Text Classification Method Based on Cost-Sensitive Ensemble Extreme Learning Machine Postprint

Authors: Li Ming, Xiao Peilun, Zhang Ju, Gu Xinmeng

Date: 2018-09-27T00:00:00+00:00

Abstract

Weighted Extreme Learning Machine assigns different weights to samples from different categories, thereby improving classification accuracy to a certain extent; however, it only considers inter-category sample differences while neglecting sample noise and intra-category sample variations. This paper proposes an Extreme Learning Machine ensemble method based on text category information entropy. The method utilizes Adaboost.M1 as the algorithmic framework, generates text category information entropy through intra-class distribution entropy and inter-class distribution entropy of the text, constructs a cost-sensitive matrix from the text category information entropy, and integrates cost-sensitive Extreme Learning Machine into the Adaboost.M1 framework. Experimental results demonstrate that this method achieves superior accuracy and generalization performance compared with other types of Extreme Learning Machine.

Full Text

Ensemble Extreme Learning Machine Based on Cost Sensitivity for Text Classification

Li Ming[1],[2], Xiao Peilun[2],[3], Gu Xinmeng

[1](High Performance Computing Application R&D Center, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China)
[2](Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, Beijing 100084, China)
[3](The University of Edinburgh, College of Science & Engineering, Edinburgh EH1, England)
(Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract

The weighted extreme learning machine (WELM) assigns different weights to samples from different categories, which improves classification accuracy to some extent. However, WELM only considers differences between samples from different categories while neglecting sample noise and differences among samples within the same category. This paper proposes an ensemble extreme learning machine method based on text category information entropy. Using Adaboost.M1 as the algorithmic framework, we generate text category information entropy through intra-class and inter-class distribution entropy of the text, construct a cost-sensitive matrix from this entropy, and integrate cost-sensitive extreme learning machine into the Adaboost.M1 framework. Experimental results demonstrate that the proposed method achieves better accuracy and generalization compared to other types of extreme learning machines.

Keywords: extreme learning machine, ensemble learning, Adaboost.M1, text classification, cost-sensitive

1 Introduction

Text classification refers to the process of automatically labeling text categories using computers within a given classification system. As a key technology in text mining, text classification is widely applied in information retrieval, search engines, question-answering systems, public opinion analysis, sentiment analysis, and other domains. With the rapid development of network technology, the number of web pages has grown exponentially, necessitating more accurate and efficient text classification technologies for effective and personalized information retrieval.

Currently mature text classification methods include: k-nearest neighbor (K-NN) [1-2], Naive Bayes [3], Decision Tree [4], Maximum Entropy [5-6], Support Vector Machine (SVM) [2-7], Neural Networks [8-9], and Fuzzy Theory [10].

Huang et al. proposed a novel single-hidden layer feedforward neural network called Extreme Learning Machine (ELM) [11]. In this algorithm, the connection weights between the input layer and hidden layer, as well as the thresholds, are randomly generated and require no adjustment during model training. Only the number of hidden layer neurons needs to be set, avoiding many problems associated with gradient descent-based learning methods such as falling into local minima and slow convergence [12]. Compared with traditional neural networks, ELM possesses equivalent global approximation capability, higher accuracy, and a simpler model. ELM also demonstrates clear advantages over other traditional machine learning methods, offering faster learning speed and better generalization performance. Ying Liu et al. compared the performance of ELM and SVM for text classification [13]; Wenbin Zheng et al. used latent semantic analysis for text dimensionality reduction and compared regularized extreme learning machine (RELM), neural networks, and SVM for text classification, with RELM showing faster learning speed and better classification performance [14]. Sub-

sequently, Wenbin Zheng et al. proposed a fast text classification framework combining a linear classifier based on non-negative matrix factorization with an ELM-based classifier [15]. Xiangguo Zhao et al. proposed an XML document classification framework based on ELM, improved voting-ELM in this framework, and successfully applied REV and RCC methods to v-ELM, achieving better results than voting-ELM [16]. Later, Xiangguo Zhao continued to improve this method by introducing an parameter in the RCC method to enhance re-voting accuracy and performed probability statistics on ELM voting results, further improving classification effectiveness [17]. Lijuan Duan used KELM to classify historical patent documents, achieving better results compared to SVM [18]. Yu Haiyan et al. performed dimensionality reduction on text features through information gain and introduced wavelet kernels to apply KELM for Chinese text sentiment classification [19]. Li Yongqiang proposed a CPSO-ELM algorithm by optimizing search strategies to select input weights and biases of hidden nodes in single-hidden layer feedforward neural networks for XML document classification [20]. Rajendra Kumar Roul et al. studied the improvement of ELM classification performance through feature extraction techniques, conducted extensive experiments on single ELM and multi-layer ELM in text classification, and achieved results surpassing many state-of-the-art methods including SVM [21]. Subsequently, Rajendra Kumar Roul proposed a text classification feature selection algorithm based on K-means clustering, combined with Wordnet to reduce text dimensionality, and applied it to single ELM and multi-layer ELM [22].

