Neural Comput & Applic (2013) 22:447–456
DOI 10.1007/s00521-011-0808-y

Text categorization based on regularization extreme learning machine

Wenbin Zheng · Yuntao Qian · Huijuan Lu

Received: 8 August 2011 / Accepted: 30 December 2011 / Published online: 12 January 2012

Abstract This article proposes a novel approach for text categorization based on a regularization extreme learning machine (RELM) in which its weights can be obtained analytically, and a bias-variance trade-off could be achieved by adding a regularization term into the linear system of single-hidden layer feedforward neural networks. To fit the input scale of RELM, the latent semantic analysis was used to represent text for dimensionality reduction. Moreover, a classification algorithm based on RELM was developed including the uni-label (i.e., a document can only be assigned to a unique category) and multi-label (i.e., a document can be assigned to multiple categories simultaneously) situations. The experimental results in two benchmarks show that the proposed method can produce good performance in most cases, and it could learn faster than popular methods such as feedforward neural networks or support vector machine.

Keywords Text categorization · Extreme learning machine · Support vector machine · Latent semantic analysis · Regularization

W. Zheng · Y. Qian (✉)
College of Computer Science and Technology,
Zhejiang University, Hangzhou 310027, China
e-mail: ytqian@zju.edu.cn; zjucsai@gmail.com

W. Zheng
e-mail: zwb@zju.edu.cn

W. Zheng · H. Lu
College of Information Engineering, China Jiliang University,
Hangzhou 310018, China

1 Introduction

Text categorization (TC) is a task of automatically assigning predefined categories to a given text document based on its content [1]. A growing number of machine learning techniques have been used for TC such as probabilistic model [2], k-nearest neighbor (KNN) [3], neural networks [4–6], support vector machines (SVM) [7, 8], and logistic regression [9, 10].

Among above methods, SVM has been regarded as one of the most successful methods in TC [1, 8, 11]. However, SVM has some disadvantages, for example, learning its parameters usually needs to spend a lot of time [7, 12]. Moreover, extending learning algorithms from binary classification to multi-classification will increase the computational cost.

Neural network is also an efficient and popular approach for TC, with which multiclass classification could be implemented easily [13]. Generally, the free parameters of the neural networks are learnt via gradient descent algorithms [14], which are relatively slow and have many issues related to its convergence such as stopping criteria, learning rate, learning epochs, and local minima.

Recently, Huang et al. [15] proposed a novel learning algorithm for single-hidden layer feedforward neural networks called extreme learning machine (ELM). It is shown that ELM not only learns much faster with higher generalization performance than the traditional gradient-based neural network methods but also avoids the convergence difficulties mentioned above [16, 17]. However, a potential disadvantage of ELM is that it tends to require more hidden neurons than conventional tuning-based algorithms in many cases [18], so its scale will become remarkable large if the input dimensionality is rather high such as text data.

In this article, we propose a novel approach based on a regularization extreme learning machine (RELM) for TC. Firstly, the latent semantic analysis (LSA) [19] was used to obtain a semantic representation of text and reduce the dimensionality to fit the input scale of ELM. Nextly, a regularization term was added into the linear system of ELM to construct a RELM in which its output weights were obtained analytically. The aim of adding regularization term is that we wish the RELM could overcome the overfitting problem, since the dimensionality in semantic space might still be high after using LSA, which may result in a low bias but large variance of the estimated weights. Finally, a TC algorithm was developed including uni-label and multi-label situations.

The major contributions of this article are as follows: (1) introducing a regularization term into the linear system of ELM, meanwhile, its analytical solution and theoretical proof are presented; (2) proposing a framework combining the LSA and RELM for TC (including uni-label and multi-label cases); (3) giving some experimental suggestions about parameter selection for TC based on RELM.

The rest of this article is organized as follows. Section 2 introduces the preliminaries and related works. Section 3 explains the proposed method in detail. Experimental results and analysis are shown in Sect. 4. Finally, we summarize the conclusions in Sect. 5.

2 Preliminaries and related works

A brief review of ELM is presented and the related works about neural networks for TC are introduced.

2.1 A review of 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 $(x_i, t_i) \in \mathbb{R}^k \times \mathbb{R}^m$, the SLFNs with $\tilde{N}$ hidden nodes and activation function $g(x)$ are mathematically modeled as

$$o_j = \sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i), \quad j = 1, \ldots, N, \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, \ldots, w_{ik}]^{\mathrm{T}}$ is the weight vector connecting the ith hidden node and the input nodes, $w_i \cdot x_j$ denotes the inner product of w_i and x_j, b_i is the threshold of the ith hidden node, and $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^{\mathrm{T}}$ 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. $\sum_{j=1}^{\tilde{N}} ||o_j - t_j|| = 0$), there exist β_i, w_i and b_i such that

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = t_j, \quad j = 1, \ldots, N. \tag{2}$$

The above N equations can be written compactly as

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

where

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^{\mathrm{T}} \\ \vdots \\ \beta_{\tilde{N}}^{\mathrm{T}} \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad T = \begin{bmatrix} t_1^{\mathrm{T}} \\ \vdots \\ t_{\tilde{N}}^{\mathrm{T}} \end{bmatrix}_{N \times m}, \tag{5}$$

H is called the hidden layer output matrix of the neural networks; the ith column of H is the ith hidden node output with respect to inputs $x_1, x_2, \ldots, x_N$.

