	LELM	SVM	LSVM	ELM	LELM
30	78.71	78.69	**79.08**	79.04	2,920	888	47	**17**
50	78.97	78.77	**79.23**	79.12	3,729	1,153	54	**16**
70	79.76	79.51	80.03	**80.06**	4,601	1,348	60	**16**
90	79.45	79.20	79.64	**79.65**	5,492	1,900	62	**25**
110	79.76	79.60	**80.23**	80.15	6,548	3,275	66	**35**
130	79.91	79.77	**80.01**	79.77	7,555	3,388	79	**32**
150	79.69	79.64	**80.09**	80.01	8,500	4,023	84	**30**
170	79.85	79.83	80.10	**80.11**	9,530	5,603	84	**42**
190	79.97	79.93	**80.41**	80.37	10,554	5,222	87	**41**
210	79.71	79.56	**80.28**	80.07	11,590	5,224	99	**37**

The bold items denote the best performance (accuracy or speed) for each dimensionality

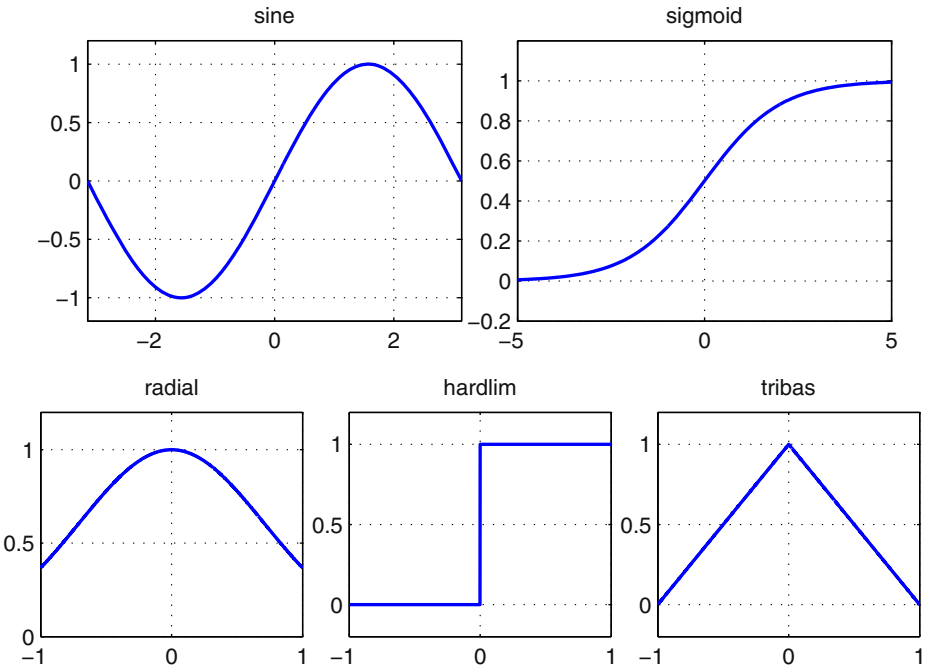


Figure 7 Shapes of five commonly used activation functions of ELM

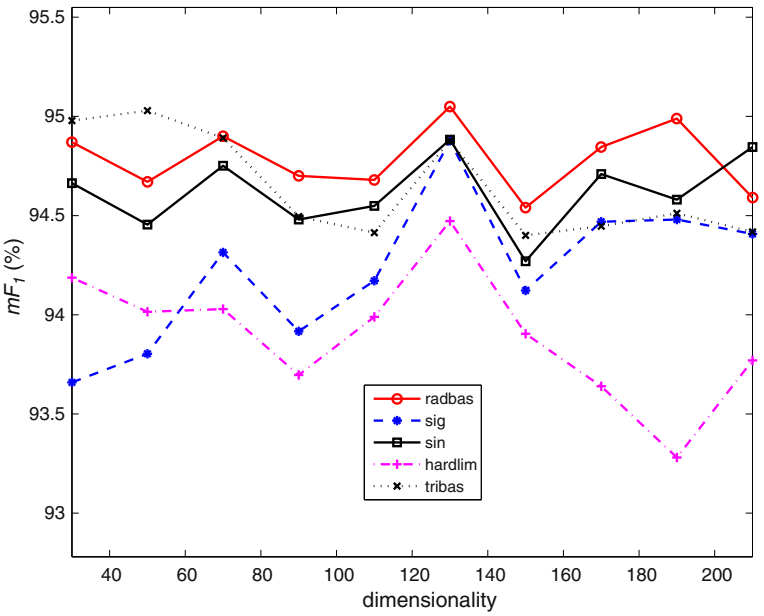


Figure 8 Accuracy comparisons of activation functions on Reuters-top10

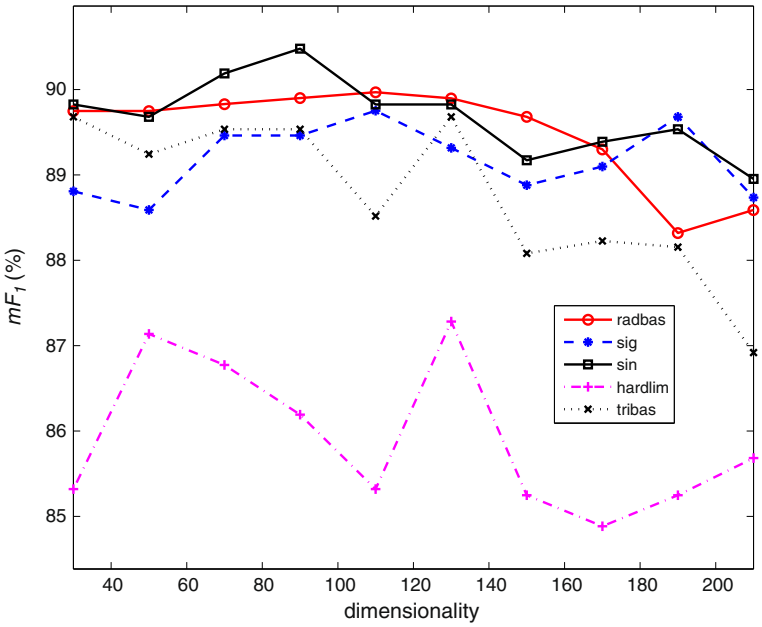


Figure 9 Accuracy comparisons of activation functions on WebKB

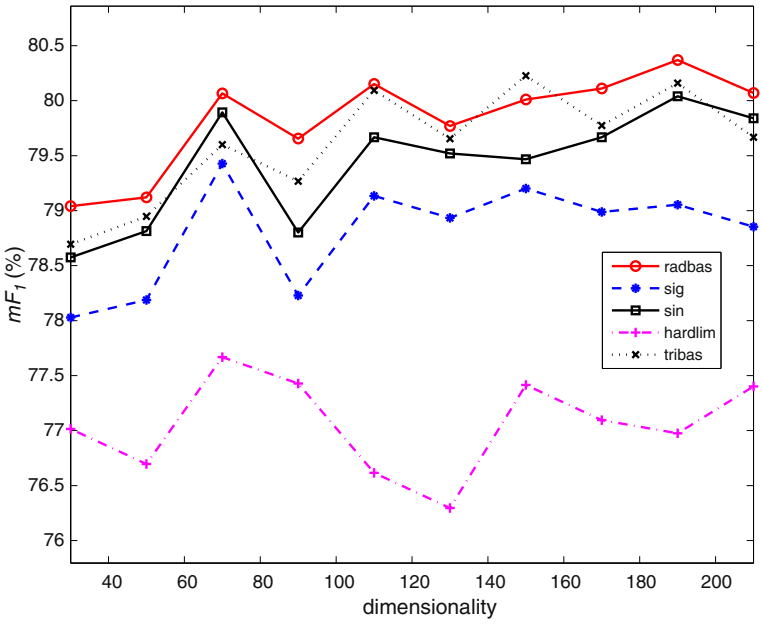


Figure 10 Accuracy comparisons of activation functions on 20-Newsgroups

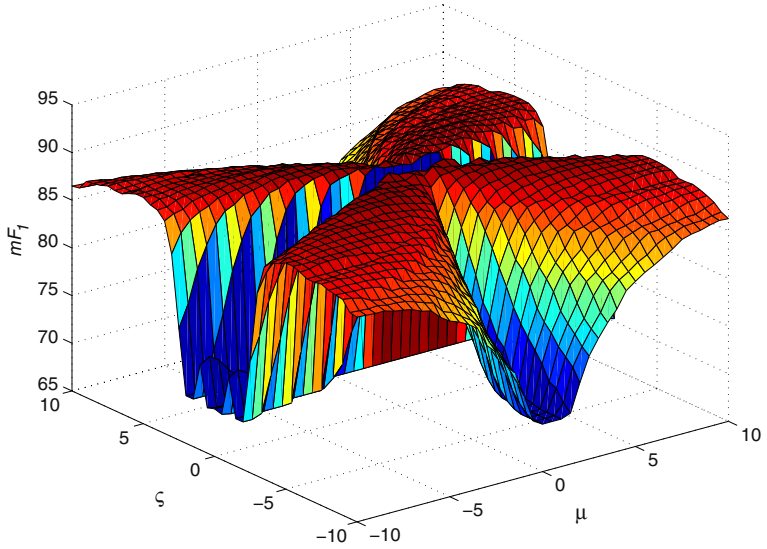
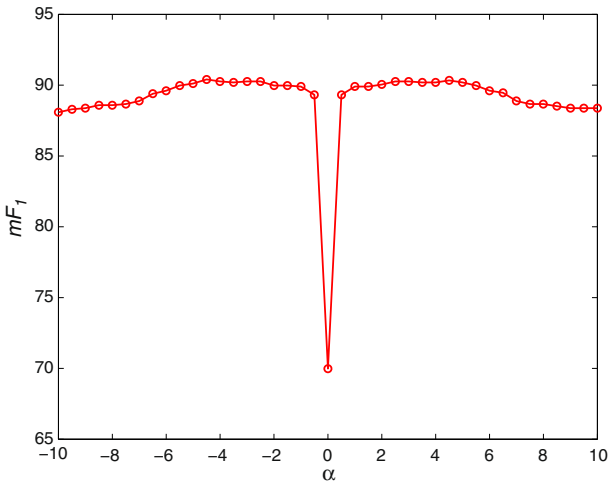


Figure 11 A parameter tuned result while using "radial" function

(i.e. center and width), so it is rather difficult to tune the two parameters. Figure 11 shows a parameter tuned result on WebKB (dimensionality = 110), where μ means the center, ς controls the width, and other parameters were fixed as mentioned above. From the figure, we can see that the two parameters are both important for the accuracy when the "radial" function is utilized.