However, the ELM algorithm has certain limitations. Its prediction performance is still affected by connection weights, thresholds, and the number of hidden layer nodes [23]. Since the input connection weights and hidden layer thresholds are randomly initialized, executing the algorithm multiple times on the same training and test samples may yield varying results, indicating suboptimal model stability. Ensemble learning can effectively improve prediction performance compared to single models [24]. Adaboost is an important ensemble learning technique that can enhance weak learners with prediction accuracy only slightly better than random guessing into strong learners with high prediction accuracy. Yunliang Jiang et al. performed dimensionality reduction on facial features through PCA and embedded ELM into the Adaboost framework for face recognition [25]. Huang Haibo et al. proposed an ELM algorithm based on Adaboost, extracted abnormal sound feature information of shock absorbers through wavelet packet decomposition, and predicted the sound quality of shock absorber abnormal noise [26]. Yan Xu et al. applied the Adaboost-based ELM method to traffic sign recognition [27]. Kuan Li et al. proposed a boosting weighted ELM method that seamlessly embeds weighted ELM into boosting ELM, adjusting sample distribution weights during each Adaboost iteration. However, this method only considers inter-class imbalance in datasets without considering intra-class imbalance, which also significantly impacts classification performance [28].

This paper proposes a novel ELM model combining cost-sensitive ELM under

the Adaboost.M1 framework and applies it to text classification. First, we use word vectors to obtain high-quality, low-dimensional text feature vectors. Then, we construct a cost-sensitive matrix using text category information entropy and employ cost-sensitive weighted ELM as the base classifier, adjusting sample distribution through cost-sensitive factors in the multi-class Adaboost.M1 framework. Finally, we compare the accuracy and generalization performance of ELM, Voting-ELM, Adaboost-WELM, and three methods under our proposed Adaboost-WELM framework (AE1-WELM, AE2-WELM, and AE3-WELM) on three standard text datasets: 20newsgroups, Reuters52, and Webkb. Experimental verification demonstrates that the AE3-WELM algorithm exhibits more significant classification performance, stability, and generalization compared to other ELM methods.

2.1 Weighted Extreme Learning Machine

Weighted Extreme Learning Machine (WELM) introduces a weight matrix $\mathbf{W}$ into ELM to weight each sample, reducing potential imbalance between sample classes and thereby improving overall recognition rate. According to KKT theory:

$$\mathbf{L}_{\mathrm{WELM}} = \frac{1}{2}\|\beta\|^2 + \frac{C}{2}\sum_{i=1}^{N} w_i \|\xi_i\|^2$$

where $\mathbf{W}$ is a diagonal matrix with each diagonal element representing the weight value of a sample. W. Zong et al. empirically proposed two weighting schemes [28]:

Scheme 1: $w_i = \frac{1}{N_{t_i}}$

Scheme 2: $w_i = \frac{1}{N_{t_i}}$

However, WELM simply uses the sample numbers of majority and minority classes to assign weights, giving larger weights to minority class samples but equal weights to samples within the same class. This only considers inter-class imbalance without addressing intra-class imbalance, which also significantly impacts classification performance. Kuan Li et al. improved upon this by adopting different update methods for weights of different class samples, but similarly failed to consider weight differences among samples within the same class [28].

2.2 Adaboost-Based Weighted Extreme Learning Machine

The basic idea of the AdaBoost algorithm is to combine several weak classifiers according to certain rules into a strong classifier with powerful classification capability. Freund and Schapire improved the original Adaboost designed for binary classification problems, generating Adaboost.M1 and Adaboost.M2 algorithms for multi-class problems while also providing an extended form of

Adaboost.M1. This paper uses the extended form of Adaboost.M1 [29], then embeds the weighted extreme learning machine from Section 2.1 into the Adaboost.M1 framework to generate the Adaboost-based weighted extreme learning machine algorithm. The algorithm uses $h_m(x)$ to represent the m-th weak classifier and combines weak classifiers through weak classifier weights α_m to obtain a strong classifier. The main steps of the algorithm are as follows:

Step 1: Initialize sample weights using Equation (2):

$$D_1(i) = \frac{1}{N_{k_t}}$$

where k_t represents the class containing sample i and N_{k_t} denotes the number of samples in class k_t.

