NODE SLICING BROAD LEARNING SYSTEM FOR TEXT CLASSIFICATION

Fagui Liu, Xinjie Wu, Chao Li

South China University of Technology, Guangzhou

ABSTRACT

Text classification is playing an increasingly important role in natural language processing (NLP). Most research adopts deep structure neural networks to achieve text classification tasks. However, deep structure networks often suffer from time-consuming trainning process and hardware dependence. In this paper, a flat network called broad learning system (BLS) is employed to derive a novel learning method — node slicing broad learning system (NSBLS). Firstly, one-to-one correspondence between the words and the feature node groups is established to obtain a feature layer with rich words, on the basic of which the enhancement layer is generated representing the global information. Then we activate some nodes in the feature node groups, compact them with the enhancement layer and use ridge regression to obtain multiple outputs. Finally, an intergration BLS layer is used to correct and combine the multiple outputs to get the final output. The experiment shows that NSBLS has good performance on several datasets.

Index Terms— text classification, broad learning system, node slicing, feature node groups, enhancement layer

1. INTRODUCTION

Text classification is a typical and important task in natural language processing (NLP) techniques. In general, the key words or words of importance contained in sentences greatly affect the category of the text. Key words make a significant contribution to the classification task, while others play a supplementary role. Therefore, one effective way to achieve good performance in text classification is to extract these words of importance.

With the development of deep learning, deep structured networks have been frequently applied in text classification [1, 2]. Various approaches such as convolutional neural network (CNN) [3, 4] and recurrent neural network (RNN) [5, 6] have been presented to provide solutions. Irsoy at [7] proposed a text analysis model based on CNN, which can identify the negative vocabulary and double negative sentence structure in the sentence. Socher R at [8] proposed a novel encode-decoder algorithm based on RNN, in which the encoder is responsible for converting the input into a vector, while the decoder decodes the input information to restore the output sequence. Tang [9] et al. combined CNN and GRU to learn sentence representation from word embeddings and adaptively encode the sentence semantics and their internal relations. However, deep structure networks offen suffer from a time-consuming training process because of a large number of connecting parameters in filters and layers.

To tackle this problem, Chen at [10] proposed a kind of flat network called BLS, which is inspired by random vector functional-link neural network (RVFLNN) [11]. There is no need for back propagation in the learning process of BLS. Instead, The model adopts pseudo-inverse to achieve one-step learning. However, the original BLS model mapped entire samples to feature nodes, which cannot accurately capture words information for text tasks. Zhang [12] proposed BMT-Net that fully explores appropriate feature information in depth and breadth, in which BLS is applied to downstream tasks. However, BMT-Net focuses more on the exploration of pretrained language model and doesn't conduct in-depth research on the flat network structure. Though these networks do better in training consumption compared to deep learning networks, we think that there is still room for improvement in BLS-based models in the field of text classification.

In this paper, we propose a BLS-based model suitable for text classification. In the model, each word in text corresponds to multiple feature nodes, which we call a feature node group. The enhancement layer is generated by these feature node groups and represent global information. Different from the traditional flat networks, we no longer compact all the feature nodes into a feature layer for learning, since it causes huge computing power and time-consuming during counting pseudo-inverse. Some nodes in each feature group are activated and participate in the pseudo-inverse operation together with the enhancement layer. We use a matrix to describe this process. Each matrix describes a slice of feature nodes. Multiple processes are carried out in parallel, and the same number of outputs are obtained. These output results are input into the integrated BLS layer and get the final results. Our model has achieved good results in both training time and test results and the contributions are:

(1) Aiming at text classification, this paper proposes a BLS-based model to train efficiently.

(2) A feature layer with rich words information is introduced to learn words of importance. At the same time, a global enhancement layer is generated through the feature

layer to learn the relationship between words in the text.

(3) Multiple feature node slices replace the huge feature layer, so that the pseudo-inverse can be obtained more easily and conveniently. In addition, the node slicing module supports parallel operations.

(4) We use an integrated BLS layer to combine and correct the learning results of multiple slices, so that the performance of the model is improved.

2. RELATED WORKS

The flat network model can learn different information simultaneously [10, 13]. Each tiled subnetwork learns a different kind of information, and then the outputs of all subnetworks are connected to the output layer simultaneously [14]. One effective approach to improve performance is to construct corresponding network structures to learn the key information in the samples. Chen at [15] proposed a BLS structure based on cascaded nodes. The cascaded structure allows the network to learn different depth node information. Feng at [16, 17] proposed a fuzzy BLS. The model integrates BLS and TS fuzzy systems, and uses fuzzy nodes to replace the original feature nodes. Z.shi at [18] proposed a multiview BLS, which captured data information from multiple perspectives through independent enhancement layers and shared enhancement layers.