On the other hand, Sigmoidal activation function has only one parameter to tune. Figure 12 gives its parameter tuned result on WebKB (dimensionality = 110), where α control the shape of the function, and other parameters were fixed as mentioned above. It shows that the more simple tuned way might obtain comparable accuracy.

Figure 12 A parameter tuned result while using "sigmoid" function



More similar results can be observed when the dimensionality was set to other values or other datasets were used. Thus, an experimental suggestion is that the Sigmoidal function should also be considered as a choice as activation function in text categorization. Of course, different data distribution needs different nonlinear transformation, the choice of activation functions is still an open problem. The reader may refer to [28] for localized generalization error analysis of architecture selection.

5 Conclusions

This paper proposes a collaborative work method for fast text categorization. With a modified non-negative matrix factorization algorithm, two transformation matrices are obtained, one is to map documents from the high dimensional term space into a low dimensional semantic subspace, the other is to map documents from the semantic subspace into the class space. Based on these transformations, a linear classifier is constructed, which has a very low time complexity. After that, most documents are classified via the linear classifier, only a few of documents are classified via ELM according to some given criteria. Extensive experimental results have shown that the proposed approach achieves a good tradeoff between the accuracy and classification speed. Especially, the collaborative work method obtains the best performance in some cases, which implies some advantages about the combination of linear and nonlinear transformations for classification.