Step 2: Normalize sample weights:

$$D_m(i) = \frac{D_m(i)}{\sum_{i=1}^{N} D_m(i)}$$

Step 3: For $m = 1$ to M (where M is the number of weak classifiers): 1. Train samples using the weak learning algorithm to obtain weak classifier $h_m(x)$ 2. Calculate the classification error rate ϵ_m of $h_m(x)$:

$$\epsilon_m = \sum_{i=1}^{N} D_m(i) \cdot \mathbb{I}(h_m(x_i) \neq y_i)$$

3. When the classifier error rate is greater than 0 and less than 0.5, update sample weights according to Equation (5); otherwise exit the loop:

$$D_{m+1}(i) = \frac{D_m(i)}{Z_m} \times \begin{cases} e^{-\alpha_m} & \text{if } h_m(x_i) = y_i \\ e^{\alpha_m} & \text{if } h_m(x_i) \neq y_i \end{cases}$$

where Z_m is the normalization factor.

Step 4: The combined classifier output is:

$$H(x) = \arg\max_{y \in \mathcal{Y}} \sum_{m=1}^{M} \alpha_m \cdot \mathbb{I}(h_m(x) = y)$$

3.1 Document Vector Generation Based on Word Vectors

Massive training samples and excessively high vector dimensionality are characteristics of text classification. Overly high-dimensional feature sets increase the computational burden of extreme learning machines. Traditional approaches reduce the dimensionality of the text vector space and decrease interference from noisy information to ensure text classification accuracy. Word vector text representation achieves better feature expression than traditional manually extracted

feature vectors. Word vectors map each word to a low-dimensional real vector by training unlabeled corpora [30], describing semantic similarity between words through distances between low-dimensional real vectors while effectively avoiding the curse of dimensionality in feature vectors. Mikolov et al. proposed two word vector learning models: CBOW (Continuous Bag of Words) and Skip-gram [31].

The Skip-gram model uses the current word as input to a log-linear classifier to predict words in the context. Given a word sequence $\{w_1, w_2, ..., w_N\}$ where N is the sequence length, in the Skip-gram network structure, inputting the i-th word w_i from the word sequence uses the current word w_i to predict context with window size b. The objective function maximized by the Skip-gram model is shown in Equation (8):

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{-b \leq j \leq b, j \neq 0} \log p(w_{i+j}|w_i)$$

The softmax function used to calculate $p(w_{i+j}|w_i)$ defined by the Skip-gram model is shown in Equation (9):

$$p(w_{i+j}|w_i) = \frac{\exp(\mathbf{c}_{i+j}^T \mathbf{c}_i)}{\sum_{k=1}^{V} \exp(\mathbf{c}_k^T \mathbf{c}_i)}$$

where $\mathbf{c}_{i+j}$ and $\mathbf{c}_i$ are the word vectors of w_{i+j} and w_i, respectively.

The Skip-gram model demonstrates good performance in both text similarity measurement and text classification tasks. The word vector model adopted in this paper is the Skip-gram model. We first generate word vectors for feature words, where d represents the dimension of word vectors. We use $\mathbf{c}_{i,j}$ to denote the word vector of the j-th word in the i-th document, and generate document vectors through Equation (10):

$$\mathbf{v}_i = \frac{1}{J_i} \sum_{j=1}^{J_i} \mathbf{c}_{i,j}$$

where J_i represents the number of words in the i-th document.

3.2 Category Information Entropy

Typically, to improve text classification performance, researchers focus on two aspects: first, improving classification algorithms (or learning models); second, improving text data representation models. Traditionally, we represent text through the Vector Space Model (VSM) [32], which represents each text document as a vector composed of weight values of a certain number of feature words before classification. Feature words appear in texts of different categories with certain uncertainty, which can be measured using entropy.

The weight of feature words should be allocated according to their importance in text classification, and the importance of feature words is reflected in their category discriminative power, as words with strong category discriminative power help distinguish texts of different categories. We adopt Shannon' s information entropy to measure the ability of feature words to distinguish categories and obtain the definition of the inter-class information entropy function for feature words.

Definition 1: If the training set contains m categories, feature word t_i appears in category c_j $(j = 1, 2, ..., m)$ with frequency $D_{i,j}^F$, and appears in all documents with frequency D_i^F, then the inter-class information entropy function of feature word t_i (denoted as $E_D(t_i)$) is defined as:

$$E_D(t_i) = -\sum_{j=1}^{m} \frac{D_{i,j}^F}{D_i^F} \log_2 \left(\frac{D_{i,j}^F}{D_i^F} \right)$$

According to the definition of information entropy and the maximum entropy theorem, the more uniformly feature word t_i is distributed across categories, the larger the entropy value $E_D(t_i)$. Conversely, the more non-uniformly t_i is distributed across categories, the smaller $E_D(t_i)$. The entropy value $E_D(t_i)$ reaches its maximum if and only if t_i is uniformly distributed across all categories. Therefore, we take the reciprocal of $E_D(t_i)$ and add a parameter θ to prevent division by zero.