Huang et al. [15, 16] 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 β are estimated as

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

where $H^{\dagger}$ is the Moore–Penrose generalized inverse of the hidden layer output matrix H. Thus, the output weights β 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. [20] 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.

2.2 Neural networks for text categorization

Since several years ago, neural networks have been applied to TC tasks. In [4], Ng et al. used the perceptrons to construct a text classifier and reported a surprisingly high performance. Moreover, multilayer perceptron method was used for subject categorization [21] or authorship attribution classification [22]. To overcome the high dimensionality problem, Wang and Yu [5] introduced a combination of modified back propagation neural network (BP) and LSA, they used LSA to map the high dimensional term space into a low dimensional semantic space, so the dimensionality was reduced dramatically, and the performance was reported to be improved. However, the learning

process is quite slow because of the convergence issues. In [23], Liu et al. introduced ELM for TC and reported the performance comparison with SVM, they pointed out experimentally that SVM still outperforms ELM in terms of F_1 value (see 4.2 for the definition of F_1) even the ELM has higher accuracy. Nevertheless, they did not introduce the multi-label case of TC, and the learning and classification time were not mentioned. Different from that, our RELM method tends to have better generalization performance due to the regularization constrain, and our classification algorithm can deal with the uni-label or multi-label cases for TC.

3 Text categorization based on regularization extreme learning machine

Generally, TC based on machine learning techniques consists three parts: text representation method, classification algorithm and performance evaluation. LSA is a classical text representation method, which could not only greatly reduces the dimensionality but also discovers the important associative relationship between terms [19]. Thus, LSA is used to project the original high dimensional term vectors into the low dimensional semantic vectors for text representation. Next, a regularization extreme learning machine (RELM) and its solution are presented. Finally, a TC algorithm based on RELM is developed.

3.1 Representation of text

Given a document $d = (t_1, t_2, \ldots, t_n)^{\mathrm{T}}$, where n is the dimensionality in the term space. The *tfidf* value [24] for each term is defined as:

$$tfidf(t_i, d) = tf(t_i, d) \times idf(t_i), \tag{7}$$

where $tf(t_i, d)$ denotes the number of times that t_i occurred in 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. Then a document can be represented as a vector:

$$d = (w_1, w_2, \ldots, w_n)^{\mathrm{T}}, \tag{8}$$

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

Here, all document vectors are combined as a term by document matrix $D \in \mathbb{R}^{n \times N}$, where n and N are commonly considerable large. Generally, the matrix D is not suitable to be used as input for ELM directly. With a singular value decomposition: $D = U \times \Sigma \times V^{\mathrm{T}}$, where U and V are two orthogonal matrices and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is the diagonal matrix of singular values. The best approximation of D with rank-k matrix is $D_k = U_k \times \Sigma_k \times V_k^{\mathrm{T}}$, where U_k is comprised of the first k columns of the matrix U and V_k^{T} is comprised of the first k rows of matrix V^{T} corresponding to the largest k singular values, which form the diagonal matrix $\Sigma_k = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$. Thus the matrix D_k captures most of the important latent semantic of the term by document matrix D. Consequently, a document vector $d = (w_1, w_2, \ldots, w_n)^{\mathrm{T}}$ can be projected from the term space into the k-dimensional semantic space and represented by

$$\hat{d} = d^{\mathrm{T}} U_k \Sigma_k^{-1}. \tag{9}$$

Therefore, the dimensionality is reduced from n to k, and all of the training and test examples could be represented by this way.

3.2 Regularization extreme learning machine

Assuming $X \in \mathbb{R}^{k \times N}$ is a training example matrix obtained by (9), the ELM with $\tilde{N}$ hidden nodes and activation function $g(x)$ are mathematically modeled as $H\beta = T$ (3), where H is the hidden layer output matrix of the neural network (4). To solve the linear system is equivalent to finding a least-squares solution $\hat{\beta}$ to satisfy follow equation:

$$||H\hat{\beta} - T||_{\mathrm{F}}^2 = \min_{\beta} ||H\beta - T||_{\mathrm{F}}^2, \tag{10}$$

where $||\cdot||_{\mathrm{F}}$ is the Frobenius norm.

According to the text data, the high dimensional and sparse characteristic might lead to the overfitting problem. The estimated weight $\hat{\beta}$ often have a low bias but large variance such that the model performs well on the training set but poorly on any other set. Regularization [25] is an effective way to deal with this problem by sacrificing a little bias to reduce the variance of the predicted values and hence may improve the overall prediction accuracy.

A lot of regularization methods are used in the linear system, for example, ridge regression [26], lasso [27], elastic net [28], or even the nonconvex regularizer $l_{1/2}$ [29] and minimax concave term [30]. Nevertheless, these approaches, except ridge regression, need an iterative estimated algorithm. In order to keep the advantage that the linear system of ELM can be solved analytically, we use the Frobenius norm as a regularization term and rewrite (10) as

$$||H\hat{\beta} - T||_{\mathrm{F}}^2 = \min_{\beta}(||H\beta - T||_{\mathrm{F}}^2 + \lambda||\beta||_{\mathrm{F}}^2), \tag{11}$$

where λ is a parameter used to control the trade-off between the approximation error and the regularization degree, and $\hat{\beta}$ can be obtained by theorem 1.