For further study, we are trying to analyze the localized generalization error bound of ELM and focus on the comparison of the prediction accuracy between ELM and rule-based systems together with their refinements.

Acknowledgements The authors are grateful to anonymous reviewers for their constructive suggestions.

References

1. Cao, J., Lin, Z., Huang, G.B., Liu, N.: Voting based extreme learning machine. Inf. Sci. **185**(1), 66–77 (2012)
2. De Souza, A., Pedroni, F., Oliveira, E., Ciarelli, P., Henrique, W., Veronese, L., Badue, C.: Automated multi-label text categorization with vg-ram weightless neural networks. Neurocomputing **72**(10–12), 2209–2217 (2009)
3. Gabrilovich, E., Markovitch, S.: Text categorization with many redundant features: using aggressive feature selection to make svms competitive with c4. 5. In: International Conference on Machine Learning, pp. 321–328 (2004)
4. Henzinger, M., Chang, B.W., Milch, B., Brin, S.: Query-free news search. World Wide Web **8**(2), 101–126 (2005)
5. Hmeidi, I., Hawashin, B., El-Qawasmeh, E.: Performance of knn and svm classifiers on full word arabic articles. Adv. Eng. Inform. **22**(1), 106–111 (2008)
6. Huang, G., Chen, L.: Convex incremental extreme learning machine. Neurocomputing **70**(16–18), 3056–3062 (2007)
7. Huang, G., Ding, X., Zhou, H.: Optimization method based extreme learning machine for classification. Neurocomputing **74**, 155–163 (2010)
8. Huang, G., Wang, D., Lan, Y.: Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. **2**(2), 107–122 (2011)
9. Huang, G., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. Part B **42**(2), 513–529 (2011)