Through experiments, we found that the reciprocal of document inter-class distribution entropy, $\frac{1}{E_D(t_i)+\theta}$, is usually large. Although the differences in inter-class information entropy between documents are significant and can characterize document distribution, they overwhelm the values of subsequent intra-class information entropy, making the effect of intra-class information entropy on document distribution completely invisible. Therefore, we take the logarithm of $\frac{1}{E_D(t_i)+\theta}$ to obtain the final inter-class information entropy of the text.

Definition 2: If the training set contains m categories, feature word t_i appears in k documents of category c_j $(j = 1, 2, ..., m)$ with frequency $TF(t_i, d_k)$, then the intra-class information entropy function of feature word t_i (denoted as $E_C(t_i)$) is defined as:

$$E_C(t_i) = \max_{1 \le j \le m} \left\{ -\sum_{k=1}^{n_j} \frac{TF(t_i, d_k)}{TF(t_i, c_j)} \log_2 \left(\frac{TF(t_i, d_k)}{TF(t_i, c_j)} \right) \right\}$$

where $TF(t_i, d_k)$ represents the frequency of feature word t_i in the k-th document d_k of category c_j; $TF(t_i, c_j)$ represents the total frequency of feature t_i in texts of category c_j; and n_j represents the number of documents in category c_j. For category c_j $(j = 1, 2, ..., m)$, we take the maximum value as the intra-class information entropy of feature word t_i.

Considering that some feature words with high category discriminative ability are low-frequency words, and low-frequency words have low intra-class information entropy that is almost zero, directly using $[1, \frac{1}{E_C(t_i)}]$ as the intra-class information entropy function would lead to incorrect information measurement results after combining inter-class and intra-class information entropy. Moreover, experiments show that the feature information entropy function with e as an adjustment factor demonstrates better and more significant text discriminative ability than directly using $\frac{1}{E_C(t_i)}$. Combining the inter-class information entropy function and intra-class information entropy of feature words yields the category information entropy function:

$$EDC(t_i) = \log\left(\frac{1}{E_D(t_i) + \theta}\right) \times \log(e + E_C(t_i))$$

A larger category information entropy value of a feature word indicates more obvious category discrimination effect, implying high discriminative power and importance for text classification. The category information entropy of feature words not only characterizes text distribution between different categories but also characterizes text distribution within the same category, providing a relatively fine-grained description of texts.

The main steps of the document category information entropy generation algorithm are as follows:

Step 1: Preprocess texts d_i $(i \leq N)$ in the text set $\mathcal{D}$.
Step 2: After preprocessing, each text d_i in the text set is represented as a set of feature words $\{t_1, t_2, ..., t_{nd}\}$.
Step 3: Calculate the inter-class information entropy for each feature word in the training documents:
- Compute D_i^F and $D_{i,j}^F$ values for feature word t_i in document d_i;
- Calculate $E_D(t_i)$ for feature word t_i in document d_i according to Equation (12).
Step 4: Calculate the intra-class information entropy for each feature word in the training documents:
- Compute $TF(t_i, d_k)$, $TF(t_i, c_j)$, and $\max TF(t_i, c_j)$ values for feature word t_i;
- Calculate $E_C(t_i)$ for feature word t_i according to Equation (14).
Step 5: Calculate $EDC(t_i)$ for feature word t_i in the document according to Equation (13).
Step 6: Normalize $EDC(t_i)$.
Step 7: Calculate the EDC value for each document in the text set:

$$EDC(d_i) = \sum_{j=1}^{nd} EDC(t_j)$$

3.3 Cost-Sensitive Ensemble Extreme Learning Machine

Both weighted extreme learning machine and boosting weighted extreme learning machine fail to consider weight differences among samples of the same class. To further improve ELM classification performance, we introduce cost sensitivity into ELM by assigning different weights w_i $(i = 1, 2, ..., N)$ to different samples. We construct the cost-sensitive matrix $\mathbf{C}$ using text category information entropy. The output parameter β of cost-sensitive weighted extreme learning machine is:

$$\beta = (\mathbf{H}^T\mathbf{C}\mathbf{H} + \frac{\mathbf{I}}{C})^{-1}\mathbf{H}^T\mathbf{C}\mathbf{T}$$