Theorem 1 *The minimization problem of Eq.* (11) *has an optimal solution when λ is a positive constant, the solution is given by:*

$$\hat{\beta} = (H^{\mathrm{T}}H + \lambda \mathrm{I})^{-1}H^{\mathrm{T}}T. \quad (12)$$

Proof Let us denote the objective function of (11) by $l(\beta)$, i.e. $l(\beta) = \|H\beta - T\|_{\mathrm{F}}^2 + \lambda \|\beta\|_{\mathrm{F}}^2$. Setting

$$\frac{\mathrm{d}l(\beta)}{d\beta} = 0, \quad (13)$$

we have

$$(H^{\mathrm{T}}H + \lambda \mathrm{I})\beta = H^{\mathrm{T}}T, \quad (14)$$

because $(H^{\mathrm{T}}H + \lambda\ \mathrm{I})$ is an invertible matrix when $\lambda > 0$. So,

$$\beta = (H^{\mathrm{T}}H + \lambda \mathrm{I})^{-1}H^{\mathrm{T}}T. \quad (15)$$

The second-order derivative of $l(\beta)$ w.r.t. β is

$$\frac{\mathrm{d}^2 l(\beta)}{d\beta\beta^{\mathrm{T}}} = 2(H^{\mathrm{T}}H + \lambda \mathrm{I}), \quad (16)$$

it is a positive define matrix when $\lambda > 0$. Therefore, (12) is the optimal solution of (11) when $\lambda > 0$. □

It should be noted that a similar result is also mentioned in [17] from the point of stable view. Recently, Huang et al. [31] did a more general discussion about the constrained optimization based ELM in, and different solutions can be obtained based on the concerns on the efficiency in different size of training datasets.

3.3 Algorithm for text categorization

Generally, ELM focus on the function approximation applications. According to the classification problem, we need some category discrimination function.

Here, we code the category information as the target vector of training set. In order to represent the coding for uni-label or multi-label corpus uniformly, we define the target vector corresponding to a document d as

$$t = (b_1, \ldots, b_i, \ldots, b_m)^{\mathrm{T}}, \quad (17)$$

where m is the number of categories in corpus, and b_i is equal to 1 or -1 depending on whether the related document belongs to the corresponding categories. For example, supposing there are five categories ($m = 5$), a document $d = (w_1, w_2, \ldots, w_k)^{\mathrm{T}}$ belongs to the first and the forth categories, then the related target vector is $t = (1, -1, -1, 1, -1)^{\mathrm{T}}$. For test set, the output target matrix can be evaluated as

$$Y = \tilde{H}\hat{\beta}, \quad (18)$$

where $\tilde{H}$ is the hidden layer output matrix of testing data.

According to the uni-label corpus, we define the category discrimination function as:

Algorithm 1 TC based on RELM

Input: the training and testing set and the setting of parameters

Output: the labels of testing set

Learning stage:

1: evaluating the semantic representation of the training set using Eq. (8) and Eq. (9)

2: generating the input weights and hidden biases randomly

3: evaluating the hidden layer output matrix H with Eq. (4)

4: evaluating the matrix T of the training set with Eq. (17) and (5)

5: evaluating the output weights $\hat{\beta}$ with Eq. (12)

Classification stage:

6: evaluating the semantic representation of the testing set using Eq. (8) and Eq. (9)

7: evaluating the hidden layer output matrix $\tilde{H}$ with Eq. (4)

8: evaluating the output target value Y with Eq. (18)

9: classifying the testing document using Eq. (19) or (20) depending on whether the corpus belongs to uni-label or multi-label

$$Category(d_j) = \arg\max_i(Y_j), \quad (19)$$

where d_j is the jth sample and Y_j is the jth row output vector of Y.

According to the multi-label corpus, we define the category discrimination function as:

$$Category(d_j) = \arg_i(Y_j > \theta), \quad (20)$$

where d_j is the jth sample, Y_j is the jth row output vector of Y, and $\theta = 0$ or can be estimated by cross verification.

Algorithm 1 gives the implementation pseudo code of TC based on RELM.

4 Experiments

This section firstly introduces the datasets used in the experiments, then the evaluation measures of performance are given. The results and analysis are presented finally.

Some commonly used notations are listed in Table 1 for convenience. Moreover, a boldface in a table means better performance when the setting is same.

Table 1 Some commonly used notations in the experiments

Notation	Meaning
mF1	*Micro − averaged* F_1
MF1	*Macro − averaged* F_1
#dim	Dimensionality of documents
#node	The number of hidden nodes
$train	The time cost of training (unit: second)
$test	The time cost of testing (unit: second)