10. Huang, G., Zhu, Q., Siew, C.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the International Joint Conference on Neural Networks, vol. 2, pp. 985–990 (2004)
11. Huang, G., Zhu, Q., Siew, C.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
12. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: 10th European Conference on Machine Learning, pp. 137–142 (1998)
13. Kumar, M.A., Gopal, M.: A comparison study on multiple binary-class svm methods for unilabel text categorization. Pattern Recogn. Lett. **31**(11), 1437–1444 (2010)
14. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
15. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems, pp. 556–562 (2001)
16. Lewis, D., Ringuette, M.: A comparison of two learning algorithms for text categorization. In: Third Annual Symposium on Document Analysis and Information Retrieval, vol. 33, pp. 81–93 (1994)
17. Li, W., Zhong, N., Yao, Y., Liu, J.: An operable email based intelligent personal assistant. World Wide Web **12**(2), 125–147 (2009)
18. Liu, Y., Loh, H., Tor, S.: Comparison of extreme learning machine with support vector machine for text classification. Innov. Appl. Artif. Intell. **3533**, 390–399 (2005)
19. Man, Z., Lee, K., Wang, D., Cao, Z., Khoo, S.: Robust single-hidden layer feedforward network-based pattern classifier. IEEE Trans. Neural Netw. Learning Syst. **23**(12), 1974–1986 (2012)
20. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. **24**(5), 513–523 (1988)
21. Savitha, R., Suresh, S., Sundararajan, N.: Fast learning circular complex-valued extreme learning machine (cc-elm) for real-valued classification problems. Inf. Sci. **187**(1), 277–290 (2012)
22. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**(1), 1–47 (2002)
23. Silva, C., Ribeiro, B.: Knowledge extraction with non-negative matrix factorization for text classification. In: Proceedings of Intelligent Data Engineering and Automated Learning, vol. 5788, pp. 300–308 (2009)
24. Soucy, P., Mineau, G.: A simple knn algorithm for text categorization. In: International Conference on Data Mining, pp. 647–648 (2001)
25. Wang, X., Chen, A., Feng, H.: Upper integral network with extreme learning mechanism. Neurocomputing **74**(16), 2520–2525 (2011)
26. Xing, H.J., Wang, X.M.: Training extreme learning machine via regularized correntropy criterion. Neural Comput. Appl. (2012). doi:10.1007/s00521-012-1184-y
27. Yang, Y., Chute, C.: An example-based mapping method for text categorization and retrieval. ACM Trans. Inf. Syst. **12**(3), 252–277 (1994)
28. Yeung, D.S., Ng, W.W., Wang, D., Tsang, E.C., Wang, X.Z.: Localized generalization error model and its application to architecture selection for radial basis function neural network. IEEE Trans. Neural Netw. **18**(5), 1294–1305 (2007)
29. Zakos, J., Verma, B.: A novel context-based technique for web information retrieval. World Wide Web **9**(4), 485–503 (2006)
30. Zhang, R., Huang, G., Sundararajan, N., Saratchandran, P.: Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis. IEEE/ACM Trans. Comput. Biol. Bioinf. **4**(3), 485–495 (2007)
31. Zhang, T., Oles, F.: Text categorization based on regularized linear classification methods. Inf. Retr. **4**, 5–31 (2001)
32. Zheng, W., Qian, Y., Lu, H.: Text categorization based on regularization extreme learning machine. Neural Comput. Appl. **22**(3), 447–456 (2013)
33. Zheng, W., Zhang, H., Qian, Y.: Fast text categorization based on collaborative work in the semantic and class spaces. In: International Conference on Machine Learning and Cybernetics, vol. 4, pp. 1497–1502 (2011)
34. Zhu, Q., Qin, A., Suganthan, P., Huang, G.: Evolutionary extreme learning machine. Pattern Recogn. **38**(10), 1759–1763 (2005)
35. Zong, W., Huang, G.B.: Face recognition based on extreme learning machine. Neurocomputing **74**(16), 2541–2551 (2011)