The Adaboost.M1 algorithm adjusts sample distribution by adaptively sampling training samples and adjusting sample weights. It assigns larger weights to misclassified samples and smaller weights to correctly classified samples. Such weight allocation reflects the importance of each sample from the perspective of misclassification rate. However, the importance of different samples within the same category also varies significantly. We utilize the weight allocation idea of the Adaboost.M1 algorithm in iterations, introducing cost-sensitive factors into the Adaboost.M1 framework and seamlessly integrating weighted extreme learning machine into the cost-sensitive Adaboost.M1 framework. We characterize the importance of each sample for category discrimination through text category information entropy, construct cost-sensitive factors from text category information entropy, and update sample weights according to the importance of each sample during each iteration. Based on different training sample weight update methods, we denote them as AE1-WELM, AE2-WELM, and AE3-WELM. These three methods are collectively referred to as AEx-WELM methods in this paper. The main steps of the algorithm are:

Step 1: Preprocess training and test datasets by removing stop words and special symbols; each document d_i in the text set is represented as a set of feature words $\{t_1, t_2, ..., t_{nd}\}$.
Step 2: Generate word vectors for feature words using Word2vec.
Step 3: Represent each document d_i with document vector $\mathbf{v}_i = \sum_{j=1}^{J_i} \mathbf{c}_{i,j}$.
Step 4: Generate cost-sensitive matrix $\mathbf{C} = \text{diag}(c_1, c_2, ..., c_N)$.
Step 5: Train weighted ELM with weight matrix $\mathbf{W}_t$ as weak classifier $h_m(x)$.
Step 6: Initialize training document weight distribution $D_1(i) = \frac{1}{N}$.
Step 7: For $m = 1$ to T (where T is the number of weak classifiers):
1. Calculate classification error rate ϵ_m of $h_m(x)$: same as Equation (4).
2. When classifier error rate is greater than 0 and less than 0.5, update sample weights according to Equation (14); otherwise exit the loop:

AE1-WELM:

$$D_{m+1}(i) = \frac{D_m(i)}{Z_m} \times \begin{cases} e^{-EDC(x_i)} & \text{if } h_m(x_i) = y_i \\ e^{EDC(x_i)} & \text{if } h_m(x_i) \neq y_i \end{cases}$$

AE2-WELM:

$$D_{m+1}(i) = \frac{D_m(i)}{Z_m} \times \begin{cases} e^{-EDC(x_i)/e} & \text{if } h_m(x_i) = y_i \\ e^{EDC(x_i)/e} & \text{if } h_m(x_i) \neq y_i \end{cases}$$

AE3-WELM:

$$D_{m+1}(i) = \frac{D_m(i) \cdot EDC(x_i)}{Z_m} \times \begin{cases} e^{-\alpha_m} & \text{if } h_m(x_i) = y_i \\ e^{\alpha_m} & \text{if } h_m(x_i) \neq y_i \end{cases}$$

Normalization of $D_{m+1}(i)$ values is the same as Equation (6), and α_m values are the same as Equation (7).

Step 8: Given test sample x, output the category label:

$$H(x) = \arg\max_{y \in Y} \sum_{m=1}^{T} \alpha_m \cdot \mathbb{I}(h_m(x) = y)$$

The AE2-WELM and AE3-WELM algorithms are essentially consistent with the AE1-WELM algorithm, with the difference lying in sample weight updates. The method using Equation (14a) is AE1-WELM, using Equation (14b) is AE2-WELM, and using Equation (14c) is AE3-WELM.

4.1 Experimental Datasets

We evaluate all models on three standard text datasets: the 20 Newsgroups dataset collected by Ken Lang[1], the WebKB dataset collected by the CMU project[2], and the Reuters52 dataset released by David D. Lewis[3]. The first is a balanced dataset, while the latter two are imbalanced datasets.

The 20Newsgroups corpus contains 20 different categories of English news with a total of 18,846 documents. To improve experimental reliability, all duplicate files were removed, leaving 11,293 documents for training and 7,528 documents for testing. The original WebKB corpus contains approximately 8,300 English websites divided into 7 categories. We selected the 4 most commonly used categories: student, faculty, course, and project, comprising 4,199 documents. Similarly, to improve reliability, duplicate files were removed, leaving 2,756 documents for training and 1,375 documents for testing. The Reuters52 dataset uses the 52 most frequently used categories from the 90 categories in the Reuters21578 dataset, called the R52 dataset. The R52 dataset contains 9,100 documents total, with 6,532 for training and 2,568 for testing. We first preprocess the datasets by removing stop words, eliminating single characters and non-alphabetic symbols, converting uppercase letters to lowercase, performing stemming, and removing low-frequency words. We use the word vector training tool word2vec provided by Google for word vector model training .