Table 2 Performance comparison between RELM and ELM in Reuters-top10

#dim	#node	RELM ($\lambda = 1$)				ELM			
		$train	$test	mF1	MF1	$train	$test	mF1	MF1
50	100	**0.02**	0.03	91.48	79.07	0.11	0.03	**91.72**	**79.88**
	200	**0.05**	0.04	91.64	79.37	0.24	0.04	**92.16**	**81.5**
	400	**0.15**	0.05	91.81	79.94	0.71	0.05	**92.82**	**83.41**
	1,000	**0.68**	0.08	91.78	79.57	9.72	0.09	**92.64**	**84.74**
	2,000	**2.32**	0.14	**91.95**	**80.36**	74.39	0.16	88.48	79.45
	4,000	**8.75**	0.25	**92.09**	**80.91**	548.95	0.25	57.41	44.23
100	100	**0.07**	0.03	88.24	77.87	0.11	0.03	**90.97**	**82.27**
	200	**0.07**	0.04	92.78	84.26	0.23	0.04	**93.17**	**85.59**
	400	**0.17**	0.06	93.13	85	0.7	0.05	**93.4**	**86.38**
	1,000	**0.69**	0.09	**93.25**	85.41	9.48	0.1	93.23	**86.85**
	2,000	**2.32**	0.15	**93.42**	**85.94**	75.63	0.17	91.88	85.64
	4,000	**8.81**	0.29	**93.61**	**86.32**	653.21	0.29	74.58	62.49
200	100	**0.03**	0.03	69.24	49.66	0.11	0.04	**70.8**	**54.56**
	200	**0.07**	0.04	88.89	81.16	0.25	0.04	**90.12**	**82.97**
	400	**0.18**	0.06	93.66	86.95	0.7	0.06	**93.82**	**87.59**
	1,000	**0.72**	0.11	**93.84**	87.34	8.86	0.11	93.69	**87.53**
	2,000	**2.44**	0.18	**94.05**	**87.96**	73.89	0.21	93.05	87.17
	4,000	**8.97**	0.33	**94.14**	**88.06**	656.87	0.34	81.39	73.11

4.1 Datasets

Two popular TC benchmarks are tested in our experiments: Reuters-21578 and WebKB. The Reuters-21578 dataset[1] is a standard multi-label TC benchmark and contains 135 categories. In our experiments, we use a subset of the data collection which includes the 10 most frequent categories among the 135 topics and we call it Reuters-top10. We divide it into the training and testing 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.

WebKB dataset is a standard uni-label TC benchmark which contains web pages gathered from university computer science departments. We use the subset called WebKB4[3] including four most populous entity-representing categories. After pre-processed procedure, 2,777 training documents and 1,376 testing documents with 7,287 term features are obtained.

4.2 Evaluation measures

In TC, the most commonly used performance measures are recall, precision and their harmonic mean F_1 [1]. Given a specific category c_i from the category space $\{c_1, \ldots, c_m\}$, the corresponding recall (Re_i), precision (Pr_i) are defined by:

$$Re_i = \frac{TP_i}{TP_i + FN_i}, \quad Pr_i = \frac{TP_i}{TP_i + FP_i}, \tag{21}$$

where TP_i (true positives) is the number of documents assigned correctly to class i, FP_i (false positives) is the number of documents that do not belong to class i but are assigned to this class incorrectly and FN_i (false negatives) is the number of documents that actually belong to class i but are not assigned to this class. The corresponding F_{1i} is defined as:

$$F_{1i} = \frac{2 \times Re_i \times Pr_i}{Re_i + Pr_i}. \tag{22}$$

The average performance of a binary classifier over multiple categories is derived from the micro-averaged and the macro-averaged. For micro-averaged, the measures are computed globally without categorical discrimination. The micro-averaged recall $\widehat{Re}^U$ and micro-averaged precision $\widehat{Pr}^U$ are defined as:

$$\widehat{Re}^U = \frac{\sum_{i=1}^{m} |TP_i|}{\sum_{i=1}^{m} (|TP_i| + |FN_i|)}, \quad \widehat{Pr}^U = \frac{\sum_{i=1}^{m} |TP_i|}{\sum_{i=1}^{m} (|TP_i| + |FP_i|)}, \tag{23}$$

and the micro-averaged F_1 is defined as

$$micro - averaged\, F_1 = \frac{2 \times \widehat{Pr}^U \times \widehat{Re}^U}{\widehat{Pr}^U + \widehat{Re}^U}. \tag{24}$$

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

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

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

For macro-averaged, F-measure is computed locally over each category c_i first and then the average over all categories is taken:

$$macro - averaged\, F_1 = \left(\sum_{i}^{m} F_{1i}\right)/m. \quad (25)$$

Table 3 Performance comparison between RELM and ELM in WebKB4

#dim	#node	RELM ($\lambda = 1$)				ELM			
		$train	$test	mF1	MF1	$train	$test	mF1	MF1
70	100	**0.04**	0.02	85.32	83.34	0.14	0.02	**86.63**	**85.08**
	200	**0.03**	0.02	87.19	85.72	0.13	0.02	**87.42**	**85.87**
	400	**0.09**	0.03	**87.8**	**86.4**	0.46	0.03	87.27	85.64
	1,000	**0.35**	0.05	**88.1**	**86.75**	8.69	0.05	84.19	82.65
	2,000	**1.21**	0.09	**88.55**	**87.15**	74.38	0.09	69.69	67.8
	4,000	**4.64**	0.16	**88.87**	**87.47**	202.61	0.15	52.48	50.68
150	100	**0.01**	0.02	**73.8**	69.08	0.11	0.02	73.44	**69.13**
	200	**0.03**	0.02	86.76	85.2	0.15	0.02	**87.02**	**85.45**
	400	**0.09**	0.03	**88.58**	**87.22**	0.44	0.03	88.45	87.16
	1,000	**0.36**	0.05	**88.86**	**87.59**	8.27	0.05	86	84.27
	2,000	**1.24**	0.1	**88.73**	**87.48**	73.49	0.1	75.33	73.07
	4,000	**4.76**	0.18	**88.9**	**87.65**	204.02	0.17	65.89	63.56
300	100	**0.02**	0.02	**60.59**	**51.71**	0.06	0.02	58.85	49.65
	200	**0.04**	0.02	73.8	69.49	0.15	0.02	**74.43**	**69.94**
	400	**0.1**	0.03	86.6	**84.96**	0.56	0.03	**86.66**	84.95
	1,000	**0.41**	0.07	**88.44**	**87.09**	7.26	0.07	87.29	85.83
	2,000	**1.34**	0.13	**87.96**	**86.53**	70.42	0.13	78.75	76.35
	4,000	**4.87**	0.22	**88.03**	**86.46**	208.31	0.2	74.29	72.04