In addition, two more aspects need to be considered except the structure of the flat network. One is that it would be difficult to count pseudo-inverse if the input layer size was too large, the other is how to correct the deviation during the one-step learning process. The former research includes Yang [19] and Yu [20]. Yang at [19] used principal component analysis (PCA) to capture more concise information, but PCA can limit effect when the feature layer size is large enough. Yu at [20] proposed a CNN-based model which compresses the data size through a pooling layer. The latter, such as Stacked BLS [21], correct the learning error of the model based on the principle of residual error. In this paper, we slice the feature layer to reduce the time-consuming of pseudo-inverse, and use an integrated BLS layer to correct errors and unify the slicing results.

3. PROPOSED METHODS

3.1. feature extractor

Given a dataset$\left\{x^i, y^i\right\}$, where $i = 1$ to N, N is the number of samples, x denotes each input data, y denotes corresponding label. Let $X = \left[x^i\right] \in \mathbb{R}^{N\times M\times E}$ denote the input data matrix, where M is the model input length and also the fixed text length, E is the size of the word vector. Let $Y = \left[y^i\right] \in \mathbb{R}^{N\times L}$ denote the one-hot output matrix, where L is the total number of label classes.

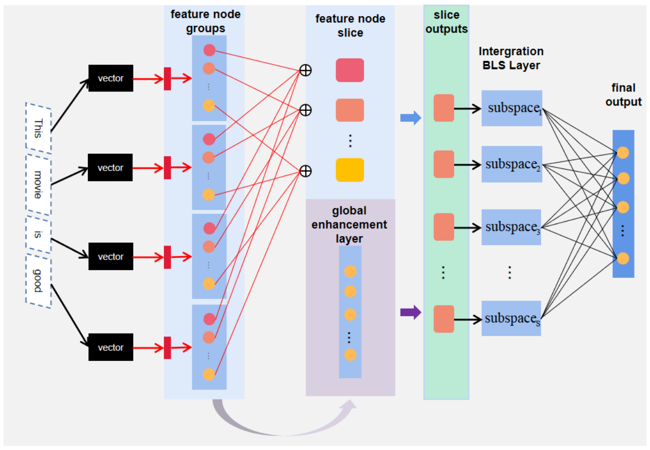

Fig. 1. The decomposition of NSBLS. The structure indicated by red arrows and lines supports parallel computing. The model parts (feature node groups, feature node slice, slice outputs, integration BLS Layer, etc.) are per labeled.

For each word or word vector, we map it to a new space called a feature node. In order to obtain a feature representation from different perspectives, one word or word vector corresponds to multiple feature nodes called feature nodes group. Let $G_m = \left\{{F_m}^j\right\}$ be a feature nodes group, where $j = 1$ to J, J is the number of nodes in each group. Let ${F_m}^j = f\left(X_m W_{mj} + \beta_{mj}\right)$ be a feature node, where $m = 1$ to M, f is the activation function, W_{mj}, β_{mj} are randomly generated weights and biases.

We denote $G^m = \left[G_1, G_2, ..., G_m\right]$, which is the concatenation of m groups of feature nodes. Similarly, the nth group of enhancement nodes, $g\left(G^m W_{h_n} + \beta_{h_n}\right)$ is denoted as H_n, where g is also the activation function and can be the same as f. The global enhancement layer is denoted as $H^n = [H_1, H_2, ..., H_n]$. G^m aims to capture words of importance due to the one-to-one mapping, while H^m is designed to learn the global imformation such as relation of words.

3.2. feature node slice

In a flat network, pseudoinverse can be considered as a convenient approach to train the model. We denote P the input layer, then we can calculate the learnable weight W through the following objective function [15]:

$$\underset{W}{\arg\min} : \|PW - Y\|_2^2 + \lambda \|W\|_2^2 \tag{1}$$

Where λ is the constraint of the square weight W. Increasing λ can alleviate the problem of model overfitting. By using ridge regression we can obtain the formula:

$$W = (\lambda P + P^T P)^{-1} P^T Y \tag{2}$$

Now we know the formula for calculating the weight W of P. To get P, to compact all feature node groups and enhancement layer is an approach, but it is not efficient. The huge size of the input layer P can lead to complex pseudo-inverse calculations and insufficient learning capabilities. Thus we apply feature node slicing to solve the problem. In our method, the feature node groups participate in the calculation locally, and some nodes in each group are activated. In node groups with J nodes in the M groups, we activate k nodes in each group before a slicing operation, and k is set to be divisible by J. The activated nodes in one slice are no longer be in another. We introduce a binary matrix $A \in \mathbb{R}^{M\times J}$ to record the activation history. If a_{mj} in A is 1, the jth node in the mth group is activated. If a_{mj} is 0, it means that the node does not participate in the calculation. For all matrices $A_1, A_2, ..., A_s$, there is $\sum_{s=1}^{S} A_s = [1]_{M\times J}$. Let $T^m = G^m_{A_s}$ denote a slice feature layer, then there is $P = [T^m|H^n]$. A slice operation can be expressed as follows:

$$\begin{aligned} SO_s &= [T^m|H^n]\, W_s \\ &= [f\left(X_m W_{mj_1} + \beta_{mj_1}\right), \\ &\qquad ..., f\left(X_m W_{mj_m} + \beta_{mj_m}\right)|g\left(G^m W_{h_1} + \beta_{h_1}\right), \\ &\qquad ..., g\left(G^m W_{h_n} + \beta_{h_n}\right)] W_s \end{aligned} \tag{3}$$

s number of slices can get a set of outputs $SO^s = \{SO_1, SO_2, ..., SO_s\}$ and a set of binary matrices $A^s = \{A_1, A_2, ..., A_s\}$.

3.3. intergration BLS Layer

Through the slicing operation, we get multiple outputs $SO^s \in \mathbb{R}^{N\times L}$. Here, we propose intergration BLS layer where the added BLS are trained to combine the outputs and approximate the residuals of the last few BLS networks. For a slice output SO^s, we map it to a sub-network space, which is denoted as $Z_s = \theta\left(SO_s W_{o_s} + \beta_{o_s}\right)$. θ is a radial basis function and W_s, β_s are generated using the sparse autoencoder in [10]. We merge the sub-network spaces by compacting them so that we have $Z^s = \{Z_1, Z_2, ..., Z_s\}$. Finally, Formula 2 will be called to obtain the weight W_f of Z^s.

3.4. eval model

The eval process of the model is similar to the training process, except that the weights in the eval model no longer need to be regenerated or calculated by pseudo-inverse, but all come from the training model. The eval model can be expressed as

$$Y_f{}^{test} = [Z_1{}^{test}, Z_2{}^{test}, ..., Z_s{}^{test}] W_f \tag{4}$$

where Z_s^{test} is the sub-network space mapped by the eval data slice output, and the matrices A_s determine which feature nodes will be activated. details of the training process is showed in Algorithm 1.

Algorithm 1: The training process of NSBLS

```
Data: The input data matric:$\boldsymbol{X}$,and the label
      matric:$\boldsymbol{Y}$
Result: $\boldsymbol{W^s}, \boldsymbol{W_f}$
for $m \leftarrow 1$ to $M$ do
    for $j \leftarrow 1$ to $J$ do
        Random $\boldsymbol{W_{mj}}, \boldsymbol{\beta_{mj}}$ ;
        $\boldsymbol{F_m{}^j} \leftarrow \boldsymbol{f}\left(\boldsymbol{X_m W_{mj}} + \boldsymbol{\beta_{mj}}\right)$;
    end
    $\boldsymbol{G_m} \leftarrow \left\{\boldsymbol{F_m{}^j}\right\}$;
end
$\boldsymbol{G^m} \leftarrow [\boldsymbol{G_1}, \boldsymbol{G_2}, ..., \boldsymbol{G_m}]$;
for $n \leftarrow 1$ to $N$ do
    Random $\boldsymbol{W_{h_n}}, \boldsymbol{\beta_{h_n}}$ ;
    $\boldsymbol{H_n} \leftarrow \boldsymbol{g}\left(\boldsymbol{G^m W_{h_n}} + \boldsymbol{\beta_{h_n}}\right)$
end
$\boldsymbol{H^n} \leftarrow [\boldsymbol{H_1}, \boldsymbol{H_2}, ..., \boldsymbol{H_n}]$;
for $s \leftarrow 1$ to $J/k$ do
    Choose $k$ nodes from each $\boldsymbol{G_m}$ and get $\boldsymbol{A_s}$;
    $\boldsymbol{T^m} \leftarrow \boldsymbol{G^m_{A_s}}$;
    Call formula (2) to calculate $\boldsymbol{W_s}$;
    $\boldsymbol{SO_s} \leftarrow [\boldsymbol{T^m}|\boldsymbol{H^n}]\, \boldsymbol{W_s}$;
    Use sparse autoencoder to generate $\boldsymbol{W_{o_s}}, \boldsymbol{\beta_{o_s}}$;
    $\boldsymbol{Z_s} \leftarrow \boldsymbol{\theta}\left(\boldsymbol{SO_s W_{o_s}} + \boldsymbol{\beta_{o_s}}\right)$
end
$\boldsymbol{W^s} \leftarrow [\boldsymbol{W_1}, \boldsymbol{W_2}, ..., \boldsymbol{W_s}]$;
$\boldsymbol{A^s} \leftarrow [\boldsymbol{A_1}, \boldsymbol{A_2}, ..., \boldsymbol{A_s}]$;
$\boldsymbol{Z^s} \leftarrow [\boldsymbol{Z_1}, \boldsymbol{Z_2}, ..., \boldsymbol{Z_s}]$;
Call formula (2) to calculate $\boldsymbol{W_f}$;
```