4.2 Evaluation Metrics

Precision, Recall, and F1-score are widely used for classification effectiveness evaluation. Micro-average and Macro-average are two methods for global evaluation of classification results: Micro-average first calculates classification results for all documents and then averages across all documents; Macro-average first calculates classification results for each category and then averages across all categories. The specific definitions are as follows:

Macro-average Precision and Recall:

$$P_{\text{macro}} = \frac{1}{m}\sum_{i=1}^{m}\frac{a_i}{b_i}, \quad R_{\text{macro}} = \frac{1}{m}\sum_{i=1}^{m}\frac{a_i}{d_i}$$

Micro-average Precision and Recall:

$$P_{\text{micro}} = \frac{\sum_{i=1}^{m} a_i}{\sum_{i=1}^{m} b_i}, \quad R_{\text{micro}} = \frac{\sum_{i=1}^{m} a_i}{\sum_{i=1}^{m} d_i}$$

where b_i is the number of documents in class c_i in the test set; a_i is the number of documents correctly classified as class c_i; and d_i is the number of documents belonging to class c_i. Micro-average tends to favor majority classes, while Macro-average tends to favor minority classes. To provide an overall performance measure for classification, this paper uses the micro-average of F1-score for evaluation.

4.3 Experimental Results Analysis

(1) Parameter Settings

All experiments run on a 2.4GHz CPU with 4GB memory. ELM-related algorithms are implemented in Python. Each experiment runs 10 times, with the average taken as the result. Text input dimension is varied between 50 and 500 dimensions. Common activation functions for extreme learning machines include Sigmoid, RBF, and tanh. Through performance comparison of these activation functions, we select the tanh function as the activation function for hidden nodes. The hyperparameters c and L of the extreme learning machine are selected using grid search, with c values searched in $\{10^{-5}, 10^{-4}, ..., 10^{5}\}$ and L searched in $\{100, 200, ..., 1000\}$.

(2) Performance Comparison

To verify the performance of AEx-WELM, we compare it with ELM, Voting-ELM, and Adaboost-WELM on three standard datasets. To examine performance under varying text dimensions, we select different text dimensions and compare changes in micro-average F1 (mf1), macro-average F1 (MF1), training

time, and testing time. Detailed results are shown in Tables -, where data represent optimal performance values at the best (c, L) settings. From Tables 1-3, we observe that ELM performs worst in all tests. On all three standard datasets, AE1-WELM and AE3-WELM achieve higher MF1 values than all other ELM methods. On the 20newsgroups and Reuters52 datasets, AE1-WELM and AE3-WELM demonstrate significantly higher mf1 values than other ELM methods, indicating better performance. This demonstrates that integrating cost-sensitive extreme learning machine into the Adaboost framework is effective. On imbalanced datasets Reuters52 and Webkb, AE1-WELM and AE3-WELM significantly outperform ELM, Adaboost-WELM, and AE2-WELM, showing that the proposed method can improve classification effectiveness for imbalanced multi-class problems. On the balanced dataset 20newsgroups, the performance improvement of AE1-WELM and AE3-WELM is even more pronounced, surpassing all other ELM methods. Moreover, on all three standard text datasets, AE1-WELM and AE3-WELM exhibit stable performance, demonstrating good generalization capability. Among the three AEx-WELM methods, AE2-WELM performs worst, with performance almost similar to Adaboost-WELM. The distribution weights in AE2-WELM change dramatically during iterations, which may be why its classification effect is not as good as the other two methods. Comparing AE1-WELM and AE3-WELM, the latter shows superior classification performance and stability as text dimension increases, making AE3-WELM the recommended method.

AEx-WELM vs. ELM: Figures [Figure 1: see original paper]-[Figure 3: see original paper] show that for all datasets, the four methods embedding extreme learning machine into the Adaboost framework (Adaboost-WELM, AE1-WELM, AE2-WELM, AE3-WELM) achieve significantly improved classification performance over ELM, with AE1-WELM and AE3-WELM showing more pronounced advantages. On all three standard datasets, the comprehensive metric MF1 values of AE1-WELM and AE3-WELM exceed those of ELM and other ELM variants, with particularly notable improvements on 20newsgroups (balanced) and Reuters52 (imbalanced). On the Webkb dataset (imbalanced), the mf1 values are slightly lower than Voting-ELM, but MF1 values remain higher than Voting-ELM.

AEx-WELM vs. Voting-ELM: Figures 1-3 show that on the balanced 20newsgroups dataset, AE1-ELM and AE3-WELM are slightly better than Voting-ELM, while on the imbalanced Reuters52 dataset, Voting-ELM performs much worse than AE1-ELM and AE3-WELM. However, on the imbalanced Webkb dataset, Voting-ELM achieves better results than all other ELM methods. This is because Webkb is a small dataset, and our experiments only use 4 common categories, resulting in a small number of classes. Since mf1 tends to favor majority classes, the mf1 values of AE1-ELM and AE3-WELM are slightly lower than Voting-ELM. This also relates to the random sampling algorithm used in Voting-ELM that fully exploits the diversity of text features in the dataset. Therefore, Voting-ELM demonstrates good performance across all three text datasets. Nevertheless, on Webkb, the MF1 values of AE1-ELM

and AE3-WELM are still slightly higher than Voting-ELM. Moreover, on larger datasets like 20newsgroups and Reuters52, Voting-ELM' s performance 明显下降, failing to achieve as significant classification effects as AE1-ELM and AE3-WELM.