Table 4 Performance comparison between RELM and BP

Dataset	Reuters-top10							
	RELM ($\lambda = 0.5$, #node = 1,000)				BP (#node = 50)			
#dim	$train	$test	mF1	MF1	$train	$test	mF1	MF1
40	**0.68**	0.09	91.21	77	300.17	**0.04**	92.73	83.13
60	**0.68**	0.09	92.21	83.01	361.99	**0.05**	92.39	83.82
80	**0.7**	0.09	93	85.02	641.95	**0.05**	93.13	85.88
100	**0.7**	**0.09**	93.51	85.68	851.81	0.12	92.36	85.9
120	**0.7**	**0.1**	93.41	86.07	1,192.42	0.13	91.41	83.71
140	**0.7**	**0.1**	93.76	87.16	1,933.44	0.14	92.4	84.73
160	**0.72**	**0.1**	93.83	87.24	2,672.61	0.44	91.85	83.43
Dataset	WebKB4							
	RELM ($\lambda = 1$, #node = 700)				BP (#node = 70)			
#dim	$train	$test	mF1	MF1	$train	$test	mF1	MF1
60	**0.15**	0.03	86.26	84.14	236.73	0.03	81.69	79.63
90	**0.18**	0.03	88.79	87.58	516.46	0.03	80.38	78.62
120	**0.18**	0.04	89.18	87.98	571.3	0.03	81.9	79.83
150	**0.19**	0.04	88.86	87.54	717.44	0.04	77.76	75.63
180	**0.19**	0.04	89.16	87.86	823.81	0.04	81.83	80.16
210	**0.2**	0.04	89.22	87.97	1,113.35	0.04	80.45	77.01
240	**0.21**	**0.04**	88.94	87.58	1,161.3	0.05	77.98	75.64

To evaluate the performance overall, we adapt the $micro - averaged\ F_1$ and $macro - averaged\ F_1$ as the performance measures.

4.3 Results and analysis

To verify the performance of RELM, we compare it with the standard ELM [23], back propagation neural network (BP) [5], and SVM [7]. All experiments are carried out in MATLAB 2010a environment running in a 2.8 GHZ CPU and 8 G memory. For each experiment, we run the test 10 times and take their averaged values as the results. Because all experiments take the same representation method (using LSA), we does not count the time cost of the dimensionality reduction. All experiments below using RELM or ELM are taken radial basis function as their active functions.

4.3.1 Comparison with ELM

The performance comparisons between ELM and RELM are presented in Tables 2 and 3. For brief, we only give three situations (about 1, 2, and 5% of the original dimensionality) and others have similar cases. In these tables, the training speed of RELM is much faster than ELM. The performances of RELM increases during the number of hidden nodes increases; however, the performances of ELM become instable and drop dramatically when the number of hidden nodes reaches some certain numbers which might be induced by overfitting. Therefore, RELM obtains more stable performance and possesses the potential to improve performance by increasing the scale of networks.

4.3.2 Comparison with BP

In the BP experiments, we assigned a small number to the number of hidden nodes because the training time increases tremendously as the number of hidden nodes increases, and its performance does not necessarily increases in the same time. Even #node=50, the time cost is remarkable. From Table 4, the training speed of BP is much slower than RELM; however, the performance of BP is inferior to RELM in most cases. So, RLEM is obviously better than BP in our experiments.

Table 5 Performance comparison between RELM and SVM

Dataset	Reuters-top10							
	RELM ($\lambda = 20$, #node = 6,000)				SVM			
#dim	$train	$test	mF1	MF1	$train	$test	mF1	MF1
50	19.96	**0.37**	91.62	79.35	**15.75**	6.25	92.73	82.75
60	19.99	**0.37**	92.4	82.62	**17.54**	6.95	93.48	85.61
70	20.1	**0.39**	92.51	83.53	**19.55**	7.72	93.21	85.51
80	**20.03**	**0.39**	92.86	84.54	21.56	8.53	93.64	86.28
90	**20.07**	**0.41**	93.1	85.04	23.82	9.45	93.53	86.23
100	**20.04**	**0.41**	93.2	85.05	25.95	10.18	93.64	86.91
200	**20.23**	**0.47**	93.76	87.2	49.87	19.72	94.3	89.26
300	**20.43**	**0.53**	94.06	87.75	75.01	30.96	94.31	88.82
400	**20.6**	**0.61**	94.34	88.07	104.54	43.21	94.38	88.54
500	**20.68**	**0.67**	94.16	88.18	134.03	56.29	94.29	88.33
Dataset	WebKB4							
	RELM ($\lambda = 10$, #node = 6,000)				SVM			
#dim	$train	$test	mF1	MF1	$train	$test	mF1	MF1
70	11.18	**0.22**	87.83	86.48	**5.89**	2.99	87.65	86.3
80	11.15	**0.23**	88.46	87.31	**6.61**	3.36	89.03	87.97
90	11.17	**0.23**	88.9	87.63	**7.31**	3.74	89.1	88.12
100	11.18	**0.23**	89.51	88.53	**8.02**	4.1	89.17	88.34
200	**11.3**	**0.26**	89.69	88.55	15.82	8.21	89.83	88.76
300	**11.37**	**0.31**	89.08	87.83	24.07	13.03	88.88	87.46
400	**11.43**	**0.35**	88.62	87.23	32.96	18.4	88.52	86.77
500	**11.55**	**0.39**	88.66	87.31	42.64	24.22	88.52	86.61
600	**11.61**	**0.43**	88.62	86.94	52.6	30.11	89.24	87.12
700	**11.73**	**0.46**	88.31	86.55	64.55	36.92	89.32	87.08