Datasets	train	test	L	Max	V
RS	20336	8716	3	20	50
ETCR	16050	6879	2	50	100
TS	10000	5000	3	40	50
R8	5485	2189	8	50	20

Table 1. Summary of datasets. ”train” and “test” refer to the number of labeled data in training and test partitions respectively. L represents the number of label categories, max represents the maximum number of words in each sample in the dataset, and V stands for vector size, the length of the word vector.

4. EXPERIMENTS

4.1. dataset and baselines

We validate our proposed method on four datasets. Reddit sentiment (RS) consists of reviews collected on the Reddit and labeled as negative, objective, and positive. Emotions Towards COVID-19 on Reddit (ETCR) consists of some com-

NSBLS	RS			ETCR			TS			R8		
	Acc	F1	Train	Acc	F1	Train	Acc	F1	Train	Acc	F1	Train
-NS+ENH	61.21	54.46	302.48	86.67	86.37	59.15	65.04	64.58	25.77	88.16	61.84	7.05
+NS-ENH	72.11	64.19	**32.66**	88.37	87.82	**9.56**	73.96	73.12	**6.35**	90.63	58.91	**3.71**
+NS+ENH	**72.79**	**65.04**	48.57	**88.62**	**87.94**	17.89	**74.16**	**73.31**	9.13	**91.41**	**63.61**	4.96

Table 2. Results of ablation experiments over ETCR, RS, TS, R8. NS means the feature nodes slicing and ENH is the enhancement layer. Acc is an abbreviation for accuracy, while F1 for F1score and Train for training time. The unit of training time is s. The best results on each dataset are set to **bold**.

Models	RS			ETCR			TS			R8		
	Acc	F1	Train	Acc	F1	Train	Acc	F1	Train	Acc	F1	Train
NSBLS	**72.79**	**65.04**	48.57	**88.62**	**87.94**	17.89	**74.16**	**73.31**	9.13	**91.41**	**63.61**	4.96
ELM	53.2	28.61	**1.09**	83.77	55.69	**1.18**	57.44	28.25	**0.729**	71.12	33.07	**0.41**
CNN-BLS	68.94	60.42	4.71	84.26	83.93	4.01	68.28	68.08	2.01	79.34	43.87	0.91
LSTM-BLS	68.8	61.34	6.84	87.97	87.69	1.64	71.38	71.09	9.31	90.49	61.96	8.94
CFE+Stacked BLS	69.11	56.69	24.12	87.73	87.51	14.04	72.24	72.06	4.978	90.36	61.16	11.4

Table 3. Results of NSBLS with the compared baselines over ETCR, RS, TS, R8. Acc is an abbreviation for accuracy, while F1 for F1score and Train for training time. The unit of training time is s. The best results on each dataset are set to **bold**.

ments on COVID-19 collected on Reddit, and its labels are negative and positive. Twitter sentiment (TS) consists of reviews collected on the Twitter and labeled as negative, objective, and positive. R8 is a dataset for single-label text classification and it is very skewed. Details of the dataset can be found in Table 1. We use the classic word2vect [22, 23] to transform the words into the vector, and the model is pretrained for each dataset separately. The length of the word vector is shown in Table 1.

We conducted experiments on a server with two Intel(R) Xeon(R) CPU E5-2680 v4, 2.40GHz. In the ablation experiment, we discuss the node segmentation module and enhancement layer module of the model. The number of nodes J in each node group is set to 6, and 1 node is activated in each slice. In addition, the number of enhancement layer nodes n is 10. The parameter settings of the complete module are retained while the corresponding module is eliminated. As a comparative experiment, ELM [24], CNN-BLS [19], LSTM-BLS [14], CFEBLS [15] combined with stacked BLS [21] are chosen. All the networks mentioned above accept the data with same specifications, and execute classification task.