Text information expression based on word vectors also enriches text feature representation to a certain extent, not only effectively reducing text dimensionality but also achieving classification performance at low dimensions (typically 100 dimensions) comparable to traditional VSM feature expression at 1000 dimensions (or even higher). Additionally, we observe from Figures 1-3 that the performance of all six ELM methods declines after text dimension exceeds 400, indicating that excessively high dimensions not only burden extreme learning machines but also increase noise and affect classification performance.

Figures [Figure 4: see original paper]-[Figure 6: see original paper] show the variation of AE3-WELM mf1 values with hidden nodes and c values when text vectors are 300-dimensional on three standard text datasets. We observe that while more hidden nodes help AE3-WELM achieve better classification results, mf1 values become relatively stable after the number of hidden nodes exceeds 400, with larger hidden node numbers having minimal impact. When hidden nodes reach a certain level, classification performance becomes unstable, and performance decreases with increasing hidden nodes beyond 800, likely due to overfitting. Figures 4-6 also show that the regularization parameter c is a crucial factor, with performance reaching a stable value as c decreases. The impact of regularization parameter c on performance is greater than that of hidden nodes. When c is small (close to 10^{-5}), AE3-WELM is not sensitive to the selection of hidden node parameters.

Figures [Figure 7: see original paper]-[Figure 9: see original paper] show the variation in model training time for six ELM methods with text vector dimensionality when text vectors are 300-dimensional on three standard text datasets. Since Voting-ELM needs to integrate multiple sub-classifiers and Adaboost-WELM and AEx-WELM methods require many iterations, their training time consumption is much longer than ELM. AEx-WELM and Adaboost-WELM training times are significantly longer than ELM and also somewhat longer than Voting-ELM. However, we observe an important detail: Voting-ELM' s training time is a multiple of ELM' s, with this multiple depending on the number of ensemble classifiers used in Voting-ELM, so this training time increase is linear. Although AEx-WELM and Adaboost-WELM training times are much longer than ELM, and the scale of training time increase is related to Adaboost iteration count, we find that this iteration count does not increase linearly with text vector dimension and hidden nodes. That is, the overall training time increase is sub-linear, which on the Reuters52 dataset results in the phenomenon that the four Adaboost-based ELM methods and Voting-ELM become increasingly close as text vector dimension increases. Among the three AEx-WELM methods (AE1-WELM, AE2-WELM, AE3-WELM), training and testing times are essentially consistent with no significant differences, and no method shows

obvious advantage.

Traditional VSM text expression produces high-dimensional sparse text features that increase the computational burden of extreme learning machines. To address this issue, this paper adopts word vector models as text expression methods. We measure sample importance through text category information entropy and generate cost-sensitive matrices and cost-sensitive factors from the perspective of sample importance, integrating cost-sensitive extreme learning machine into the Adaboost.M1 framework to improve text classification performance. Experiments demonstrate that on both imbalanced and balanced datasets, the comprehensive classification performance metrics of AE1-WELM and AE3-WELM outperform other types of extreme learning machines, with AE3-WELM showing overall better performance than AE1-WELM. In future work, we will investigate how to further reduce text feature dimensionality on the basis of word vectors, select more reasonable cost-sensitive functions to further reduce computational cost of AEx-WELM, and further optimize the AEx-WELM framework to achieve better text classification performance.

References

[1] Samworth R J. Optimal weighted nearest neighbour classifiers[J]. The Annals of Statistics, 2012, 40(5): 2733-2763.
[2] Zhang H, Berg A C, Maire M, et al. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition[C]. Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, 2006: 2126-2136.
[3] Chen J, Huang H, Tian S, et al. Feature selection for text classification with Naïve Bayes[J]. Expert Systems with Applications, 2009, 36(3): 5432-5435.
[4] Takahashi F, Abe S. Decision-tree-based multiclass support vector machines[C]. Neural Information Processing, 2002. ICONIP' 02. Proceedings of the 9th International Conference on, 2002.
[5] Liu W, Song N. A fuzzy approach to classification of text documents[J]. Journal of Computer Science and Technology, 2003, 18(5): 640-647.
[6] Widyantoro D H, Yen J. A fuzzy similarity approach in text classification task[C]. Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on, 2000: 653-658.
[7] Chang C-C, Lin C-J. LIBSVM: a library for support vector machines[J]. ACM transactions on intelligent systems and technology (TIST), 2011, 2(3): 27.
[8] Ghiassi M, Olschimke M, Moon B, et al. Automated text classification using a dynamic artificial neural network model[J]. Expert Systems with Applications, 2012, 39(12): 10967-10976.
[9] Lam H-K, Ekong U, Liu H, et al. A study of neural-network-based classifiers for material classification[J]. Neurocomputing, 2014, 144: 367-377.
[10] Bigi B. Using Kullback-Leibler distance for text categorization[C]. European Conference on Information Retrieval, 2003: 305-319.