4.3.3 Comparison with SVM

Table 5 presents the comparison results between RELM and SVM in Reuters-top10 and WebKB4. The dimensionality is from about 1 to 10% of the number of the original features. From this table, we can observe that the F_1 performance of RELM is slightly lower than SVM in most cases; however, the speed of RELM is much faster than SVM, especially in the cases that the dimensionality is relatively large.

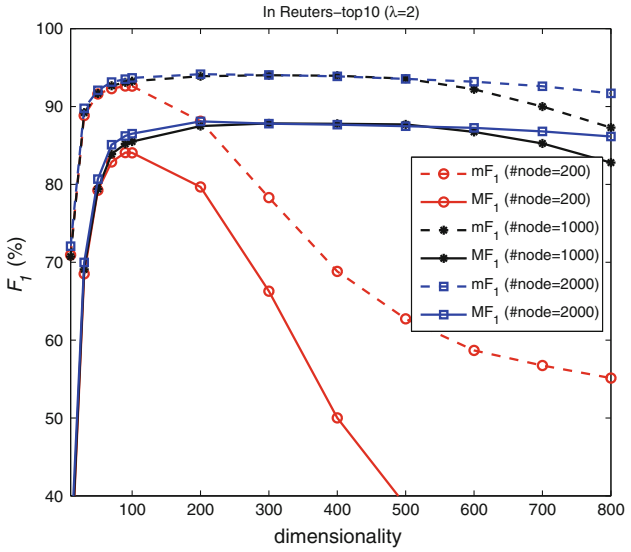

Fig. 1 Performance in Reuters-top10 while the dimensionality varies (the dimensionality can be selected randomly in the suggested interval when #node is relatively large)

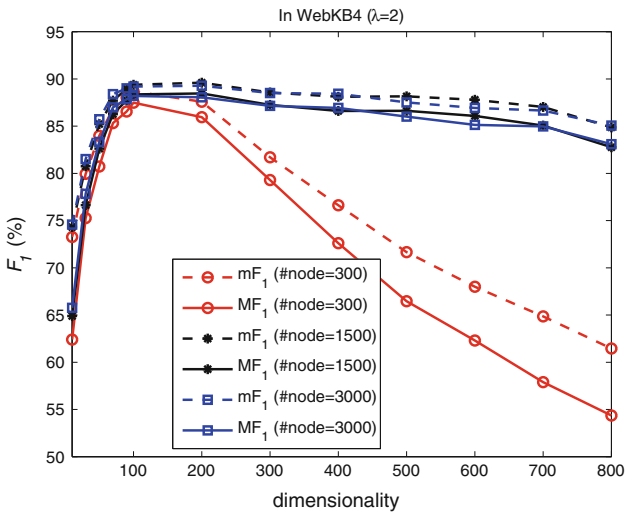

Fig. 2 Performance in WebKB4 while the dimensionality varies (the dimensionality can be selected randomly in the suggested interval when #node is relatively large)

4.3.4 Parameter discussions

The parameters of RELM include the following: the active function, the input dimensionality, the number of hidden nodes, and the regularization factor λ. Although these parameters are also need to be tuned, they are actually very easy to determined for TC. Here, we give some suggestions how to tune these parameters experimentally. Since all experimental results are too many, we only present some typical cases or give the conclusions directly for concise.

According to the active functions selection for TC, a experimental suggestion is: radial basis function $\geq$ triangular basis function $\geq$ sine function $\gg$ sigmoid $\geq$ hard limit function, where $\geq$ means the performance is slightly

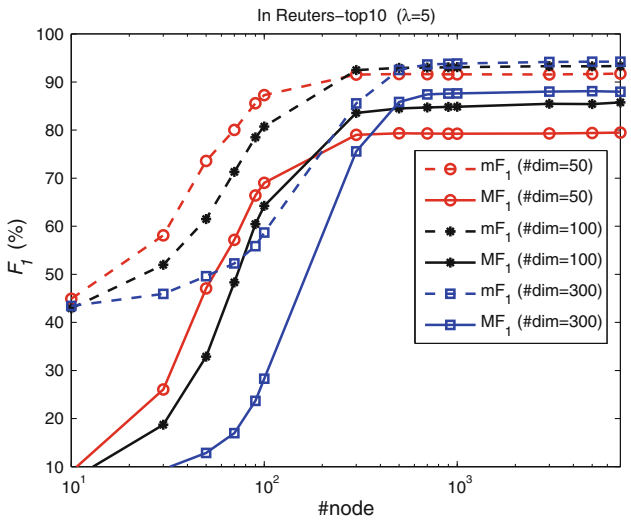

Fig. 3 Performance in Reuters-top10 while #node varies (the performance increases as #node increase until it reaches a stable situation)