4.2. results and discussion

Table 2 shows that the accuracy and F1score of the model reduces while the node slicing module is eliminated. Especially on the datasets RS and TS, the accuracy of module improves by 11.58% and 9.12% respectively. The huge improvement comes from the fact that a single BLS module does not have the ability to fit large-scale and complex data. It also proves that it is feasible to slice the feature layer and integrate the results of multiple slices operations. Furthermore, feature node slicing greatly improves the training speed of the model, and this improvement is mostly in the process of counting pseudo-inverse. The enhancement layer module slightly improves the performance of the model . However, adding an enhancement layer module does not create a big burden on time consumption, which we think is worthwhile.

Table 3 shows the comparison results between NSBLS and several other models. The accuracy of NSBLS on RS, ETCR, TS, R8 are 72.79%, 88.64%, 74.16%, 91.41% and F1score are 65.04%, 73.31%, 87.94%, 63.61% respectively. The two indicators of NSBLS are the best among several models that also use word2vect. In terms of training time, NSBLS and other BLS-based models require less than 1 minute of training time. NSBLS spends more time when the scale of dataset is large.

Overall, abundant feature nodes and enhancement layers improve the fitting ability of the model, and in order to solve the time consumption caused by feature nodes groups, the feature node slicing module is introduced. Furthermore, the integration layer is set to improve the model performance.

5. CONCLUSION

We propose a model called NSBLS for text input. The model can achieve a second-level training speed in a ordinary CPU powered environment. When combining the feature node groups, the feature node slice module and the intergration BLS layer, NSBLS performs better than several other BLS-based text tasks.

6. REFERENCES

[1] Zi-Yi Dou, "Capturing user and product information for document level sentiment analysis with deep memory network," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 521–526.

[2] Yichun Yin, Yangqiu Song, and Ming Zhang, "Document-level multi-aspect sentiment classification as machine comprehension," in *Proceedings of the 2017 conference on empirical methods in natural language processing*, 2017, pp. 2044–2054.

[3] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, vol. 6, 2014.

[4] Xiang Zhang, Junbo Zhao, and Yann LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015.

[5] Barak A Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 263–269, 1989.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[7] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[8] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[9] Duyu Tang, Bing Qin, and Ting Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1422–1432.

[10] CL Philip Chen and Zhulin Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 1, pp. 10–24, 2017.

[11] Y-H Pao and Yoshiyasu Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.

[12] Tong Zhang, Xinrong Gong, and CL Philip Chen, "Bmt-net: Broad multitask transformer network for sentiment analysis," *IEEE Transactions on Cybernetics*, 2021.

[13] Xinrong Gong, Tong Zhang, CL Philip Chen, and Zhulin Liu, "Research review for broad learning system: Algorithms, theory, and applications," *IEEE Transactions on Cybernetics*, pp. 1–29, March 2021.

[14] Jie Du, Chi-Man Vong, and CL Philip Chen, "Novel efficient rnn and lstm-like architectures: Recurrent and gated broad learning systems and their applications for text classification," *IEEE transactions on cybernetics*, vol. 51, no. 3, pp. 1586–1597, 2020.

[15] CL Philip Chen, Zhulin Liu, and Shuang Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 4, pp. 1191–1204, 2018.

[16] Shuang Feng and CL Philip Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE transactions on cybernetics*, vol. 50, no. 2, pp. 414–424, 2018.

[17] Shuang Feng and CL Philip Chen, "Nonlinear system identification using a simplified fuzzy broad learning system: Stability analysis and a comparative study," *Neurocomputing*, vol. 337, pp. 274–286, 2019.

[18] Zhenhua Shi, Xiaomo Chen, Changming Zhao, He He, Veit Stuphorn, and Dongrui Wu, "Multi-view broad learning system for primate oculomotor decision decoding," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 9, pp. 1908–1920, 2020.

[19] Fan Yang, "A cnn-based broad learning system," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. IEEE, 2018, pp. 2105–2109.

[20] Wanke Yu and Chunhui Zhao, "Broad convolutional neural network based industrial process fault diagnosis with incremental learning capability," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 6, pp. 5081–5091, 2019.

[21] Zhulin Liu, CL Philip Chen, Shuang Feng, Qiying Feng, and Tong Zhang, "Stacked broad learning system: From incremental flatted structure to deep model," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 209–222, 2020.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[24] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.