[11] Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: theory and applications[J]. Neurocomputing, 2006, 70(1): 489-501.
[12] Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE transactions on Evolutionary Computation, 2009, 13(2).
[13] Liu Y, Loh H, Tor S. Comparison of extreme learning machine with support vector machine for text classification[J]. Innovations in Applied Artificial Intelligence, 2005: 390-399.
[14] Zheng W, Qian Y, Lu H. Text categorization based on regularization extreme learning machine[J]. Neural Computing and Applications, 2013, 22(3-4): 447-456.
[15] Zheng W, Tang H, Qian Y. Collaborative work with linear classifier and extreme learning machine for fast text categorization[J]. World Wide Web, 2015, 18(2): 235-252.
[16] Zhao X-G, Wang G, Bi X, et al. XML document classification based on ELM[J]. Neurocomputing, 2011, 74(16): 2444-2451.
[17] Zhao X, Bi X, Qiao B. Probability based voting extreme learning machine for multiclass XML documents classification[J]. World Wide Web, 2014, 17(5): 1217-1231.
[18] Duan L, Yuan B, Wu C, et al.: Text-image separation and indexing in historic patent document image based on extreme learning machine, Proceedings of ELM-2014 Volume 2: Springer, 2015.
[19] Yu H, Chen L, Zheng W. Chinese text sentiment classification based on kernel extreme learning machines[J]. Journal of China University of Metrology, 2016, 2: 020.
[20] Li Yongqiang. Research and Application of XML Classification Based on Extreme learning Machine with Particle Swarm Optimization[D]. Northeastern University, 2013.
[21] Roul R K, Nanda A, Patel V, et al. Extreme learning machines in the field of text classification[C]. Software Engineering, Artificial Intelligence, Networking Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on, 2015.
[22] Roul R K, Sahay S K. K-means and Wordnet Based Feature Selection Combined with Extreme Learning Machines for Text Classification[C]. International Conference on Distributed Computing and Internet Technology, 2016: 103-112.
[23] Lu Hui-Juan, An Chun-Lin, Ma Xiao-Ping, et al. Disagreement measure Based Ensemble of Extreme learning Machine for Gene Expression Data Classification[J]. Chinese Journal of Computers. 2013, (2): 341-348.
[24] Ditterrich T. Machine learning research: four current direction[J]. Artificial Intelligence Magazine, 1997, 4: 97-136.
[25] Jiang Y, Shen Y, Liu Y, et al. Multiclass AdaBoost ELM and its application in LBP based face recognition[J]. Mathematical Problems in Engineering, 2015, 2015.
[26] Huang Hai-bo, Li Ren-xian, Huang Xiao-rong, et al. Prediction of a suspension shock absorber' s sound metric based on sample entropy and

ELM-adaboost[J]. Journal of Vibration and Shock, 2016, (13).
[27] Xu Y, Wang Q, Wei Z, et al. Traffic sign recognition based on weighted ELM and AdaBoost[J]. Electronics Letters, 2016, 52(24): 1988-1990.
[28] Li K, Kong X, Lu Z, et al. Boosting weighted ELM for imbalanced learning[J]. Neurocomputing, 2014, 128: 15-21.
[29] Freund Y, Schapire R E. A decision-theoretic generalization of on-line learning and an application to boosting[C]. European conference on computational learning theory, 1995: 23-37.
[30] Turian J, Ratinov L, Bengio Y. Word representations: a simple and general method for semi-supervised learning[C]. Proceedings of the 48th annual meeting of the association for computational linguistics, 2010: 384-394.
[31] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.
[32] Sebastiani F. Machine learning in automated text categorization[J]. ACM computing surveys (CSUR), 2002, 34(1): 1-47.

(Corresponding author: Li Ming, E-mail: liming@magicalthink.com)

Author Contribution Statement:
Li Ming: Proposed research ideas, designed research plan, conducted experiments, drafted paper;
Xiao Peilun: Data acquisition, provision and analysis, conducted experiments;
Gu Xinmeng: Conducted experiments;
Zhang Ju: Revised final version of paper.

---

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo20/www/data/news20.html
[2] http://web.ist.ult.pt/~acardoso
[3] https://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
https://code.google.com/p/word2vec

Note: Figure translations are in progress. See original paper for figures.