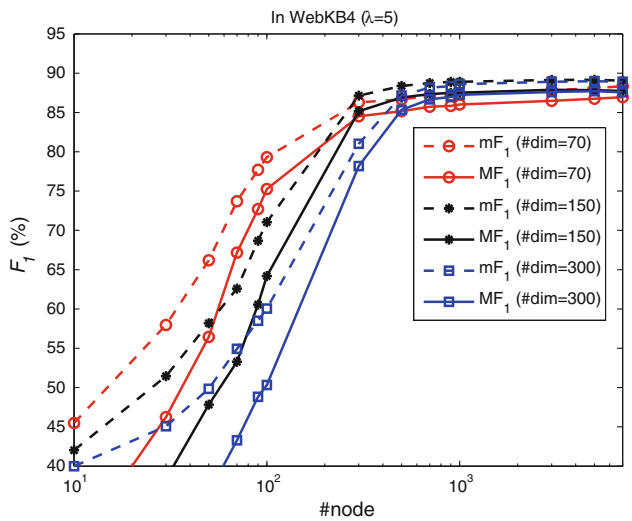

Fig. 4 Performance in WebKB4 while #node varies (the performance increases as #node increase until it reaches a stable situation)

superior and $\gg$ means the performance is considerable superior.

From the experimental results above and Figs. 1 and 2, we can see that the performance is very poor when the dimensionality is exceedingly small (commonly <1% of the original dimensionality) or when it is close to or larger than the number of hidden nodes (these cases can be observed in the lines of #node = 200 in Fig. 1 and in the lines of #node = 300 in Fig. 2). Apart from that, the performance is almost insensitive to the dimensionality which means the dimensionality can be selected randomly in a specific interval (commonly lies in 2–5% of the original dimensionality) when the number of hidden nodes is relatively large enough.

We also observe that the number of hidden nodes is an important factor. In Figs. 3 and 4, the performance increases when the number of hidden nodes increase until the it reaches a stable situation. Therefore, the number of hidden nodes should be take enough large for example greater than 10 times of the input dimensionality.

For the regularization factor λ, an experimental suggestion is $\lambda \in [0.1\ 20]$. The selected rule is: the more scale of RELM the larger λ should be taken, i.e. λ should be increased while the dimensionality and the number of hidden nodes increase. Generally, λ could be selected in interval $[0.1\ 1]$ when the scale of RELM is small (for example, the cases #dim = 50 in Fig. 5 or #dim = 70 in Fig. 6), and λ could be selected in interval $[10\ 20]$ when the scale of RELM is relatively large (for example, the cases #dim = 150 in Fig. 5 or #dim = 300 in Fig. 6).

In sum, the radial basis function and triangular basis function are suggested to be taken as the active function for TC, and the input dimensionality could be selected randomly in our propositional interval (commonly lies in 2–5% of the original dimensionality). Moreover, the number of hidden nodes should be relatively large (commonly greater than 10 times of the input dimensionality), and λ should be tuned in $[0.1\ 20]$ according to the rule: the more scale of RELM the larger λ.

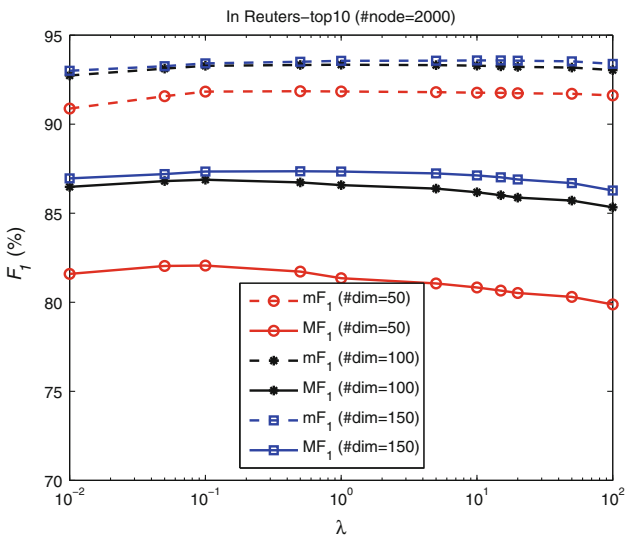

Fig. 5 Performance in Reuters-top10 while λ varies (λ can be selected in the suggested interval according to the scale of RELM)

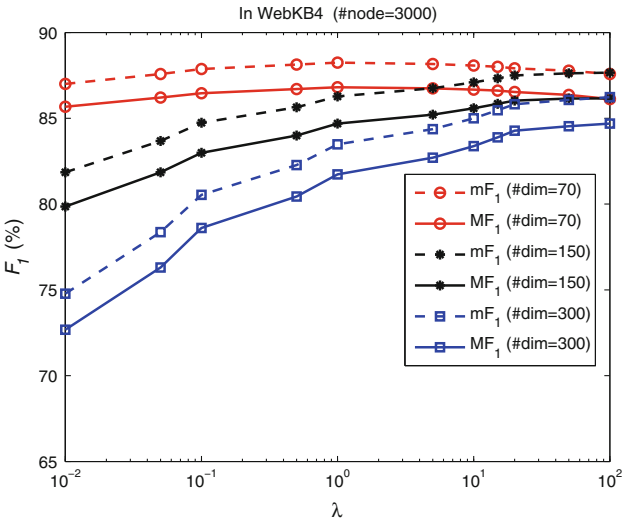

Fig. 6 Performance in WebKB4 while λ varies (λ can be selected in the suggested interval according to the scale of RELM)

5 Conclusions

In this article, a regularization ELM is presented and its analytical solution and theoretical proof are given simultaneously. Moreover, a TC framework combining the LSA and RELM is proposed, and an algorithm including uni-label and multi-label classification for TC is developed. The experimental results show that the proposed method can produce good performance in most cases and can learn faster than conventional popular learning algorithms such as feedforward neural networks or support vector machine.

The features of the proposed approach include much faster learning and classification speed, ease of implementation, and least human intervene. It might become a promising technique for TC and its applications such as news section classification, quick text retrieval, and real-time topic tracking.

For the further researches, we are trying to incorporate cognitive information into RELM to improve the classification performance.

Acknowledgments The authors are grateful to anonymous reviewers for their constructive suggestions. This work was supported by the 973 Program (Grant No. 2012CB316400), the National Natural Science Foundation of China (Grant No. 61171151), and the Natural Science Foundation of Zhejiang Province (Grant No. Y6110147 and Y1110342).

References

1. Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv 34(1):1–47
2. Lewis DD, Ringuette M (1994) A comparison of two learning algorithms for text categorization. In: Third annual symposium on document analysis and information retrieval, vol 33. Citeseer, pp 81–93
3. Soucy P, Mineau GW (2001) A simple knn algorithm for text categorization. In: IEEE international conference on data mining, pp 647–648
4. Ng HT, Goh WB, Low KL (1997) Feature selection, perceptron learning, and a usability case study for text categorization. In: 20th Annual international ACM SIGIR conference on research and development in information retrieval, pp 67–73
5. Wang W, Yu B (2009) Text categorization based on combination of modified back propagation neural network and latent semantic analysis. Neural Comput Appl 18(8):875–881
6. De Souza AF, Pedroni F, Oliveira E, Ciarelli PM, Henrique WF, Veronese L, Badue C (2009) Automated multi-label text categorization with vg-ram weightless neural networks. Neurocomputing 72(10–12):2209–2217
7. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: 10th European Conference on Machine Learning, pp 137–142
8. Gabrilovich E, Markovitch S (2004) Text categorization with many redundant features: Using aggressive feature selection to make svms competitive with c4. 5. In: Proceedings of the twenty-first international conference on Machine learning, pp 321–328
9. Genkin A, Lewis DD, Madigan D (2007) Large-scale bayesian logistic regression for text categorization. Technometrics 49: 291–304
10. Aseervatham S, Antoniadis A, Gaussier E, Burlet M, Denneulin Y (2011) A sparse version of the ridge logistic regression for large-scale text categorization. Pattern Recognit Lett 32:101–106
11. Hmeidi I, Hawashin B, El-Qawasmeh E (2008) Performance of knn and svm classifiers on full word arabic articles. Adv Eng Inform 22(1):106–111
12. Zhao M, Ren J, Ji L, Fu C, Li J, Zhou M (2011) Parameter selection of support vector machines and genetic algorithm based on change area search. Neural Comput Appl (in press)
13. Anand R, Mehrotra K, Mohan CK, Ranka S (1995) Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Netw 6(1):117–124
14. Man Z, Wu HR, Liu S, Yu X (2006) A new adaptive backpropagation algorithm based on lyapunov stability theory for neural networks. IEEE Trans Neural Netw 17(6):1580–1591
15. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the IEEE international joint conference on neural networks, vol 2, pp 985–990
16. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
17. Huang GB, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122
18. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recognit 38(10):1759–1763
19. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407
20. Huang GB, Ding X, Zhou H (2010) Optimization method based extreme learning machine for classification. Neurocomputing 74:155–163
21. Nakayama M, Shimizu Y (2003) Subject categorization for web educational resources using mlp. In: Proceedings of 11th European symposium on artificial neural networks. Citeseer, pp 9–14
22. Tsimboukakis N, Tambouratzis G (2010) A comparative study on authorship attribution classification tasks using both neural network and statistical methods. Neural Comput Appl 19(4): 573–582
23. Liu Y, Loh HT, Tor SB (2005) Comparison of extreme learning machine with support vector machine for text classification. Innov Appl Artif Intell 3533:390–399
24. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manag 24(5):513–523
25. Tikhonov A (1963) Solution of incorrectly formulated problems and the regularization method. Sov Math Dokl 4:1035–1038
26. Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12:55–67
27. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Series B (Methodol) 58:267–288
28. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc Series B (Stat Methodol) 67(2): 301–320
29. Qian Y, Jia S, Zhou J, Robles-Kelly A (2011) Hyperspectral unmixing via $l_{1/2}$ sparsity-constrained nonnegative matrix factorization. IEEE Trans Geosci Remote Sens 99:1–16
30. Dai G, Wang J, Shi J, Ren X, Zhang Z (2011) A non-convex relaxation approach to sparse dictionary learning. In: International conference on computer vision and pattern recognition, pp 1809–1816
31. Huang GB, Zhou H, Ding X, Zhang R (2011) Extreme learning machine for regression and multi-class classification. IEEE Trans Syst Man Cybern Part B (in press)